Мурад Татаев

iOS Разработчик в
Яндекс.Маркет

Telegram: @spartanrasul
Email: spartanrasul@yandex.ru

Взлом iOS приложений

Как взломать приложение и как его защитить

Что вы узнаете

- Как взламывать приложения, с какими сложностями сталкивается хакер и как сделать жизнь хакера еще хуже
- Какие из способов защиты всё еще актуальны
- Как расшифровать приложение, подписать его и установить на устройство

Чего не будет

- Защита сетевого слоя

План

Теоретическая

- Разберёмся с некоторыми понятиями
- Зачем всё это нужно?
- Инструментарий
- Варианты защиты от взлома

Практическая

- Как получить расшифрованное приложение
- Патчинг приложения на примере обхода проверки на Jailbreak
- Подпись и установку приложения на устройство
- Подмена реализации метода на примере Swift и Objective-C

Взлом

Изменение поведения приложения в интересах хакера

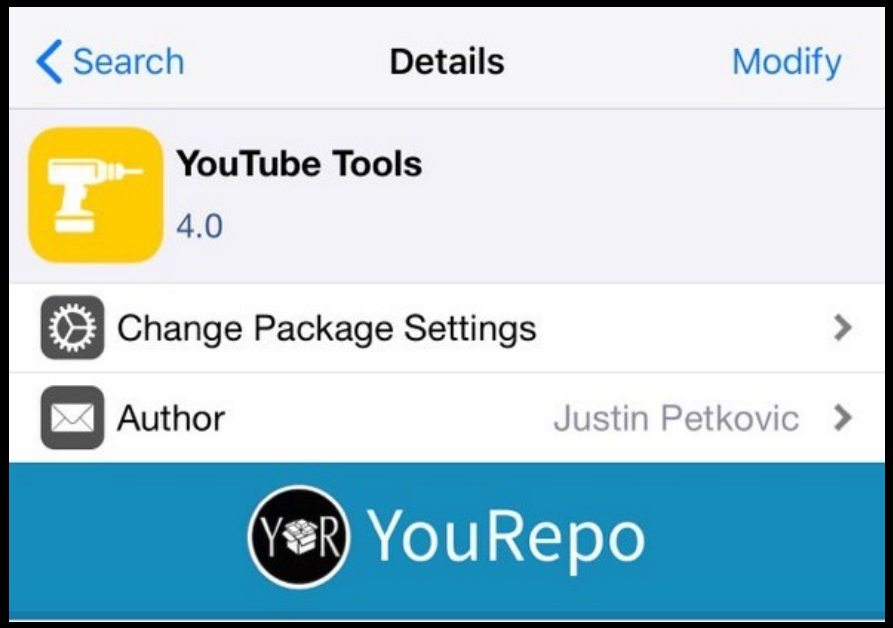
Относительная безопасность

Когда затраты на атаку, превышают полученную в результате выгоду

Почему важно защищать приложение?

Youtube Tools

Твик для отключения рекламы в приложении Youtube



https://jpet26.yourepo.com/

Plague Inc

Взломанные покупки

Minecraft

Обход платного скачивания

Как защищаться?

Варианты защиты приложения

- Проверка на Jailbreak
- Проверка на дебаггер
- Обфускация кода (security through obscurity)
- Strip Swift Symbols
- Валидация покупок на сервере
- Silent Push и Background Updates
- Писать на Swift и стараться использовать по минимуму @objc и dynamic

Совет: Иногда проверять интернет на предмет вашего взломанного приложения

Проверка на Jailbreak

Плюсы и минусы

Плюсы

- Легко использовать

Минусы

- Легко пропатчить
- Страдают мирные пользователи

https://github.com/TheSwiftyCoder/JailBreak-Detection/blob/master/JailBreak.swift

https://github.com/thii/DTTJailbreakDetection/blob/master/Sources/DTTJailbreakDetection/DTTJailbreakDetection.m

```swift
static func isJailbroken() -> Bool {

    guard let cydiaUrlScheme = NSURL(string: "cydia://package/com.example.package") else { return false }
    if UIApplication.shared.canOpenURL(cydiaUrlScheme as URL) {
        return true
    }

    #if arch(i386) || arch(x86_64)
        // This is a Simulator not an idevice
        return false
    #endif

    let fileManager = FileManager.default
    if fileManager.fileExists(atPath: "/Applications/Cydia.app") ||
        fileManager.fileExists(atPath: "/Library/MobileSubstrate/MobileSubstrate.dylib") ||
        fileManager.fileExists(atPath: "/bin/bash") ||
        fileManager.fileExists(atPath: "/usr/sbin/sshd") ||
        fileManager.fileExists(atPath: "/etc/apt") ||
        fileManager.fileExists(atPath: "/usr/bin/ssh") ||
        fileManager.fileExists(atPath: "/private/var/lib/apt") {
        return true
    }

    if canOpen(path: "/Applications/Cydia.app") ||
        canOpen(path: "/Library/MobileSubstrate/MobileSubstrate.dylib") ||
        canOpen(path: "/bin/bash") ||
        canOpen(path: "/usr/sbin/sshd") ||
        canOpen(path: "/etc/apt") ||
        canOpen(path: "/usr/bin/ssh") {
        return true
    }

    let path = "/private/" + NSUUID().uuidString
    do {
        try "anyString".write(toFile: path, atomically: true, encoding: String.Encoding.utf8)
        try fileManager.removeItem(atPath: path)
        return true
    } catch {
        return false
    }
}

static func canOpen(path: String) -> Bool {
    let file = fopen(path, "r")
    guard file != nil else { return false }
    fclose(file)
    return true
}
```

Проверка на дебаггер

Плюсы и минусы

Плюсы

- Легко использовать
- Сложнее патчить
- Хорошо работает с **отложенными ошибками**

Минусы

- Это тоже можно пропатчить

Проверка на дебаггер

```
let isDebuggerAttached: Bool = {
    var kernelFlags: [Int32] = [CTL_KERN, KERN_PROC, KERN_PROC_PID, getpid()]
    var currentProcessInfo: kinfo_proc = kinfo_proc()
    var currentProcessMemorySize = MemoryLayout<kinfo_proc>.size //to read right amount of data

    let didUpdateProcessInfo = kernelFlags.withUnsafeMutableBytes { (pointer: UnsafeMutableRawBufferPointer) -> Bool in
        guard let bindedKernelFlags = pointer.bindMemory(to: Int32.self).baseAddress else { return false }
        let didUpdateKernelStateInfo = sysctl(bindedKernelFlags,
                                              4,
                                              &currentProcessInfo,
                                              &currentProcessMemorySize,
                                              nil,
                                              0)
        return -1 != didUpdateKernelStateInfo
    }

    if didUpdateProcessInfo && (currentProcessInfo.kp_proc.p_flag & P_TRACED) != 0 {
        return true
    }

    return false
}()
```

Обфускация кода
Плюсы и минусы

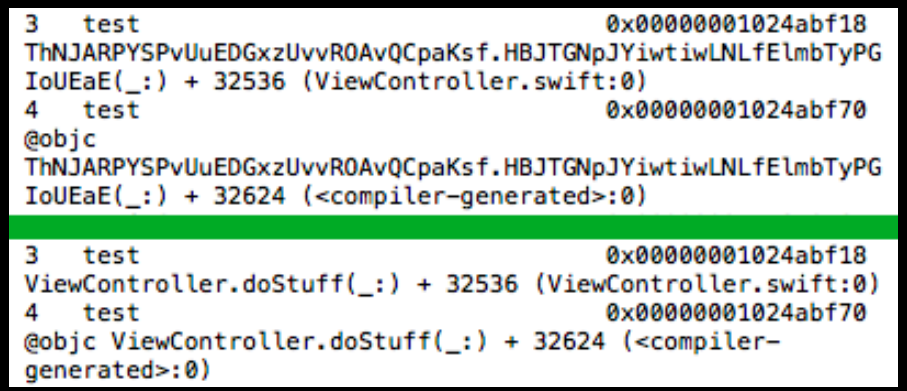
Плюсы

- Код превращается в лабиринт

Минусы

- Иногда и вы в нём оказываетесь
- Крэш логи приходится каждый раз расшифровывать
- Нет готового решения, годного для Enterprise приложений

https://github.com/rockbruno/swiftshield



Strip Swift Symbols

Плюсы и минусы

Плюсы

- Легко использовать
- Удаление части информации о Swift классах
- Включено по умолчанию

Минусы

- Информацию о классе можно найти и другим способом

Что будет, если оставить Swift Symbols?

```swift
import UIKit

class ViewController: UIViewController {

    @IBOutlet var label: UILabel?

    override func viewDidLoad() {
        super.viewDidLoad()
    }

    @IBAction private func buttonAction(_ sender: UIButton) {
        //Some internal business logic
    }

    func publicFunc() {
        print("\(#function) is called!")
    }

    func buySomethingInApp() {
        print("Buying some stuff")
    }

    static func buildScreen() {
        print("Building screen")
    }
}
```

Что будет, если оставить Swift Symbols?

```
class InvestGame.ViewController : UIViewController /System/Library/Frameworks/UIKit.framework/UIKit {

        // Properties
        var label : UILabel?

        // ObjC -> Swift bridged methods
        0x100005ff0  @objc ViewController.label.getter
        0x1000060dc  @objc ViewController.label.setter
        0x100006798  @objc ViewController.viewDidLoad()
        0x10000684c  @objc ViewController.buttonAction(_:)
        0x100006e28  @objc ViewController.init(nibName:bundle:)
        0x1000070c0  @objc ViewController.init(coder:)
        0x10000717c  @objc ViewController.__ivar_destroyer

        // Swift methods
        0x10000603c  func ViewController.label.getter // getter
        0x100006140  func ViewController.label.setter // setter
        0x100006328  func ViewController.label.modify // modifyCoroutine
        0x1000067e0  func ViewController.buttonAction(_:) // method
        0x1000068b8  func ViewController.publicFunc() // method
        0x100006a00  func ViewController.buySomethingInApp() // method
}
```

А если отрезать?

```
class InvestGame.ViewController : UIViewController /System/Library/Frameworks/UIKit.framework/UIKit {

        // Properties
        var label : UILabel?

        // ObjC -> Swift bridged methods
        0x10000504c  @objc ViewController.label.getter
        0x10000507c  @objc ViewController.label.setter
        0x1000052f8  @objc ViewController.viewDidLoad()
        0x100005324  @objc ViewController.buttonAction(_:)
        0x100005328  @objc ViewController.init(nibName:bundle:)
        0x100005438  @objc ViewController.init(coder:)
        0x10000551c  @objc ViewController.__ivar_destroyer

        // Swift methods
}
```

Валидация покупок на сервере

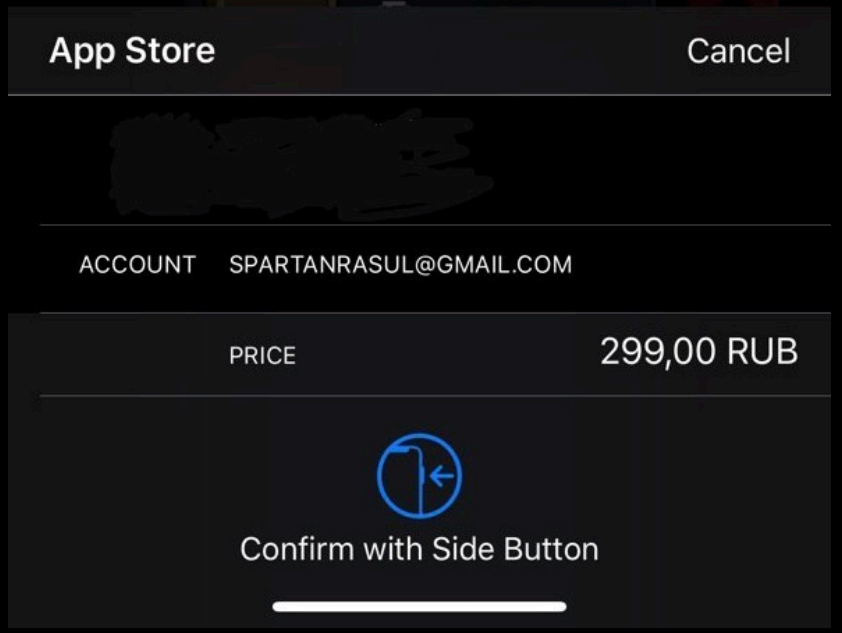
Плюсы и минусы

Плюсы

- Безопасность

Минусы

- Нужно покупать хостинг



Валидация покупок на сервере

А что если я буду валидировать на клиенте?

Плюсы

- Не нужен сервер

Минусы

- Потеря денег

LocaliAPStore - ваш худший враг

Инструменты

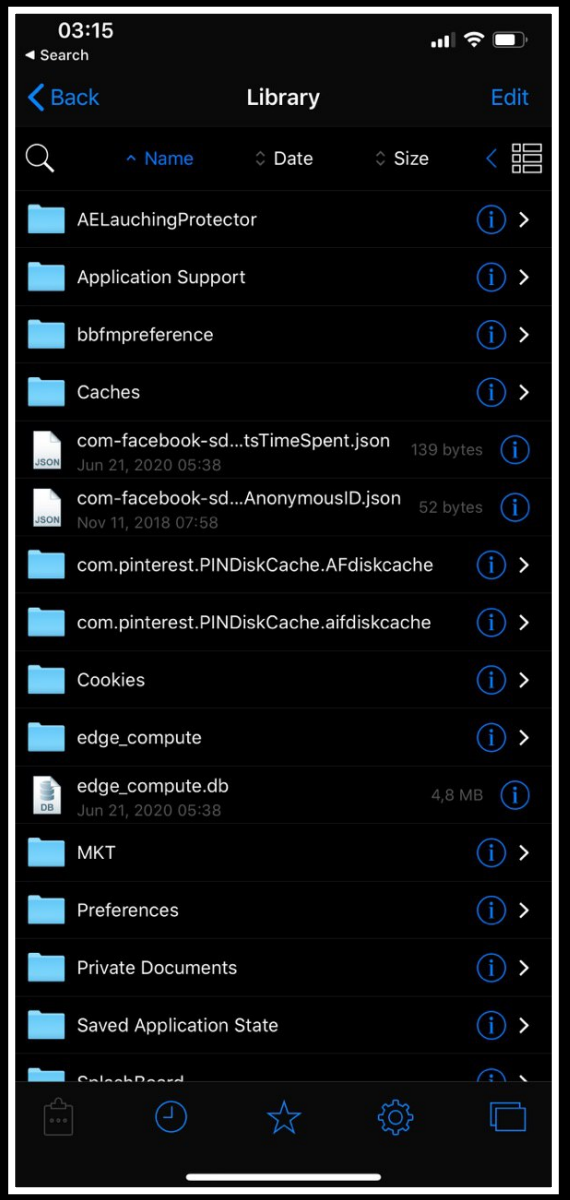
Чем я пользуюсь каждый день

- Filza - файловый менеджер
- Unc0ver - Jailbreak
- Менеджер пакетов - Cydia
- NewTerm - терминал на устройстве
- OpenSSH - доступ по SSH
- Frida - инструментарий реверс инженера
- dsdump - дамп Swift/Objective-C классов/методов
- FLEXing - инструмент для анализа
- Ghidra - дизассемблер и т.д.

А теперь конкретнее…

Filza

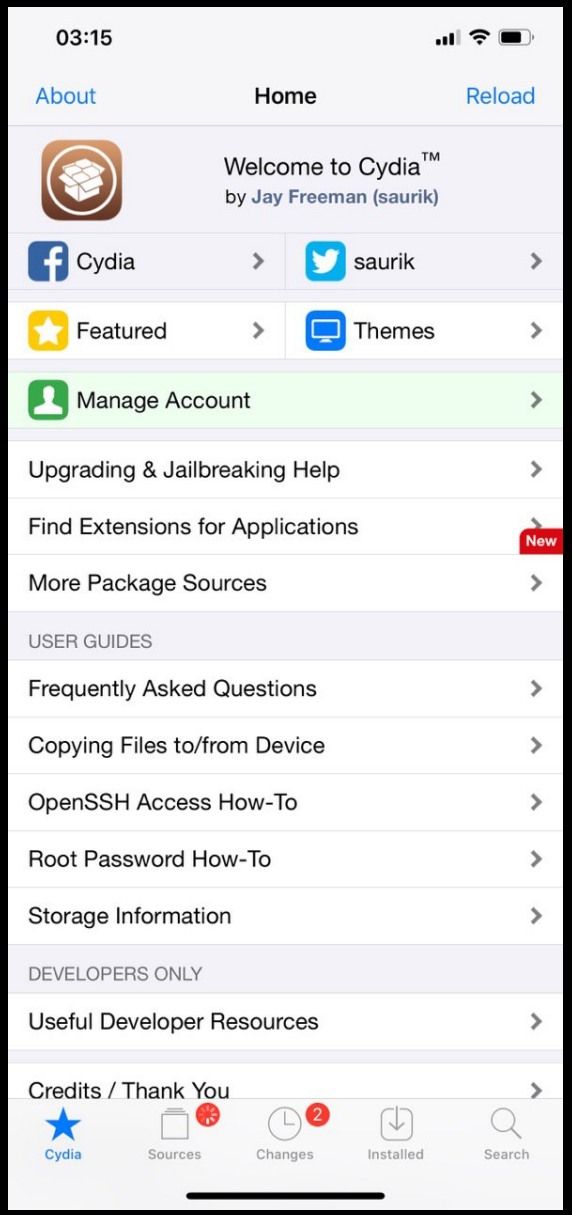
Файловый менеджер



Unc0ver
Jailbreak

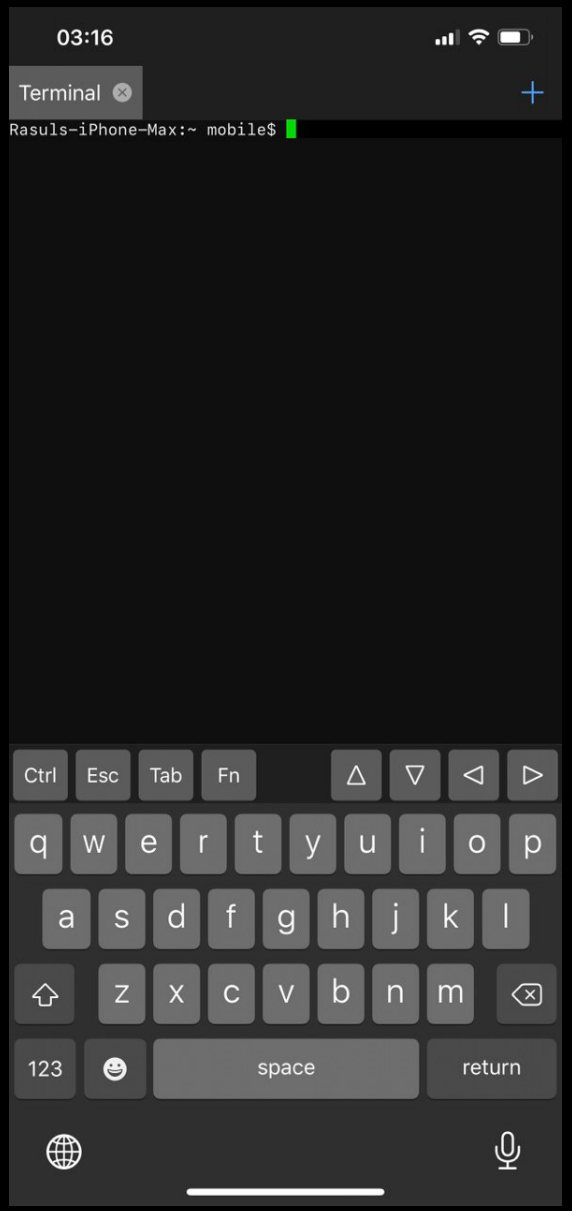
Cydia

Менеджер пакетов



NewTerm

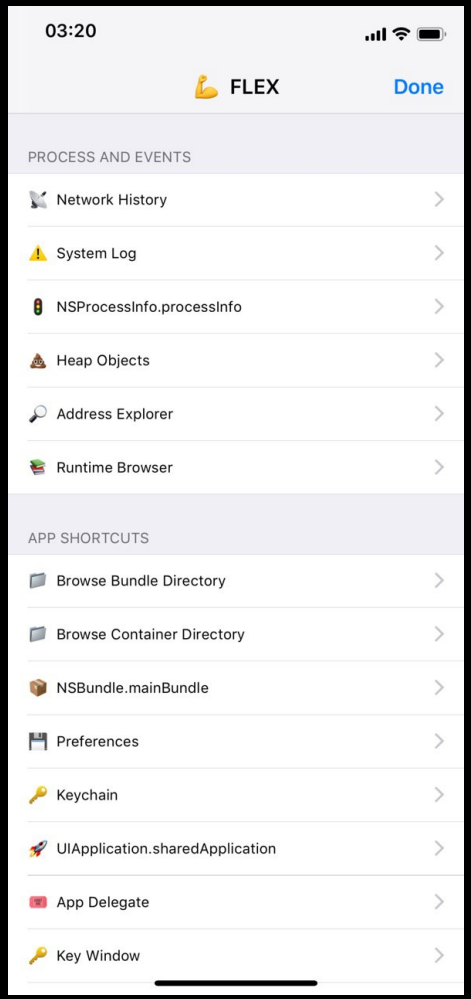
Мобильный терминал

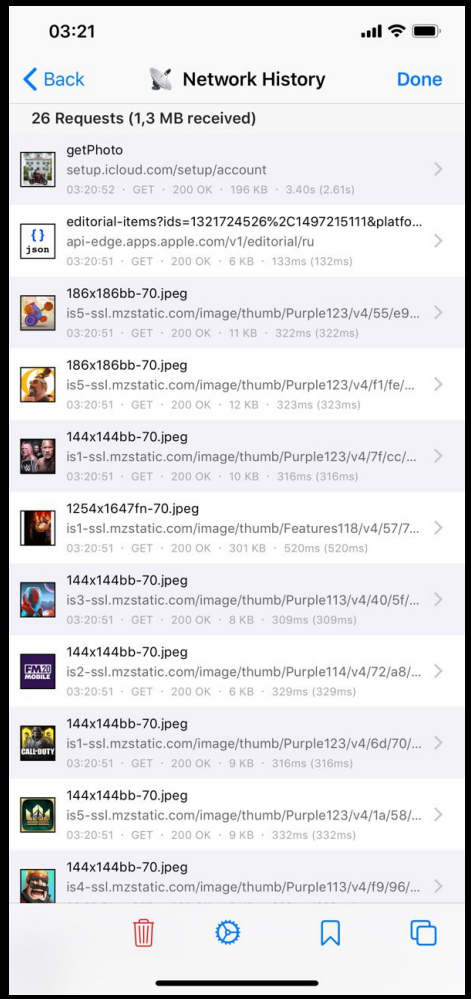


FLEXing

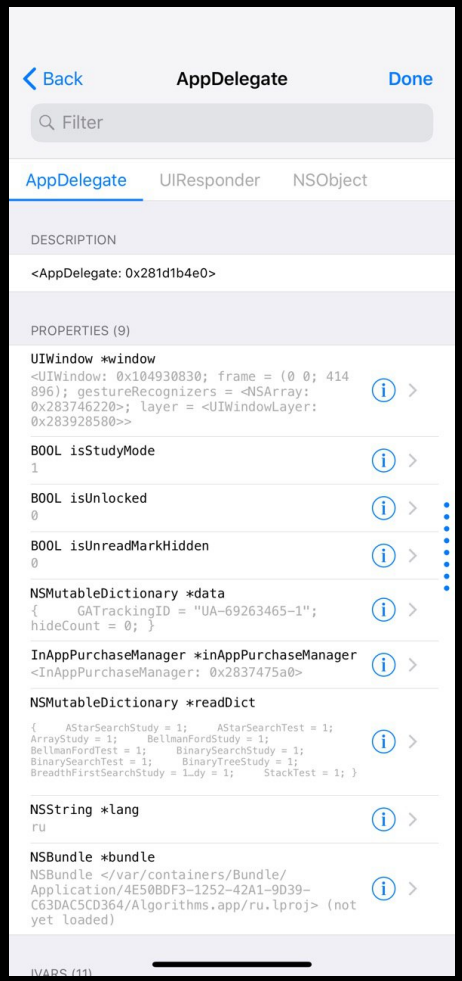
Анализ приложения на устройстве

- Интернет трафик
- Объекты в куче и вызов методов в рантайме
- Общий список классов приложения
- Просмотр Keychain (!) и UserDefaults
- И т.д.





Пример как делать не надо

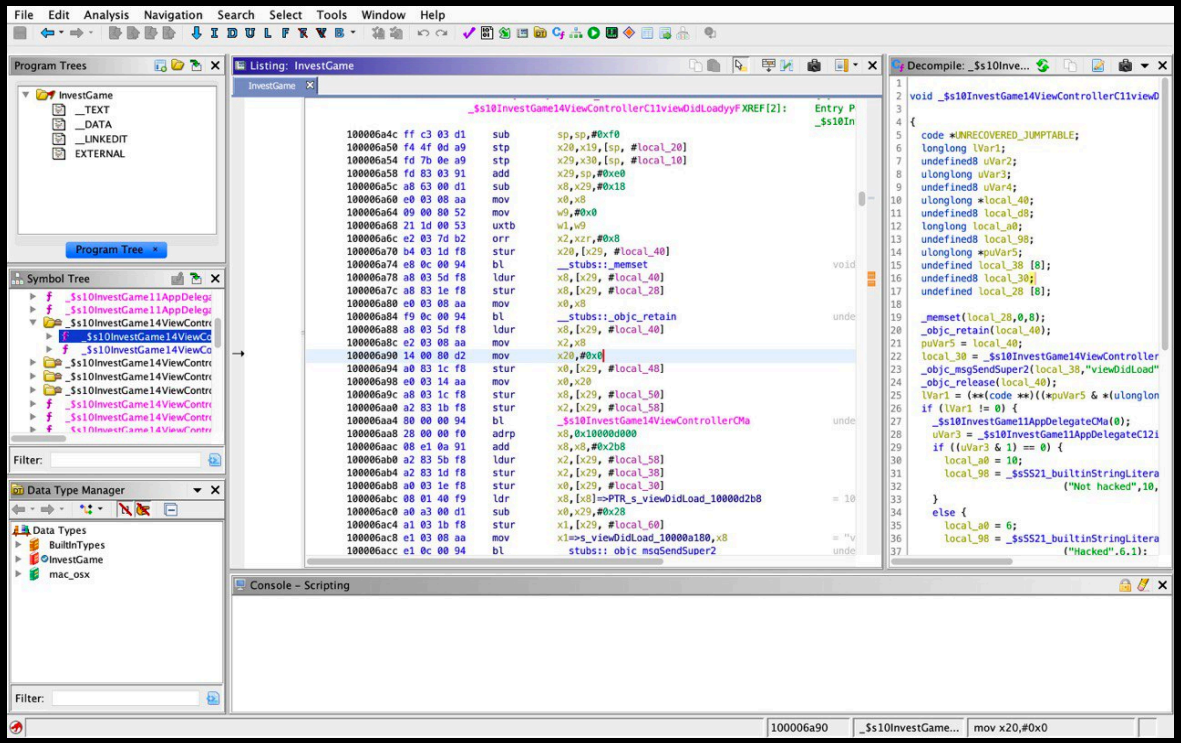


Пример как делать не надо

Ghidra

Дизассемблер от АНБ США

- Бесплатный (Open Source)
- Умеет в arm64
- Умеет в псевдокод
- Можно редактировать бинарник и позже его экспортировать



Пора переходить к практике!

https://github.com/SpartanRASUL/MobiusStep1

Как достать расшифрованное приложение?

- Клонировать репозиторий https://github.com/SpartanRASUL/MobiusStep1
- Запустить в режиме Release на взломанном устройстве
- iproxy 2222 22 (https://github.com/tcurdt/iProxy)
- frida-ps -Uai
- https://github.com/AloneMonkey/frida-ios-dump
- ./dump.py “your app name”

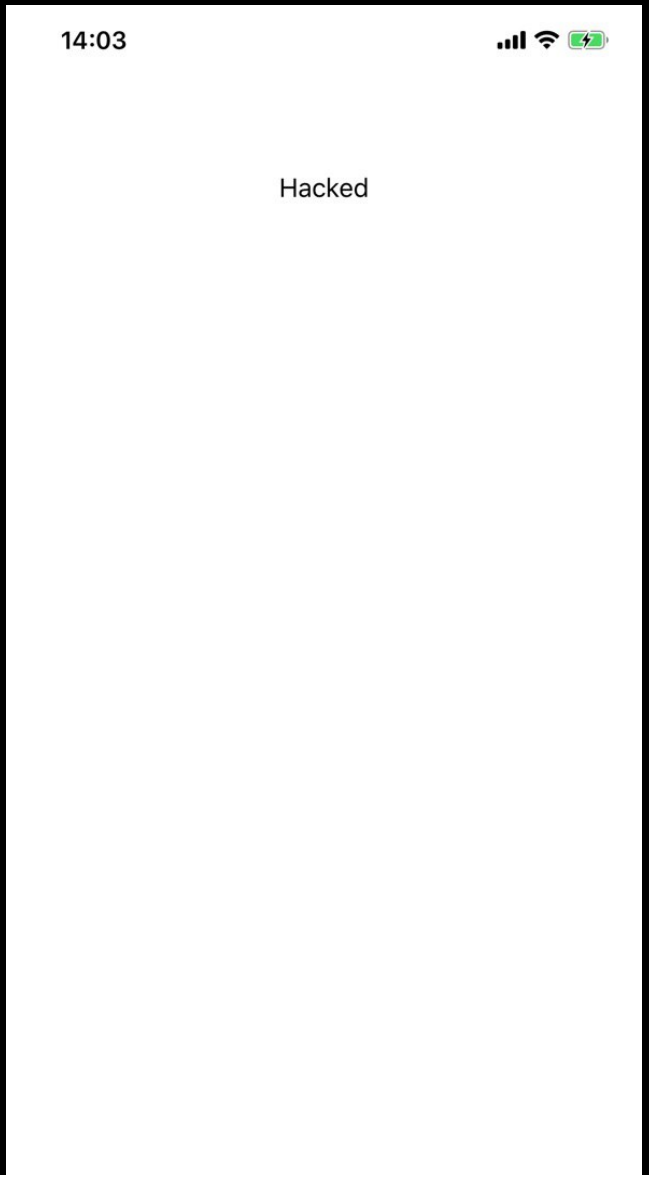
Ограничения:

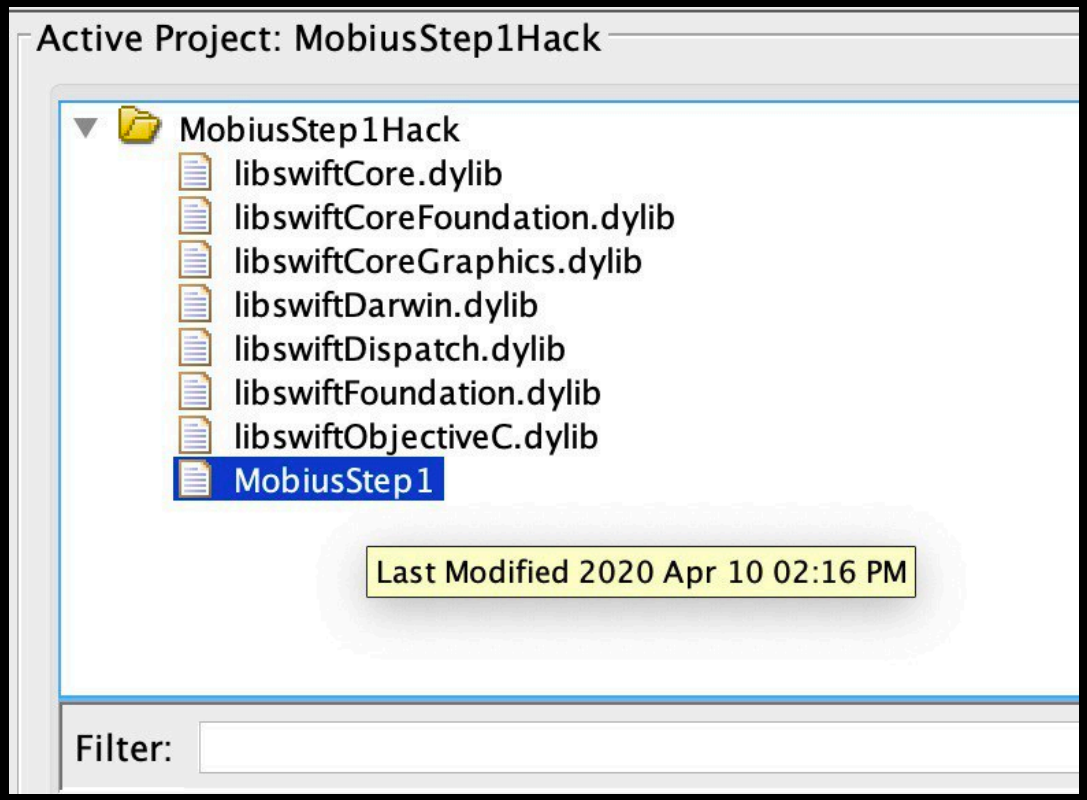
Не дампятся extension’ы

Обход проверки на Jailbreak

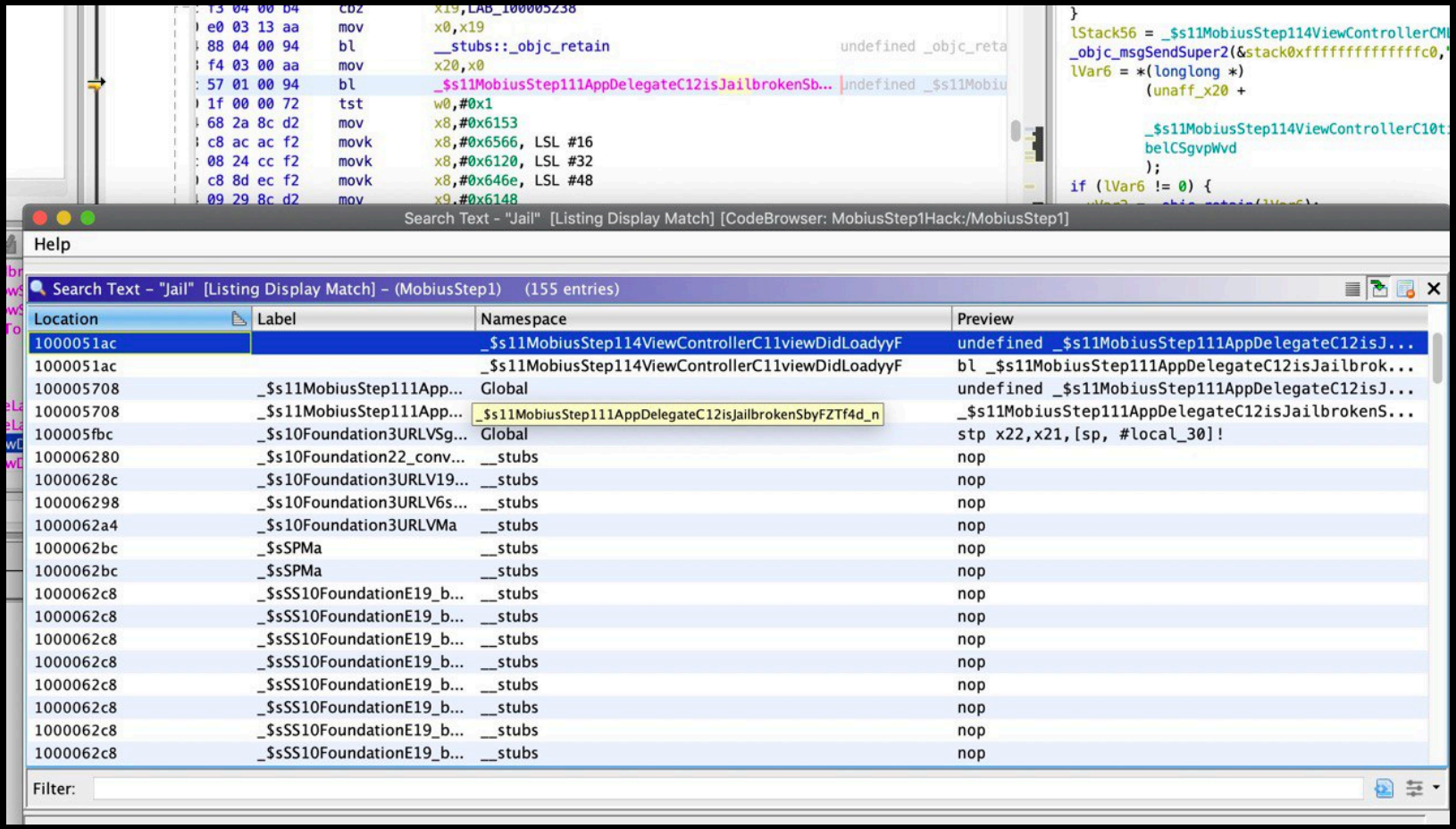
Патчим бинарник

- Достаём .ipa файл (слайды выше)
- Разархивируем и переходим в папку Payload и далее в .app
- Копируем бинарник и содержимое папки Frameworks в отдельную папку
- Запускаем Ghidra, создаем новый проект и добавляем ранее скопированные файлы




Active Project: MobiusStep1Hack
MobiusStep1Hack
libswiftCore.dylib
libswiftCoreFoundation.dylib
libswiftCoreGraphics.dylib
libswiftDarwin.dylib
libswiftDispatch.dylib
libswiftFoundation.dylib
libswiftObjectiveC.dylib
MobiusStep1
Last Modified 2020 Apr 10 02:16 PM
Filter:

Как найти необходимый участок кода?



Патчим инструкцию

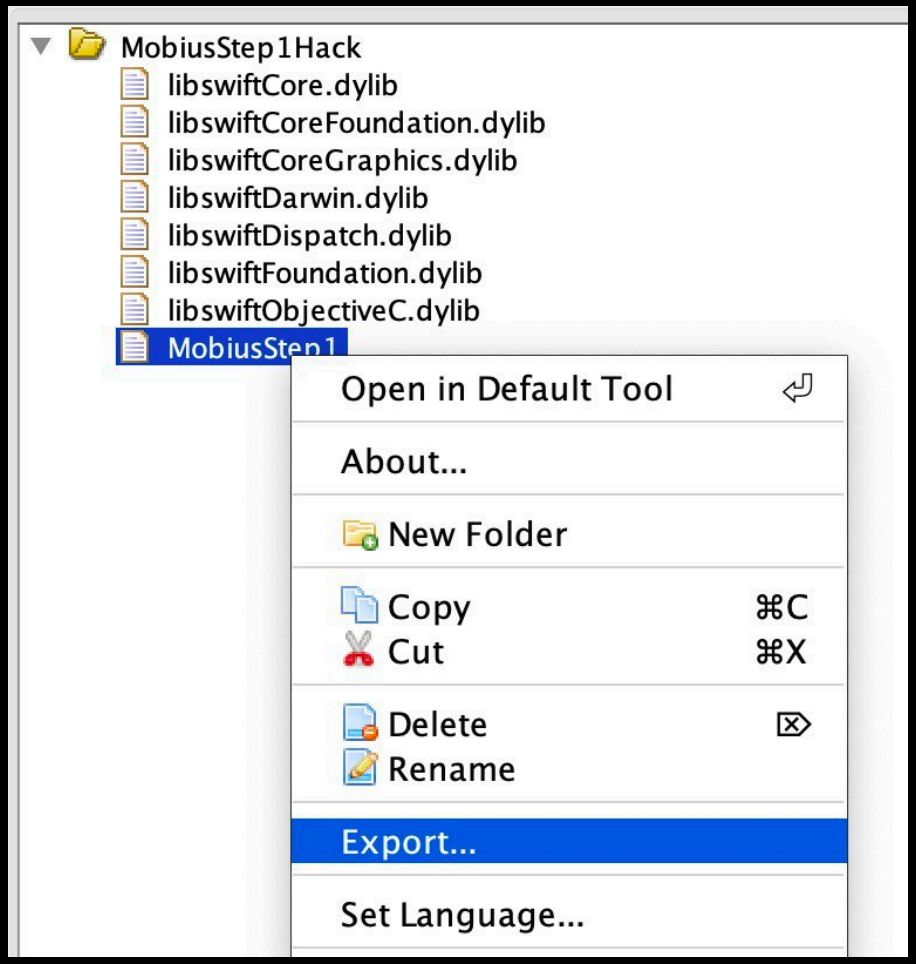
До

```
005190 1f 20 03 d5     nop
005194 e8 09 02 58     ldr        x8,_$s11MobiusStep114ViewControllerC10titleLab... = 000000000000
005198 93 6a 68 f8     ldr        x19,[x20, x8, LSL #0x0]
00519c f3 04 00 b4     cbz        x19,LAB_100005238
0051a0 e0 03 13 aa     mov        x0,x19
0051a4 88 04 00 94     bl         __stubs::_objc_retain                    undefined _obj
0051a8 f4 03 00 aa     mov        x20,x0
0051ac 57 01 00 94     bl         _$s11MobiusStep111AppDelegateC12isJailbrokenSb... undefined _$s1
0051b0 1f 00 00 72     tst        w0,#0x1
0051b4 68 2a 8c d2     mov        x8,#0x6153
0051b8 c8 ac ac f2     movk       x8,#0x6566, LSL #16
0051bc 08 24 cc f2     movk       x8,#0x6120, LSL #32
```

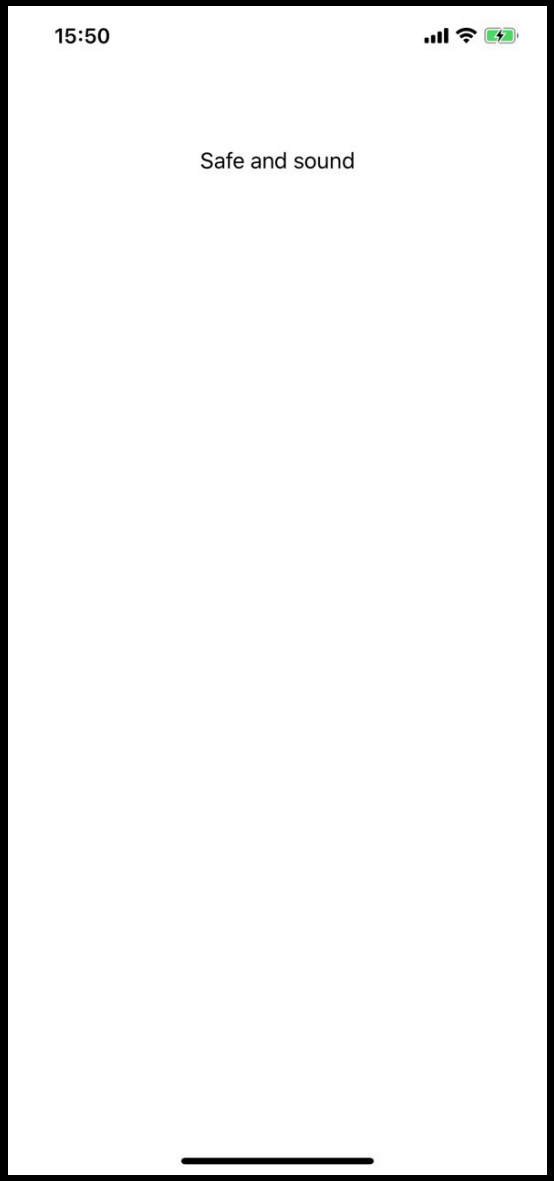
После

```
1000519c f3 04 00 b4     cbz        x19,LAB_100005238
1000051a0 e0 03 13 aa     mov        x0,x19
1000051a4 88 04 00 94     bl         __stubs::_objc_retain
1000051a8 f4 03 00 aa     mov        x20,x0
1000051ac 00 00 80 52     mov        w0,#0x0
1000051b0 1f 00 00 72     tst        w0,#0x1
1000051b4 68 2a 8c d2     mov        x8,#0x6153
1000051b8 c8 ac ac f2     movk       x8,#0x6566, LSL #16
1000051bc 08 24 cc f2     movk       x8,#0x6120, LSL #32
1000051c0 c8 8d ec f2     movk       x8,#0x646e, LSL #48
1000051c4 09 29 8c d2     mov        x9,#0x6148
1000051c8 69 6c ad f2     movk       x9,#0x6b63, LSL #16
```

Экспортируем бинарник



Собираем, подписываем, устанавливаем



Поздравляю, вы пропатчили приложение

Подпись и установка приложения

Prerequisites

- Профиль разработчика
- AdHoc provisioning profile
- libimobiledevice (brew install libimobiledevice)
- Приложение
- Терпение и усидчивость

Копируем необходимые файлы

Скопировать ваш Adhoc Provisioning Profile в папку с приложением и выполнить код:

- security cms -D -i adhoc.mobileprovision > profile.plist
- /usr/libexec/PlistBuddy -x -c 'Print :Entitlements' profile.plist > entitlements.plist

После:
скопировать Adhoc Provisioning Profile в контейнер приложения (Payload/*.app) и переименовать его в embedded.mobileprovision

Находим ключ для подписи

security find-identity -v -p codesigning

```
~ security find-identity -v -p codesigning
  1) FEE                                42D "iPhone Developer:                    (S8      G3)"
  2) 57A                                9A5 "iPhone Developer:                    (4G      PH)"
  3) 128                                39D "Apple Development: Murad Tataev (6S      93)"
  4) A23                                8D4 "iPhone Distribution:                 (SXS     A)"
  5) F7E                                ADC "iPhone Developer: Murad Tataev (6S      93)"
  6) 094                                571 "iPhone Distribution: Murad Tataev (7L      T7)"
     6 valid identities found
~
```

"iPhone Distribution: Murad Tataev (7L.....T7)"

Подписываем файлы!

- codesign -f -s "iPhone Distribution: Murad Tataev (….)” --entitlements "entitlements.plist" Payload/*.app/Frameworks/*.dylib
- codesign -f -s "iPhone Distribution: Murad Tataev (….)” --entitlements "entitlements.plist" Payload/*.app/MobiusStep1
- codesign -f -s "iPhone Distribution: Murad Tataev (….)” --entitlements "entitlements.plist" Payload/*.app

Архивируем: zip -qr resigned.ipa Payload
Устанавливаем: ideviceinstaller -i resigned.ipa

(Можно установить даже на устройство без Jailbreak)

Инъекция фреймворков

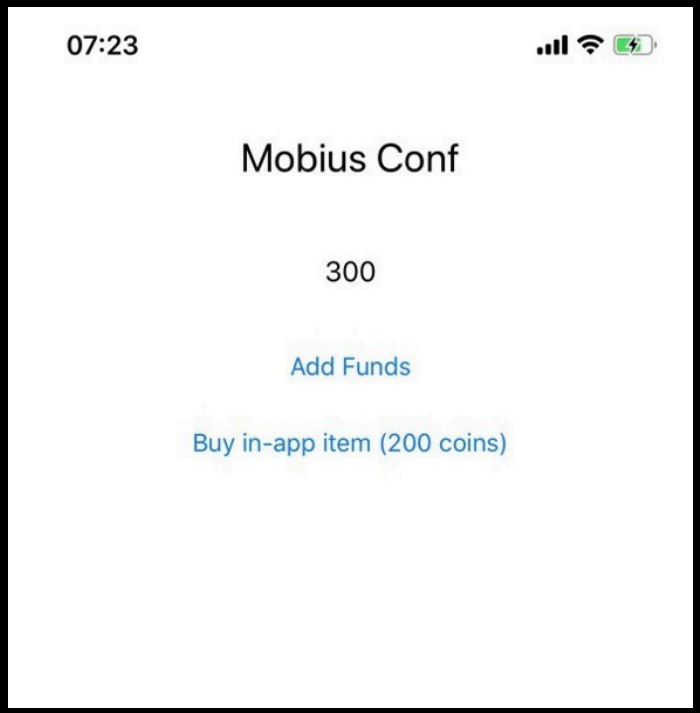
Для подмены реализации методов

Необходимые инструменты

- otool (поставляется с Xcode)
- dsdump
- optool (https://github.com/alexzielenski/optool)
- Framework

Потренируемся!

- Создайте проект и добавьте в него новый таргет. В меню, выберите фреймворк
- Во фреймворке, добавьте новый Obj-C класс. Название по вкусу.
- В имплементационный файл, добавьте код указанный на слайде ниже



https://github.com/SpartanRASUL/MobiusStep2

MobiusStep2/MOBFramework/Injector.m

```objc
#import <UIKit/UIKit.h>
#import "Injector.h"

@implementation Injector

static void __attribute__((constructor)) initialize(void){
    NSLog(@"Hello from Framework");
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(3 * NSEC_PER_SEC)), dispatch_get_main_queue(), ^{
        UIViewController *vc = UIApplication.sharedApplication.keyWindow.rootViewController; ⚠ 'keyWindow' is deprecated:
        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"Mobius Conf"
                                                                        message:@"Yaay, you have done that"
                                                                 preferredStyle:UIAlertControllerStyleAlert];

        [alert addAction:[UIAlertAction actionWithTitle:@"Close" style:UIAlertActionStyleCancel handler:nil]];
        [vc presentViewController:alert animated:true completion:nil];
    });
}

@end
```

Поиграемся с кодом

- Склонируйте https://github.com/SpartanRASUL/Substrate и скопируйте фреймворк Substrate на рабочий стол
- В фреймворк MOBFramework добавьте фреймворк Substrate (drag and drop)
- Импортируйте заголовок "CydiaSubstrate/CydiaSubstrate.h"

https://github.com/SpartanRASUL/MobiusStep2

А давайте вызовем какую нибудь функцию?

- В Xcode, в навигаторе, откройте папку Products
- Отобразите файл MobiusStep2.app в Finder'e
- Перейдите в папку MobiusStep2.app через терминал

```
~/L/D/X/D/M/B/P/D/MobiusStep2.app  dsdump -c MobiusStep2 | grep viewDidLoad
0x001000047e0 _$s11MobiusStep214ViewControllerC11viewDidLoadyyFTo
0x001000045dc _$s11MobiusStep214ViewControllerC11viewDidLoadyyF
```

```
static void __attribute__((constructor)) initialize(void){
    NSLog(@"Hello from Framework");

    void *symbol = MSFindSymbol(NULL, "_$s11MobiusStep214ViewControllerC11viewDidLoadyyF");
    ((void (*)(void)) symbol)();
}
```

Вызовем свой код, после viewDidLoad

```objc
#import <UIKit/UIKit.h>
#import <CydiaSubstrate/CydiaSubstrate.h>
#import "Injector.h"

@implementation Injector

static void (*orig_viewDidLoad)(void) = NULL;

static void __attribute__((constructor)) initialize(void){
    NSLog(@"Hello from Framework");

    void *symbol = MSFindSymbol(NULL, "_$s11MobiusStep214ViewControllerC11viewDidLoadyyF");
    MSHookFunction(symbol,
                   (void*)viewDidLoadSwizzled,
                   (void**)&orig_viewDidLoad);
}

static void viewDidLoadSwizzled() {
    orig_viewDidLoad();
    dispatch_after(dispatch_time(DISPATCH_TIME_NOW, (int64_t)(2 * NSEC_PER_SEC)), dispatch_get_main_queue(), ^{
        UIViewController *vc = UIApplication.sharedApplication.keyWindow.rootViewController; ⚠ 'keyWindow' is deprecated:
        UIAlertController *alert = [UIAlertController alertControllerWithTitle:@"Hello"
                                                                        message:@"Message from swizzled viewDidLoad"
                                                                 preferredStyle:UIAlertControllerStyleAlert];
        [alert addAction:[UIAlertAction actionWithTitle:@"Close" style:UIAlertActionStyleCancel handler:nil]];
        [vc presentViewController:alert animated:true completion:nil];
    });
}

@end
```

Хмм, а как насчёт бесконечности денег?

```objc
#import <UIKit/UIKit.h>
#import "Injector.h"
#import <CydiaSubstrate/CydiaSubstrate.h>

@implementation Injector

static void (*spendMoneyHook_orig)(int) = NULL;

static void __attribute__((constructor)) initialize(void){
    NSLog(@"Hello from Framework");
    void *spendMoneyHook = MSFindSymbol(NULL, "_$s11MobiusStep215PurchaseManagerC5spendyySiF");
    MSHookFunction(spendMoneyHook, (void*)spendMoneyHookSwizzled, (void**)&spendMoneyHook_orig);
}

static void spendMoneyHookSwizzled(int amount) {
    spendMoneyHook_orig(1);
}

@end
```

Потренировались? Круто!

Пойдем инжектить в “реальное” приложение

- Скопируйте ранее созданный фреймворк. Его можно найти в папке Products
- Соберите проект в режиме Release предварительно удалив ранее созданный фреймворк из проекта
- Скопируйте ipa файл и разархивируйте его
- Скопируйте MOBFramework.framework в папку Frameworks (Payload/MobiusStep2.app/Frameworks)

Вставляем загрузку своего фреймворка

```
~/D/h/app_mobius_2 cd Payload/MobiusStep2.app/
~/D/h/a/P/MobiusStep2.app otool -L MobiusStep2
MobiusStep2:
	/System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300.0.0, current version 1673.126.0)
	/usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
	/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1281.0.0)
	/System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, current version 61000.0.0)
	@rpath/libswiftCore.dylib (compatibility version 1.0.0, current version 1100.2.255)
	@rpath/libswiftFoundation.dylib (compatibility version 1.0.0, current version 0.0.0)
	@rpath/libswiftObjectiveC.dylib (compatibility version 1.0.0, current version 0.0.0)
~/D/h/a/P/MobiusStep2.app optool install -c load -p "@rpath/MOBFramework.framework/MOBFramework" -t MobiusStep2
Found thin header...
Inserting a LC_LOAD_DYLIB command for architecture: arm64
Successfully inserted a LC_LOAD_DYLIB command for arm64
Writing executable to MobiusStep2...
~/D/h/a/P/MobiusStep2.app otool -L MobiusStep2
MobiusStep2:
	/System/Library/Frameworks/Foundation.framework/Foundation (compatibility version 300.0.0, current version 1673.126.0)
	/usr/lib/libobjc.A.dylib (compatibility version 1.0.0, current version 228.0.0)
	/usr/lib/libSystem.B.dylib (compatibility version 1.0.0, current version 1281.0.0)
	/System/Library/Frameworks/UIKit.framework/UIKit (compatibility version 1.0.0, current version 61000.0.0)
	@rpath/libswiftCore.dylib (compatibility version 1.0.0, current version 1100.2.255)
	@rpath/libswiftFoundation.dylib (compatibility version 1.0.0, current version 0.0.0)
	@rpath/libswiftObjectiveC.dylib (compatibility version 1.0.0, current version 0.0.0)
	@rpath/MOBFramework.framework/MOBFramework (compatibility version 0.0.0, current version 0.0.0)
~/D/h/a/P/MobiusStep2.app
```

Не забудьте подписать фреймворк

codesign -f -s "identity” Payload/*.app/Frameworks/*.framework

Готово!

Но!

Хукать свифтовые методы можно только на взломанном устройстве :(

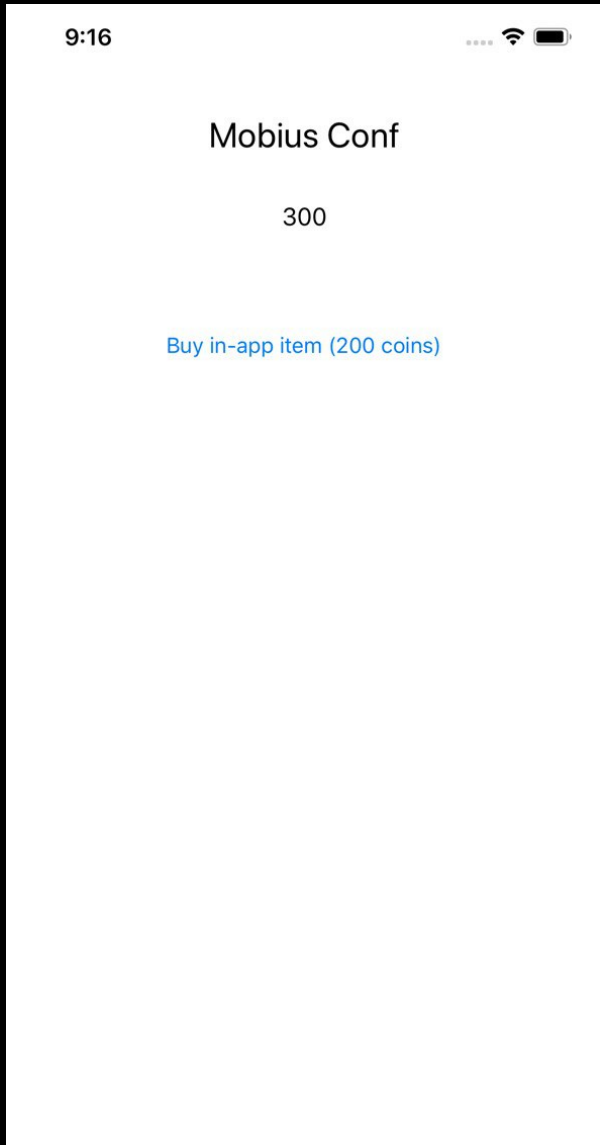
А как там дела у Objective-C?

Внешний вид приложения

Наша цель: - иметь бесконечное количество монет.

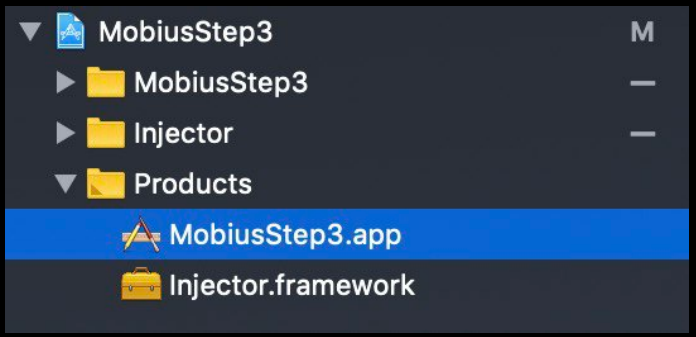
А поможет нам в этом ?... Objective-C Runtime!

https://github.com/SpartanRASUL/MobiusStep3



Подготовка проекта

- Клонируем проект https://github.com/SpartanRASUL/MobiusStep3
- Собираем (билдим) проект
- Переходим в папку с приложением и открываем бандл приложения (.app)



- Запускаем терминал в этой папке

Ищем уязвимость

```
~/L/D/X/D/M/B/P/D/MobiusStep3.app dsdump --objc MobiusStep3 -vvv
0x00100004028 ViewController : UIViewController
        // instance methods
        0x00100000eb0 -[ViewController viewDidLoad]
        0x001000011e0 -[ViewController buyInAppItem:]
        0x00100001240 -[ViewController moneyAmountLabel]
        0x00100001260 -[ViewController setMoneyAmountLabel:]


0x001000040a0 AppDelegate : UIResponder <UIApplicationDelegate>
        // instance methods
        0x001000012e0 -[AppDelegate application:didFinishLaunchingWithOptions:]
        0x00100001360 -[AppDelegate window]
        0x00100001380 -[AppDelegate setWindow:]


0x001000040c8 PurchaseManager : NSObject
        // instance methods
        0x00100001400 -[PurchaseManager init]
        0x00100001490 -[PurchaseManager addFunds:]
        0x001000014e0 -[PurchaseManager spend:]
        0x00100001550 -[PurchaseManager didChangeMoneyAmount]
        0x00100001580 -[PurchaseManager setDidChangeMoneyAmount:]
        0x001000015c0 -[PurchaseManager currentMoney]


0x00000000000 01 00 0300 /usr/lib/libobjc.A.dylib: NSObject
0x00000000000 01 00 0200 /System/Library/Frameworks/Foundation.framework/Foundation: NSString
0x00000000000 01 00 0600 /System/Library/Frameworks/UIKit.framework/UIKit: UIResponder
0x00000000000 01 00 0600 /System/Library/Frameworks/UIKit.framework/UIKit: UIViewController
~/L/D/X/D/M/B/P/D/MobiusStep3.app |
```

Ищем уязвимость

```
~/L/D/X/D/M/B/P/D/MobiusStep3.app  dsdump --objc MobiusStep3 -vvv
0x00100004028 ViewController : UIViewController
        // instance methods
        0x00100000eb0 -[ViewController viewDidLoad]
        0x001000011e0 -[ViewController buyInAppItem:]
        0x00100001240 -[ViewController moneyAmountLabel]
        0x00100001260 -[ViewController setMoneyAmountLabel:]


0x001000040a0 AppDelegate : UIResponder <UIApplicationDelegate>
        // instance methods
        0x001000012e0 -[AppDelegate application:didFinishLaunchingWithOptions:]
        0x00100001360 -[AppDelegate window]
        0x00100001380 -[AppDelegate setWindow:]


0x001000040c8 PurchaseManager : NSObject
        // instance methods
        0x00100001400 -[PurchaseManager init]
        0x00100001490 -[PurchaseManager addFunds:]
        0x001000014e0 -[PurchaseManager spend:]
        0x00100001550 -[PurchaseManager didChangeMoneyAmount]
        0x00100001580 -[PurchaseManager setDidChangeMoneyAmount:]
        0x001000015c0 -[PurchaseManager currentMoney]


0x00000000000 01  00 0300 /usr/lib/libobjc.A.dylib: NSObject
0x00000000000 01  00 0200 /System/Library/Frameworks/Foundation.framework/Foundation: NSString
0x00000000000 01  00 0600 /System/Library/Frameworks/UIKit.framework/UIKit: UIResponder
0x00000000000 01  00 0600 /System/Library/Frameworks/UIKit.framework/UIKit: UIViewController
~/L/D/X/D/M/B/P/D/MobiusStep3.app
```

Взламываем код

Работает без Jailbreak

```objc
#import <UIKit/UIKit.h>
#import <objc/runtime.h>

IMP originalImplementation;

static void __attribute__((constructor)) initialize(void){
    NSLog(@"Framework loaded");

    Class class = NSClassFromString(@"PurchaseManager");
    Class injector = NSClassFromString(@"Injector");
    SEL originalSelector = NSSelectorFromString(@"spend:");
    SEL swizzledSelector = NSSelectorFromString(@"swizzled_spend:");

    Method originalMethod = class_getInstanceMethod(class, originalSelector);
    originalImplementation = method_getImplementation(originalMethod);
    Method swizzledMethod = class_getInstanceMethod(injector, swizzledSelector);

    method_exchangeImplementations(originalMethod, swizzledMethod);
}

@interface Injector : NSObject

@end

@implementation Injector

- (void)swizzled_spend:(int)amount {
    NSLog(self.description);
    void (*callableImp)(id, SEL, int) = (typeof(callableImp)) originalImplementation;
    callableImp(self, NSSelectorFromString(@"spend:"), 1);
}

@end
```

Резюмируем

- Проверяйте на Jailbreak и на дебаггер, потому что это дешево и просто
- Не юзайте Objective-C, либо используйте по минимуму и не забудьте отрезать Swift Symbols
- Обфускация в общем хорошая идея, но используется редко и приносит неудобства с крэшлогами
- Привязываться к Hardware особенностям системы (Silent Push)
- Храните данные на сервере, потому что локальные хранилища (UserDefaults, Keychain, LocalDB) ненадёжны.
- Валидируйте покупки на сервере

И помнить - абсолютной безопасности не существует.
Но можно сделать безопаснее, чем было раньше

Полезное чтиво

- https://www.arm.com/
- https://www.mbo42.com/2018/04/01/hooking-swift-methods/
- https://medium.com/@kennethpoon/how-to-perform-ios-code-injection-on-ipa-files-1ba91d9438db
- https://www.reddit.com/r/jailbreak/
- https://labs.f-secure.com/blog/repacking-and-resigning-ios-applications/
- https://ss64.com/osx/security.html - macOS Security Utility

Telegram: @spartanrasul
Email: spartanrasul@yandex.ru