ТЕСТИРОВАНИЕ МОБИЛЬНЫХ ПРИЛОЖЕНИЙ*, РАЗРАБОТКА ПОД IOS*, ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ*, БЛОГ КОМПАНИИ «DIGITAL SECURITY»

Динамический анализ iOS-приложений без Jailbreak

ansjdnakjdnajkd 19 октября 2017 в 11:11 7,6k

В рамках данной статьи мы хотим поделиться своим опытом решения некоторых проблем, связанных с анализом безопасности iOS-приложений. Рассмотрение будет осуществляться при условии, что у нас нет iOS-устройства с JailBreak на борту.

Статический анализ vs. динамический анализ

Начнем наше погружение с типов анализа и их небольшого сравнения. Статический анализ позволяет идентифицировать большое количество проблем: от оставленных данных учетных записей до различного рода “закладок” и уязвимостей. Говоря про iOS-приложения, отметим, что полезными в этом деле будут IDA Pro, Hopper Disassembler, MobSF или Radare2. Плюсом подхода является возможность проведения автосканирования файлов, кода, в том числе, неиспользуемого с относительно полным его покрытием.

Динамический анализ помогает узнать, как ведет себя приложение

непосредственно во время работы/выполнения, например, изучить состояние программы в тот или иной момент выполнения (содержимое памяти, регистров, значение переменных), изучить логику работы и т.д.

Использование этих двух подходов позволяет наиболее качественно провести работу по анализу приложения, покрыть весь attack-surface и найти максимально возможное количество ошибок и уязвимостей.

Критерии	Статический анализ	Динамический анализ
Код vs. Данные	Проблемы	Нет проблем
Покрытие кода	Большое (но не весь)	Один путь
Данные о значениях	Нет информации	Вся информация
Самомодифицируемый код	Проблемы	Нет проблем
Уязвимости среды выполнения	Нет	Да
Неиспользуемый код	Анализируемый	Не анализируемый

Динамический анализ

Динамический анализ мобильных приложений невозможен без устройства или, в худшем случае, эмулятора. Реальные устройства почти всегда предпочтительнее (если нет большой необходимости в какой-нибудь параллелизации задачи). Но наличие реального устройства выходит и дороже для самого исследователя… Также необходима простая возможность установки на устройство собственных приложений и инструментов для проведения анализа приложений.

Ситуация с Android

Исследуя Android-приложения, зачастую можно даже обойтись старой и дешевой версией устройства или эмулятором в среде разработке. На Android существует простая возможность повысить привелегии пользователя, иными словами “получить root доступ”, для установки на него собственных инструментов и приложения. На старых девайсах китайских братьев в настройках даже был одно время переключатель для получения привилегий суперпользователя в один клик/тап. На личном устройстве, используемом каждый день, root может в неумелых руках заметно ослабить безопасность системы и стать причиной заражения вредоносным ПО, поэтому мы не советуем это делать. Но в рамках лабораторных исследований это нужная вещь, позволяющая ускорить и улучшить ход исследования.

Ситуация с iOS

При проведении аудита безопасности iOS-приложений появляются некоторые дополнительные моменты:

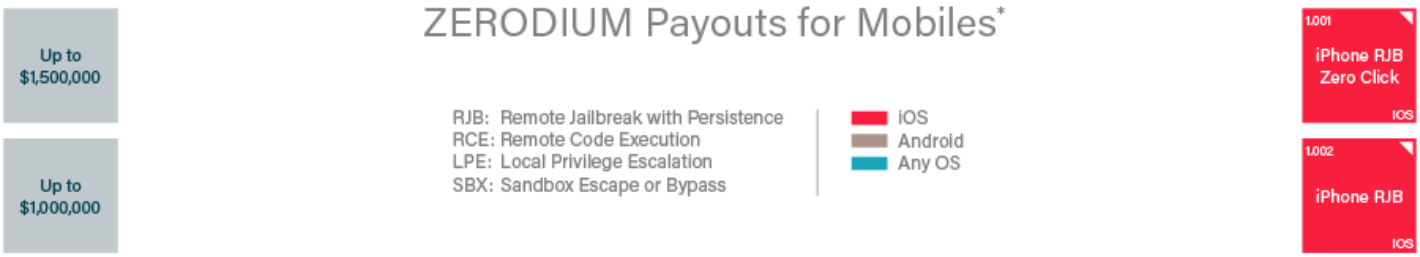
- Стоимость устройства — не самая низкая. Но отдельный девайс обязательно нужен — не использовать же личный аппарат
- Версия устройства — старые телефоны не подходят. Зачастую необходимы последние функции, включая Force Touch/Touch ID/..
- Тип устройства — телефон, планшет, часы – это все играет важную роль при проведении анализа (нам встречались случаи,

когда уязвимость в одном и том же приложении присутствовала лишь в версии под определенный тип устройства)
- Актуальность версии iOS — API iOS постоянно эволюционирует, и для качественного анализа, сам анализ должен проходить на последней версии iOS
- Наличие JailBreak на устройстве — чтобы устанавливать свои приложения и инструменты с учетом всего выше сказанного

Jailbreak

Ранее, когда только вышел первый iPhone, он был взломан за несколько дней после релиза. Довольно низкий уровень безопасности сохранялся вплоть до четвертой версии операционной системы: исследователи создавали расширения (tweak), делали свои сборки iOS и дописывали приложения. Со временем, Apple изменила своё отношение к безопасности и существенно усилила защиту системы, также добавив некоторые из самых востребованных функций, созданных для Jailbreak-устройств в своих обновлениях. Например, Control Center вначале был tweak'ом, а позже стал встроенным компонентом iOS. Каждый новый релиз системы увеличивал время на нахождение и эксплуатацию уязвимостей, по ходу меняя API и доступность функции (перемещая некоторые в Private, а некоторые в зону Public). Эти и многие другие изменения усложняли жизнь разработчикам tweak'ов (дополнений для системы) и приложений, заставляя обновлять свои детища под новые API и функции, как и исследователям безопасности. На усиление защиты устройств также повлияло пристальное внимание к платформе «плохих парней», спецслужб и всех тех, кто хотел бы получить доступ к

вашим данным на устройстве. Так, на сегодняшний день цена на удаленный JailBreak для iOS достигает 1,5млн долларов.



Время шло, и многие, кто стоял у истоков создания JailBreak, были наняты ИБ-компаниями, самой Apple или стали продавать уязвимости тем или иным лицам, не публикуя и создавая публичные JailBreak. Все это (увеличение уровня безопасности платформы и отток уязвимостей на рынок) привело к тому, что публичные JailBreak'и стали появляться все реже и реже.

На текущий момент, последней актуальной версией iOS, подходящей для использования Jailbreak, является iOS 10.2. Все остальные версии iOS10.2.1–11 пока не имеют подтверждения, а многочисленные твиты не содержат ничего, кроме видео или фото/скриншотов. Выходит, что для проведения аудита безопасности на устройстве необходимо выполнение множества факторов, и камнем преткновения становится наличие актуального JailBreak.

> **Небольшое отступление**
> На конференции Tencent Security Summit 2017 исследователь Chris Wade представил проект по полной виртуализации iPhone 6. Он не является публичным и пока неизвестно, как и когда появится для широкой аудитории. Но определенно, это очень большой шаг вперед для многих областей, связанных с iOS.
> раз два три

Анализ безопасности iOS-приложений

Теперь постепенно постепенно перейдем к основной теме статьи. Итак, существует, по большому счету, три типа анализа приложений/систем:

- WhiteBox — в рамках такого вида работ заказчик предоставляет всю информацию о приложении: исходный код, документацию и т.д. И тут мы вольны сами делать с приложением, что угодно — модифицировать, собирать, запускать, анализировать и т.д. на любом доступном устройстве даже без JailBreak.
- GrayBox — в такой ситуации заказчик не предоставляет исходного кода своего приложения по тем или иным причинам, но может сделать специальную сборку своего приложения для анализа. Например, со всеми включенными отладочными сообщениями и функциями, отключенным SSL pinning и с нашими собственными библиотеками внутри, упрощающими анализ приложения (о них подробнее далее). Такая сборка без проблем запускается и на устройстве без JailBreak.
- BlackBox — модель, полностью отражающая ситуацию с реальным злоумышленником, у которого нет никаких дополнительных сведений о приложении и возможностей влиять на его сборку. Тут просто идем в магазин распространения приложений, скачиваем одно из них и дальше пытаемся с ним что-то сделать. Здесь возможно и желание заказчика поработать в такой модели нарушителя, и участие в BugBounty-программе какого-то разработчика. Тут-то и

возникает проблема при динамическом исследовании приложений, поскольку без JailBreak, по идее, не обойтись...

Динамическое исследование безопасности iOS-приложений без Jailbreak

Проведение такого рода исследований требует некоторый подготовки исследователя и настройки окружения, поэтому далее будут описаны шаги в виде некоего мануала, необходимые для реализации задуманного. Важно отметить, что некоторые ступени могут быть пропущены, например, в случае, если использовать Xcode-проект (подписание, доставка на устройство и т.д.).

Шаг 0. Подготовка

Для того, чтобы начать исследование, необходимо соблюсти некоторые простые условия касаемо окружения:

- macOS с Xcode
- Аккаунт разработчика Apple
- iOS аппарат без JailBreak
- Расшифрованный .ipa файл исследуемого приложения
- Фреймворк, который необходимо добавить к приложению

Шаг 1. Извлечение .ipa файла

Для того, чтобы достать необходимый для анализа IPA файл, существует несколько путей:

- Из iTunes (iTunes<12.7.X+)

Покупка приложения в AppStore из iTunes.app позволяет получить привязанный .ipa файл к AppleID покупателя, это ограничивает возможность модификации, но позволяет провести статический анализ бинарного файла. Ограничение версии iTunes.app связано с последними обновлениями приложения: Apple удалила раздел AppStore.

- iFunBox, даже TestFlight(iOS≤8.3)

Приложение для управления файловой системой iOS-девайсов. Полный функционал доступен только на устройствах с iOS версией не выше iOS 8.3.

- Скачать старую версию .ipa файла из iTunes (iTunes<12.7.X+)

Это возможно благодаря использованию любого приложения, позволяющего перенаправить трафик через себя (Charles Proxy, Burp, ..). После чего необходимо запустить iTunes и скачать выбраное приложение. Далее, перехватив запрос, изменить в XML-файле пакета номер сборки, необходимой для скачивания, и продолжить исполнение. Более подробно об этом можно почитать тут и посмотреть здесь.

- Онлайн, например, с ipastore.me или 4pda (тут уже сразу расшифрованные)

Сайты и форумы с приложениями, доступные для скачиваний без

привязки к AppleID и позволяющие осуществлять необходимые манипуляции по присоединению фреймворков – наилучшее решение, но необходимо быть аккуратными с банковскими приложениями:)

Шаг 2-3. Извлечение данных и расшифровка .ipa

Здесь будет небольшое отступление от правил, когда может потребоваться JailBreak, — для шага получения расшифрованного .ipa файла. Это касается, в первую очередь, тех приложений, доступ к которым отсутствует через AppStore (например “особые сборки” или TestFlight). Получение расшифрованного файла необходимо для сборки измененного бинаря без привязки к AppleID владельца. Для этого можно воспользоваться приложениями для JailBreak девайсов (например, попросить выгрузить друга:)

- GitHub: /stefanesser/dumpdecrypted
- GitHub: /KJCracks/Clutch
- GitHub: /easonoutlook/Rasticrac

Или, если это публично доступное приложение, загрузить с таких ресурсов, как:

- iphonecake.com
- 4pda.ru

Шаг 4. Добавление фреймворка

Один из самых удобных способов добавить фреймворк к .ipa файлу — использовать Xcode проект. Существует множество проектов на GitHub, но хотелось выделить пару, на наш взгляд, самых удачных, работоспособных и интуитивно понятных.

- GitHub: /jamie72/IPAPatch (идеально для reveal / cycript)
- GitHub: /vtky/resign (идеально для frida / etc..)

В первом случае есть демка, в которой необходимо только заменить интересующий .ipa файл и собрать приложение, во втором случае — перетащить .ipa файл и Фреймворк, который необходимо присоединить. Удобство первого проекта заключается в том, что помимо внедрения библиотеки исследователь может добавить свой код, который отработает после запуска.

Что же можно положить внутрь .ipa файла?

Ответ: Что душе угодно! Но с точки зрения анализа безопасности приложения мы можем порекомендовать использовать следующие готовые фреймворки:

Frida (GitHub: /frida/frida)

Один из немногих Фреймворков, активно развивающийся сегодня и позволяющий проводить внедрение JS кода внутрь процесса, отслеживать запуск приложения и патчить его еще до процесса окончания загрузки. Преимуществами его являются легкая расширяемость под задачи, возможность заскриптовки действий и удобный клиент. Добавляя только Frida gadget к проекту, даже

ничего не выполняя, уже можно выяснить, какие вызовы происходят внутри программы и позже это применить в статическом анализе (r2+frida).

Полезные ссылки по этому проекту:

- ZeroNights'15 workshop
- Frida Objection
- Awesome Frida (examples)
- и еще один и второй

Cycript (www.cycript.org / GitHub: /nowsecure/frida-cycript)

Похожий по функционалу на Frida фреймворк, позволяет внедряться внутрь процессов и манипулировать переменными окружения и памятью через интерактивную консоль. Поддерживает не только Javascript, но и Objective-C.

Полезные ссылки по этому проекту:

- Manual
- Cycript @ 360 iDev 2013

CydiaSubstrate (cydiasubstrate.com / mobilesubstrate_iphoneos-arm.deb)

Легендарный фреймворк от самого Saurik, позволяющий модифицировать приложение без исходного кода, манипулировать API и всячески вертеть и крутить приложением, не имея

исходников. Но на текущий момент он довольно давно не обновлялся, и использовать его стоит исключительно на свой страх и риск. Помимо этого, развитие iOS вносит правки в API, поэтому в некоторых версиях iOS он и вовсе может быть бесполезен.

Reveal (revealapp.com)

Самый нестандартный из этого набора фреймворков. Подходит больше для UI/UX-сферы и изучения устройства интерфейсов на наличие скрытых полей, привязанных объектов и так далее. Плюсами является поддержка TV и Watch.

Шаг 5. Подпись приложения

Приложение не может быть установлено на смартфон без соответствующей подписи разработчика (не забываем, что устройство у нас без JailBreak). Если мы не используем Xcode, который сам, автоматически, подхватывает сертификаты, то можно сделать это вручную с помощью одного из инструментов:

- GitHub: /nowsecure/node-applesign
- GitHub: /DanTheMan827/ios-app-signer

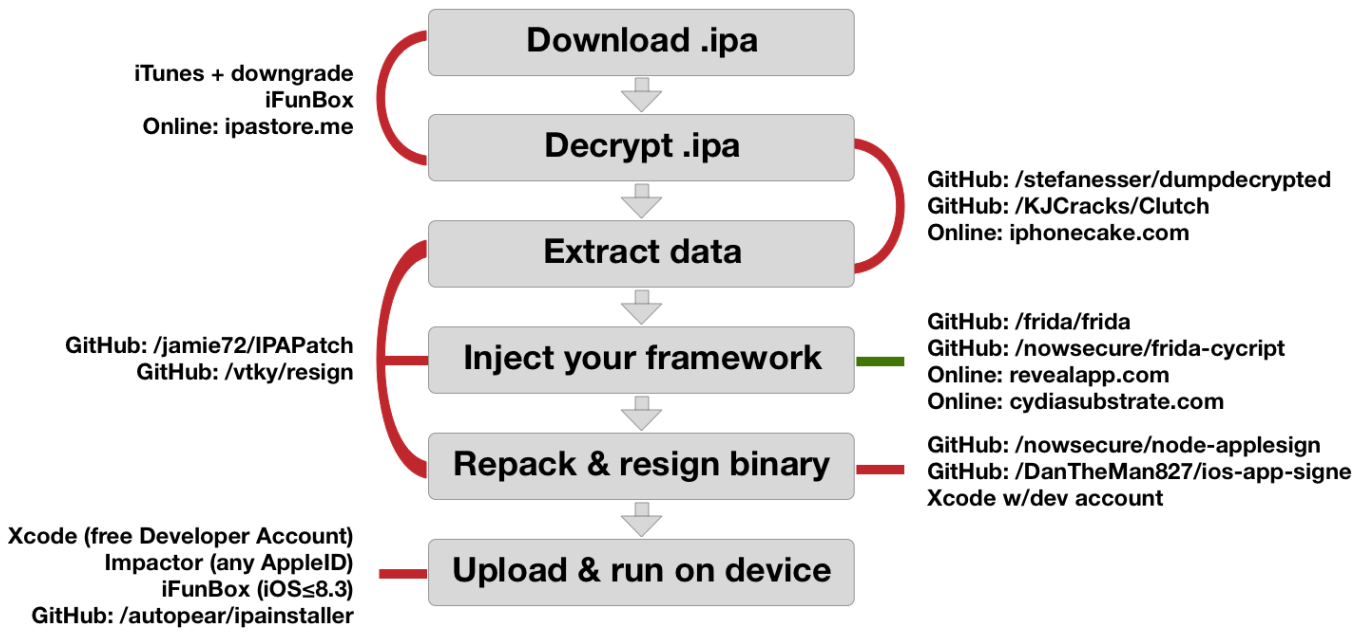
Они оба отлично справляются со своими задачами и на выходе мы получаем приложение, корректно переподписанное нашим сертификатом.

Шаг 6. Доставка на устройство

Используя проекты Xcode и устройство после выбора цели, приложение автоматически будет доставлено и запущено на смартфоне. Если на выходе есть только .ipa файл, то установить его помогут следующие утилиты:

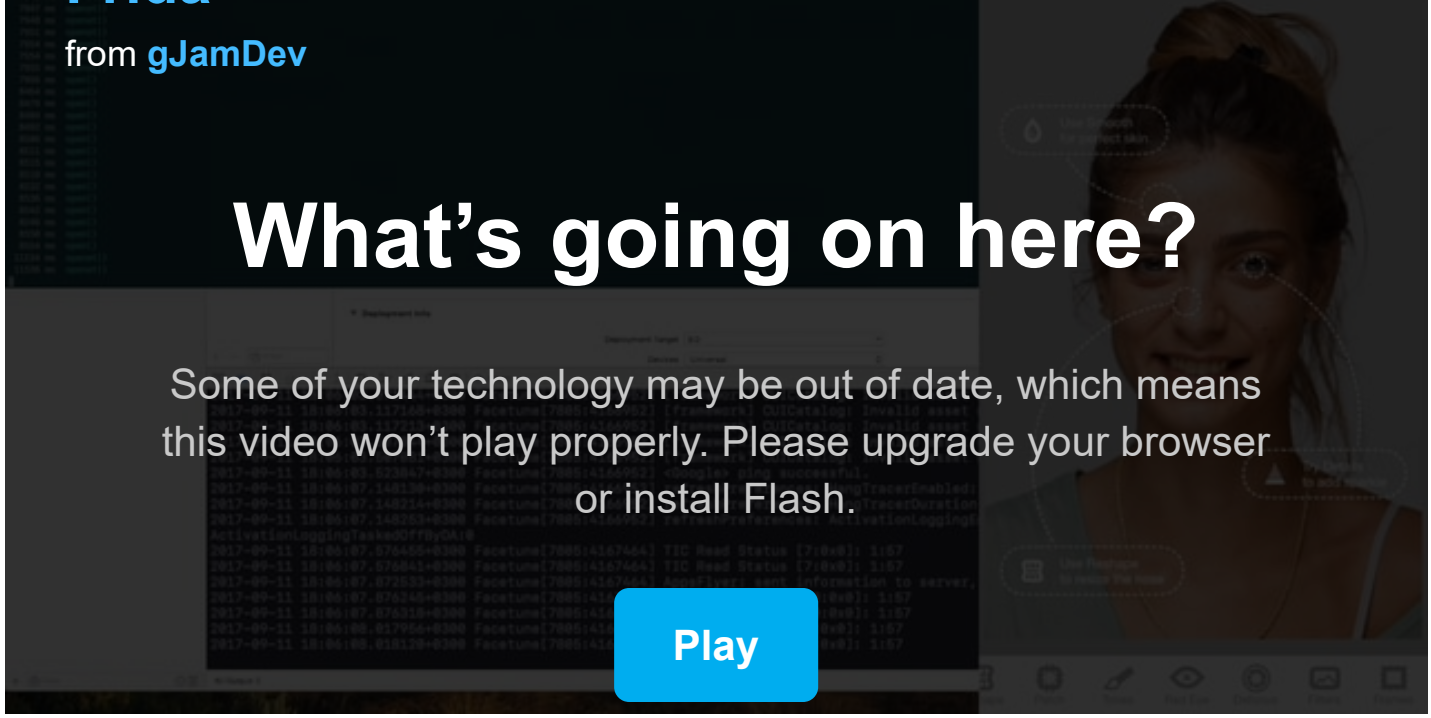
- Impactor (any AppleID)
- iFunBox (iOS≤8.3)
- JB GitHub: /autopear/ipainstaller

Если очень кратко, то схема с описанием каждого шага и утилитами, помогающими сделать каждый шаг, выглядит следующим образом:



И напоследок пару демок того, как это все выглядит в связке:

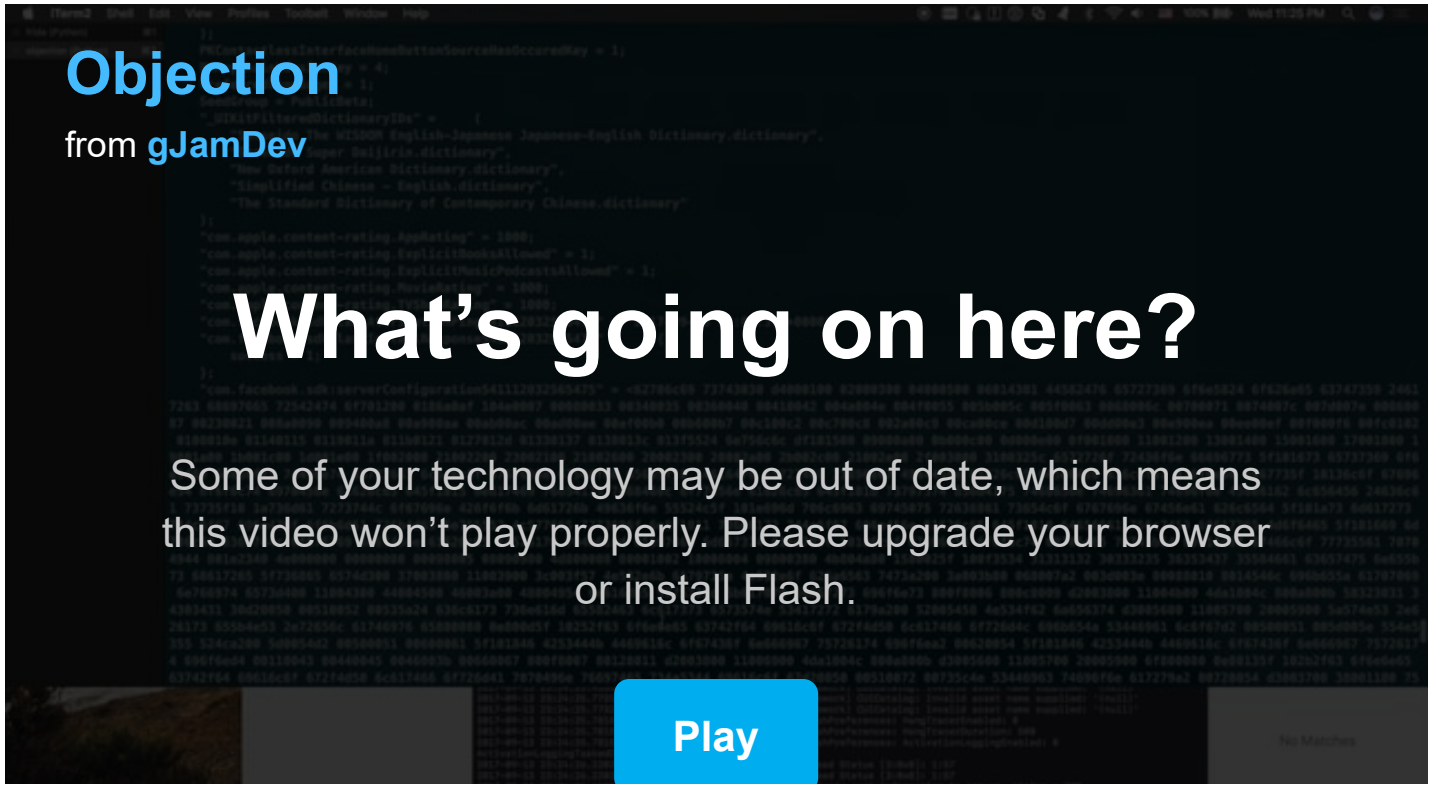
Demo 1



Описание:

Собирается проект из «раздекрипченого» .ipa FaceTune вместе с FridaGadget. Запуск его и Frida, отслеживание вызовов.

Demo 2



Описание:
Сборка проекта из Demo 1, запуск вместе с Frida дополнения Objection и просмотр содержимого папки приложения, вызов Alert'ов, чтение содержимого Keychain.

Данная работа чуть ранее была представлена на VolgaCTF'17, и слайды с доклада можно посмотреть тут.

За помощь в подготовке спасибо d1g1

UPD:

> appdb.store — позволяет подписывать приложения онлайн и бесплатно, если прикрепить свой аккаунт разрабтчика. Также оттуда можно скачать расшифрованные бинарники.

Спасибо ZonD80

Похожие публикации

XSS'им iOS устройства на примере софта от Facebook, Google, ВКонтакте

3

BeLove • 20 ноября 2013 в 00:33

Удаленный DOS exploit (device reboot) iOS ~6.1 — 7.0.3 [0day]

27

BeLove • 11 ноября 2013 в 03:37

Возможность обойти экран блокировки iOS 7.0 и получить доступ к фото. И к контактам!

28

BeLove • 20 сентября 2013 в 01:03

Популярное за сутки

Яндекс открывает Алису для всех разработчиков. Платформа Яндекс.Диалоги (бета)

69

BarakAdama • вчера в 10:52

Почему следует игнорировать истории основателей успешных стартапов

20

ПЕРЕВОД

m1rko • вчера в 10:44

Как получить телефон (почти) любой красотки в Москве, или интересная особенность MT_FREE

24

ИЗ ПЕСОЧНИЦЫ

cab404 • вчера в 20:27

Java и Project Reactor

10

zealot_and_frenzy • вчера в 10:56

Пользовательские агрегатные и оконные функции в PostgreSQL и Oracle

6

erogov • вчера в 12:46

Лучшее на Geektimes

Как фермеры Дикого Запада организовали телефонную сеть на колючей проволоке

31

NAGru • вчера в 10:10

Энтузиаст сделал новую материнскую плату для ThinkPad X200s

49

alizar • вчера в 15:32

Кто-то посылает секс-игрушки с Amazon незнакомцам. Amazon не знает, как их остановить

85

Pochtoycom • вчера в 13:06

Илон Маск продолжает убеждать в необходимости создания колонии людей на Марсе

140

marks • вчера в 14:19

Дела шпионские (часть 1)

16

TashaFridrih • вчера в 13:16