iOS App Security

Intro to iOS applications

Types of iOS applications

- **Native applications** – Developed in Objective-C or Swift. Deliver much better performance than web applications.
- **Web applications** – Developed in Html, CSS and javascript. Run using Webkit.
- **Hybrid applications** – Combination of a web and native application.

Objective-C

- Developed in 1984 by nEXT
- Taken over by Apple for iOS and Mac OS X
- Object Oriented , superset of C
- Used by Apple for iOS/OSX and also Cocoa , Cocoa Touch etc
- Provides a dynamic runtime (via a **dylib** injected into the process)
- Objective-C methods can be **swizzled** or class methods can be added at runtime. Uses the core principle of message sending via the **objc_msgSend()** runtime function
- All objects in Obj-C are **pointers**

Objective-C (continued)

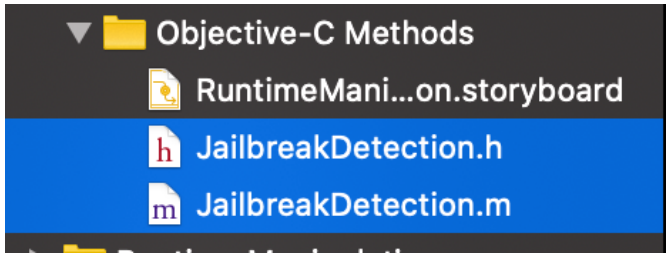
- The objc_msgSend() function takes a **target**, a **selector** and a list of arguments. For e.g
- **[array insertObject:foo atIndex:2];**
- will convert to
- **objc_msgSend(array, @selector(insertObject:atIndex:), foo, 5);**
- This concept is used when applying breakpoints and manipulating functions at runtime
- Register x0 is target, x1 is selector and x2 onwards are the arguments
- You can set breakpoint and make any modifications
- In essence, each method call leads to a **objc_msgSend()** function call
- This is known as **message dispatching**

Objective-C (continued)

```
prateekg147@Prateek ~ % lldb /bin/ls
(lldb) target create "/bin/ls"
Current executable set to '/bin/ls' (x86_64).
(lldb) b objc_msgSend
Breakpoint 1: where = libobjc.A.dylib`objc_msgSend, address = 0x0000000000005e40
(lldb) r /var/
Process 64999 launched: '/bin/ls' (x86_64)
1 location added to breakpoint 1
Process 64999 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = breakpoint 1.2
    frame #0: 0x00007fff6a41a800 libobjc.A.dylib`objc_msgSend
libobjc.A.dylib`objc_msgSend:
->  0x7fff6a41a800 <+0>:  testq  %rdi, %rdi
    0x7fff6a41a803 <+3>:  je     0x7fff6a41a878            ; <+120>
    0x7fff6a41a806 <+6>:  testb  $0x1, %dil
    0x7fff6a41a80a <+10>: jne    0x7fff6a41a883            ; <+131>
Target 0: (ls) stopped.
(lldb) register read
```

Objective-C (continued)

- Files with the extension **.h** are header files and **.m** are implementation files. Every header file has a corresponding implementation file



- Methods starting with **+** are class methods and those starting with **-** are instance or object methods.

```
+(BOOL)isJailbroken {
    //Target will be the class
    //Check for jailbroken device
}
-(BOOL)isJailbroken {
    //Target will be the object itself
    //Check for jailbroken device
}
```

Objective-C (continued)

```
MyObject *o = [[MyObject alloc] init];
MyObject *o = [[MyObject alloc] initWithString:myString];
NSArray *myArray = [NSArray arrayWithObjects:object1,object2,object3,nil];
```

Calling Methods

```
[self doSomething];
[MainViewController performTaskWithID:[NSNumber numberWithInt:1]];
```

Objective-C (continued) - method analysis

DVIA-v2 〉 DVIA-v2 〉 Vulne...ilities 〉 Jailbr...ction 〉 Obje...ethods 〉 JailbreakDetection.m 〉 No Selection

```
15
16  #import <Foundation/Foundation.h>
17  #import "JailbreakDetection.h"
18
19  @implementation JailbreakDetection
20
21  +(BOOL)isJailbroken{
22
23  #if !(TARGET_IPHONE_SIMULATOR)
24
25      if ([[NSFileManager defaultManager]
            fileExistsAtPath:@"/Applications/Cydia.app"]){
26          return YES;
27      }else if([[NSFileManager defaultManager]
            fileExistsAtPath:@"/Library/MobileSubstrate/MobileSubstrate.dylib"]){
28          return YES;
29      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/bin/bash"]){
30          return YES;
31      }else if([[NSFileManager defaultManager]
            fileExistsAtPath:@"/usr/sbin/sshd"]){
32          return YES;
33      }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/etc/apt"]){
34          return YES;
35      }
```

Objective-C (continued)

- Every class has a methods list
- This list is a dictionary, where the keys are the method name and the value is a **pointer** to the methods implementation in memory
- This connection is determined at **runtime** rather than compile time
- It is possible to change the pointer to the implementation. This technique is known as method swizzling
- The idea is to create a method in memory and replace the pointer of an existing method with your new method
- We will discuss method swizzling in detail later in this course

Introduction to Swift

- Introduced in 2014. Object Oriented and Open Source https://github.com/apple/swift
- Will replace Objective-C for iOS development ... eventually (Sure !!)
- Even without a single line of Objective-C code, every Swift app executes inside the Objective-C runtime.
- This may not always be the case – with the release of Swift-only system frameworks a full Swift runtime may appear.

Introduction to Swift

- The important thing is that instead of **objc_msgSend()** method call, Swift classes operate with **vtable** (which contains the table of available methods).
- **vtables** are created at compile time (when objc dispatch tables are generated dynamically). Contain function pointers accessed by index
- This table contains functions pointers accessed by index, so it doesn't need to bind selector to implementation.
- Called static dispatch, decision is not left until runtime to pick an implementation opposite of dynamic dispatch.

Swift – Continued

- From a security testing point-of-view, most of the stuff we'll be doing will be the same.
- The issues arise due to poor coding and design choices rather than the language used.
- You might use the most secure of languages, however an incorrect implementation of the libraries will still leave your code vulnerable
- We will be taking a black box testing approach, so language will not matter.
- Main difference during Reverse Engineering.

Objective-C symbols

```
mobile@ubuntu:~/Desktop/DVIA-Decrypted-Binary$ ARCH=arm64 jtool2 -S DamnVulnerableiOSApp | head -n 10
Note: 83513 symbols detected in this file! This will take a little while - generate a companion file or use '-q'
 for quick mode..
0000000100004500 t -[YapDatabaseViewState init]
000000010000466c t -[YapDatabaseViewState initForCopy]
000000010000471c t -[YapDatabaseViewState group_pagesMetadata_dict_deepCopy]
0000000100004854 t ___57-[YapDatabaseViewState group_pagesMetadata_dict_deepCopy]_block_invoke
0000000100004974 t ___copy_helper_block_
00000001000049c0 t ___destroy_helper_block_
00000001000049f0 t -[YapDatabaseViewState copyWithZone:]
0000000100004b60 t -[YapDatabaseViewState mutableCopyWithZone:]
0000000100004c94 t -[YapDatabaseViewState pagesMetadataForGroup:]
0000000100004d20 t -[YapDatabaseViewState groupForPageKey:]
mobile@ubuntu:~/Desktop/DVIA-Decrypted-Binary$
```

Swift symbols

```
00000001001c1ca0 T __T07DVIA_v213CoreDataStackC26persistentStoreCoordinatorSo012NSPersist
entgH0Cfg
00000001001c1e48 t __T07DVIA_v213CoreDataStackC26persistentStoreCoordinatorSo012NSPersist
entgH0CfgAFycfU_
00000001001c2874 T __T07DVIA_v213CoreDataStackC26persistentStoreCoordinatorSo012NSPersist
entgH0Cfm
00000001001c2850 t __T07DVIA_v213CoreDataStackC26persistentStoreCoordinatorSo012NSPersist
entgH0CfmytfU_
00000001001c27c0 T __T07DVIA_v213CoreDataStackC26persistentStoreCoordinatorSo012NSPersist
entgH0Cfs
00000001001c13e8 T __T07DVIA_v213CoreDataStackC29applicationDocumentsDirectory10Foundatio
n3URLVfg
00000001001c1570 t __T07DVIA_v213CoreDataStackC29applicationDocumentsDirectory10Foundatio
n3URLVfgAGycfU_
00000001001c1724 T __T07DVIA_v213CoreDataStackC29applicationDocumentsDirectory10Foundatio
n3URLVfm
00000001001c1700 t __T07DVIA_v213CoreDataStackC29applicationDocumentsDirectory10Foundatio
n3URLVfmytfU_
00000001001c1670 T __T07DVIA_v213CoreDataStackC29applicationDocumentsDirectory10Foundatio
n3URLVfs
00000001001c09cc T __T07DVIA_v213CoreDataStackC6sharedACfau
00000001001c0a2c T __T07DVIA_v213CoreDataStackC6sharedACfgZ
000000010044ee48 S __T07DVIA_v213CoreDataStackC6sharedACvZ
00000001001c09a4 T __T07DVIA_v213CoreDataStackCACyc33_4FFA85D5E28BA6BC2BCBFAA24CA3DACCLlf
```

Introduction to Swift - Reversing

- **TheTarget:**
- __T018hello_world_swift114ViewControllerC15mainButtonClickyypFTo
- __T - Indication that it is a Swift method
- **1hello_world_swift18** – Module name with length
- **14ViewController** – Class name with length
- **C** – Indication that it is a function of a class
- **15mainButtonClick** – Function name with length
- **f** – Function attribute
- **T** – parameters (zero in this case else T_)
- **T_** - Indicates the return type

Reversing Swift Applications

Function Attributes

- c – Constructor function
- C – Allocator function
- d – Destructor function
- D – Deallocator function
- g – Getter function
- f – Normal function
- s – Setter function

Parameter Types

- a – Array
- b – Boolean
- c – Unicode Scalar
- d – Double
- f – Float
- i – Integer
- u – Unsigned Integer
- Q – Implicitly Unwrapped Optional
- S – String
- T – Tuple
- O - enum

Swift-demangle using xcrun

```
⇒  xcrun swift-demangle -expand __T018hello_world_swift114ViewControllerC15mainButtonCli
ckyyFTo
Demangling for _T018hello_world_swift114ViewControllerC15mainButtonClickyyFTo
kind=Global
  kind=ObjCAttribute
  kind=Function
    kind=Class
      kind=Module, text="hello_world_swift1"
      kind=Identifier, text="ViewController"
    kind=Identifier, text="mainButtonClick"
    kind=Type
      kind=FunctionType
        kind=ArgumentTuple
          kind=Type
            kind=Tuple
        kind=ReturnType
          kind=Type
            kind=Tuple
_T018hello_world_swift114ViewControllerC15mainButtonClickyyFTo ---> @objc hello_world_sw
ift1.ViewController.mainButtonClick() -> ()
```

- Hopper has in-build demangling of functions

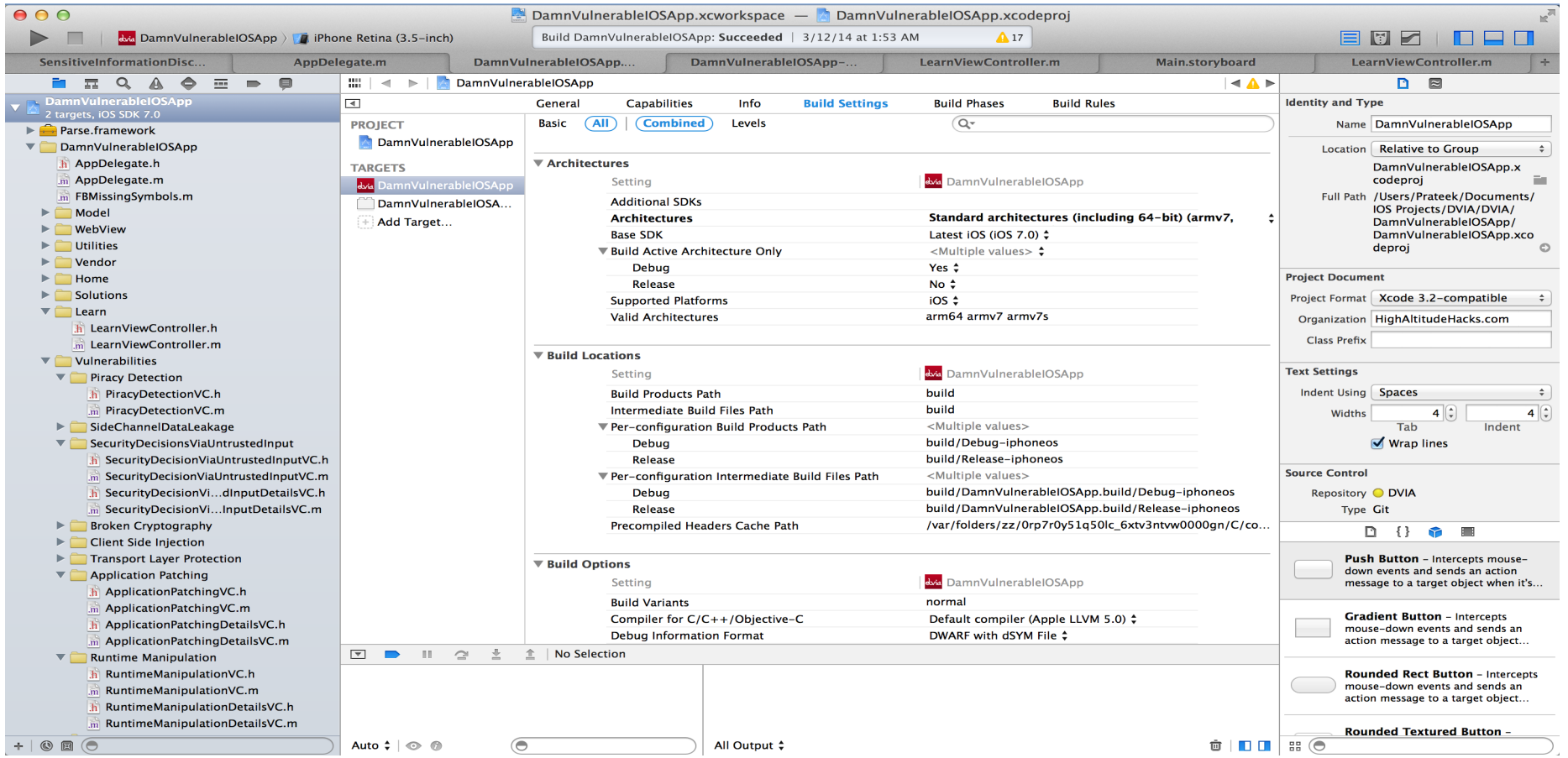
Some definitions

- **Delegate** - An object which gets notified when the object to which it is connected reaches certain events or states.
- **AppDelegate** - Connected to the App, gets when the UIApplication object reaches certain states. Can be used for some security events as well, for e.g clearing pasteboard data when the app goes to background
- **View Controller** - Manages a portion of your app's user interface as well as all interactions. Think as if each screen is a different view controller.
- **Storyboard** - Visual representation of the user interface of an iOS application

Xcode

IDE for developing native iOS applications: https://

developer.apple.com/xcode/



iOS app frameworks

- Best Mobile App development frameworks 2021 - https://mindster.com/mobile-app-development-frameworks/
- Some popular frameworks are React Native, Flutter etc
- Client Code is written mostly in Javascript

iOS security model

- All app store apps are signed using apple, and protected by code signing.
- All apps running on the device by the developer are signed by the developer's certificate.
- Every application runs inside a closed environment known as an application sandbox.
- Apps inside the sandbox cannot access data from other apps, except through certain mechanisms like protocol handlers.
- Unsigned/modified apps cannot be installed on a non-jailbroken device.
- Access to all resources (such as files, network sockets, IPCs, and shared memory) are managed by the sandbox

iOS security model - continued

- Certain system calls such as **mmap** and **mprotect** are intentionally broken in iOS to prevent attacks, for e.g setting +w is disallowed on a +x segment (and vice versa)
- Apps are confined to access data to its own directory
- Access to hardware drivers are done through public frameworks
- Address space layout randomization (**ASLR**) provide address randomisation, prevents against overflow and return-to-libc attacks
- The eXecute Never (**XN**) bit is used to mitigate code execution attacks
- Pages that are writable cannot be marked executable at the same time

Analyzing iOS apps

- IPA files are basically a zipped version of the app
- Change name to from **DVIA.ipa** to **DVIA.zip** and unzip the folder
- It creates a folder named **Payload** which contains a folder **DVIA.app**
- Inside **DVIA.app**, you can find the application binary
- Open the binary in Hopper to reverse it
- To repackage the binary, just compress the Payload folder and rename it from **Payload.zip** to **AppName.ipa**. Depending on the JB device you install, you might need to sign the app before installing.

Xcode Walkthrough

Task - 10 mins

- Find the IPA files under the folder **iOS-PreCourse -> IPA folder** provided in the Pre-course **downloadables**
- Unzip the **DVIA.ipa** and **DVIA-v2-Swift.ipa** files to get to the application binary
- Open these binaries in Hopper
- Dump the symbols for these binaries using **jtool2**
- Have a quick look at the source code for DVIA https://github.com/prateek147/DVIA and DVIA-v2 https://github.com/prateek147/DVIA-v2

Binary Analysis - Security measures

- **Position independent executable (PIE)** - Loads the executable at a different address each time. Apple support PIE for iOS apps by default. There’s no hard requirement yet and non-PIE applications are not rejected. Having ASLR means the code can't be a part of ROP gadgets.

- **Automatic reference counting (ARC)** - Enabled autorelease of objects. Protects against memory corruption vulnerabilities, specially object use-after-free vulnerabilities.

Binary Analysis

- **Stack smashing protection** - Adds a known value to be placed on the stack directly before the local variables. Protects the saved base pointer, saved instruction pointer, and function arguments to be overwritten. Detects buffer overflow attacks when stack is corrupted, can terminate the app on detecting this. Protects against buffer overflow attacks.

- It is recommended to have all these settings in your application before submitting to the App store.

Checking for PIE - otool

```
Prateek:Desktop prateekg147$ otool -hv DamnVulnerableiOSApp
DamnVulnerableiOSApp (architecture armv7):
Mach header
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
   MH_MAGIC     ARM         V7  0x00     EXECUTE    38       4292   NOUNDEFS DYL
DLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
DamnVulnerableiOSApp (architecture arm64):
Mach header
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
MH_MAGIC_64   ARM64         ALL  0x00     EXECUTE    38       4856   NOUNDEFS DYL
DLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
Prateek:Desktop prateekg147$
```

Checking for PIE - jtool

```
prateek:ARCH=arm64 jtool -pie DamnVulnerableIOSApp
Toggling Mach-O Header MH_PIE: off
Not a 64-bit or 32-bit Mach-O
Mach-O is already PIE disabled
prateek:ARCH=arm64 jtool +pie DamnVulnerableIOSApp
Toggling Mach-O Header MH_PIE: on
Not a 64-bit or 32-bit Mach-O
Warning: Destructive option. Output (4445680 bytes) written to out.bin
prateek:ARCH=arm64 jtool +pie DamnVulnerableIOSApp
```

Checking for Stack Smashing protection - otool

```
Prateek:Desktop prateekg147$ otool -I -v DamnVulnerableiOSApp | grep stack
0x00000001004b3fd8 83077 ___stack_chk_fail
0x00000001004b4890 83414 _sigaltstack
0x0000000100590cf0 83078 ___stack_chk_guard
0x00000001005937f8 83077 ___stack_chk_fail
0x0000000100593dc8 83414 _sigaltstack
Prateek:Desktop prateekg147$
```

Presence of **__stack_chk_fail** and **__stack_chk_guard** symbols confirms that the application is compiled with stack smashing protection.

Checking for ARC - otool

```
Prateek:Desktop prateekg147$ otool -I -v DamnVulnerableiOSApp | grep release
0x0046133c 82607 _CFAutorelease
0x004fe994 82607 _CFAutorelease
0x00000001004b0708 83066 ___cxa_guard_release
0x00000001004b3b70 83308 _objc_autorelease
0x00000001004b3b7c 83309 _objc_autoreleasePoolPop
0x00000001004b3b88 83310 _objc_autoreleasePoolPush
0x00000001004b3b94 83311 _objc_autoreleaseReturnValue
0x00000001004b3ca8 83334 _objc_release
0x00000001004b3cc0 83336 _objc_retainAutorelease
0x00000001004b3ccc 83337 _objc_retainAutoreleaseReturnValue
0x00000001004b3cd8 83338 _objc_retainAutoreleasedReturnValue
0x00000001004b3f60 82959 __Block_release
0x00000001004b4218 83152 _dispatch_release
0x00000001004b4af4 82519 _CFAutorelease
0x0000000100591218 83066 ___cxa_guard_release
0x0000000100593508 83308 _objc_autorelease
0x0000000100593510 83309 _objc_autoreleasePoolPop
0x0000000100593518 83310 _objc_autoreleasePoolPush
0x0000000100593520 83311 _objc_autoreleaseReturnValue
```

rabin2 - Get Binary Info

```
Prateek:Desktop prateekg147$ rabin2 -I DVIA-v2-Swift-Fat
arch     arm
binsz    7525168
bintype  mach0
bits     64
canary   true
class    MACH064
crypto   false
endian   little
havecode true
intrp    /usr/lib/dyld
lang     c
linenum  false
lsyms    false
machine  v8
maxopsz  16
minopsz  1
nx       false
os       ios
pcalign  0
pic      true
relocs   false
static   false
stripped true
subsys   darwin
va       true
```

rabin2 - Find imports

```
Prateek:Desktop prateekg147$ rabin2 -i DVIA-v2-Swift-Fat | head -n 30
[Imports]
ordinal=000 plt=0x00000000 bind=NONE type=FUNC name=AVCaptureDeviceTypeBuiltInWideAng
leCamera
ordinal=001 plt=0x00000000 bind=NONE type=FUNC name=AVLayerVideoGravityResizeAspectFi
ll
ordinal=002 plt=0x00000000 bind=NONE type=FUNC name=AVMediaTypeVideo
ordinal=003 plt=0x10032859c bind=NONE type=FUNC name=CCCrypt
ordinal=004 plt=0x1003285a8 bind=NONE type=FUNC name=CCCryptorCreate
ordinal=005 plt=0x1003285b4 bind=NONE type=FUNC name=CCCryptorFinal
ordinal=006 plt=0x1003285c0 bind=NONE type=FUNC name=CCCryptorGetOutputLength
ordinal=007 plt=0x1003285cc bind=NONE type=FUNC name=CCCryptorRelease
ordinal=008 plt=0x1003285d8 bind=NONE type=FUNC name=CCCryptorUpdate
ordinal=009 plt=0x1003285e4 bind=NONE type=FUNC name=CCHmac
ordinal=010 plt=0x1003285f0 bind=NONE type=FUNC name=CCHmacFinal
ordinal=011 plt=0x1003285fc bind=NONE type=FUNC name=CCHmacInit
ordinal=012 plt=0x100328608 bind=NONE type=FUNC name=CCHmacUpdate
ordinal=013 plt=0x100328614 bind=NONE type=FUNC name=CCKeyDerivationPBKDF
ordinal=014 plt=0x100328620 bind=NONE type=FUNC name=CC_MD5
ordinal=015 plt=0x10032862c bind=NONE type=FUNC name=CC_MD5_Final
ordinal=016 plt=0x100328638 bind=NONE type=FUNC name=CC_MD5_Init
ordinal=017 plt=0x100328644 bind=NONE type=FUNC name=CC_MD5_Update
ordinal=018 plt=0x100328650 bind=NONE type=FUNC name=CC_SHA1
ordinal=019 plt=0x10032865c bind=NONE type=FUNC name=CC_SHA1_Final
ordinal=020 plt=0x100328668 bind=NONE type=FUNC name=CC_SHA1_Init
ordinal=021 plt=0x100328674 bind=NONE type=FUNC name=CC_SHA1_Update
ordinal=022 plt=0x100328680 bind=NONE type=FUNC name=CC_SHA256
```

rabin2 - list symbols

```
Prateek:Desktop prateekg147$ rabin2 -s DVIA-v2-Swift-Fat | head -n 20
[Symbols]
vaddr=0x100287410 paddr=0x00287410 ord=000 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_$equal
vaddr=0x10028d4ec paddr=0x0028d4ec ord=001 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_$regex
vaddr=0x100230864 paddr=0x00230864 ord=002 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_AllLogDomains
vaddr=0x10023094c paddr=0x0023094c ord=003 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_AlwaysLog
vaddr=0x10044de30 paddr=0x0044de30 ord=004 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_ArrayDiff_LogDomain
vaddr=0x1002bb928 paddr=0x002bb928 ord=005 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPEncodeProperties
vaddr=0x1003b2ca8 paddr=0x003b2ca8 ord=006 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPErrorDomain
vaddr=0x10044df00 paddr=0x0044df00 ord=007 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPLifecycle_LogDomain
vaddr=0x1002b1ce8 paddr=0x002b1ce8 ord=008 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPMakeError
vaddr=0x1002bb5f0 paddr=0x002bb5f0 ord=009 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPParseProperties
vaddr=0x1002bb874 paddr=0x002bb874 ord=010 fwd=NONE sz=0 bind=GLOBAL type
=FUNC name=_BLIPReadPropertiesFromBuffer
vaddr=0x10044dee8 paddr=0x0044dee8 ord=011 fwd=NONE sz=0 bind=GLOBAL type
```

rabin2 - list sections

```
Prateek:Desktop prateekg147$ rabin2 -S DVIA-v2-Swift-Fat | head -n 20
[Sections]
idx=00 vaddr=0x100004ab8 paddr=0x00004ab8 sz=3291876 vsz=3291876 perm=m-r
-x name=0.__TEXT.__text
idx=01 vaddr=0x10032859c paddr=0x0032859c sz=9288 vsz=9288 perm=m-r-x nam
e=1.__TEXT.__stubs
idx=02 vaddr=0x10032a9e4 paddr=0x0032a9e4 sz=8988 vsz=8988 perm=m-r-x nam
e=2.__TEXT.__stub_helper
idx=03 vaddr=0x10032cd00 paddr=0x0032cd00 sz=141358 vsz=141358 perm=m-r-x
 name=3.__TEXT.__objc_methname
idx=04 vaddr=0x10034f530 paddr=0x0034f530 sz=201374 vsz=201374 perm=m-r-x
 name=4.__TEXT.__cstring
idx=05 vaddr=0x1003807ce paddr=0x003807ce sz=8038 vsz=8038 perm=m-r-x nam
e=5.__TEXT.__objc_classname__TEXT
idx=06 vaddr=0x100382734 paddr=0x00382734 sz=25916 vsz=25916 perm=m-r-x n
ame=6.__TEXT.__objc_methtype
idx=07 vaddr=0x100388c70 paddr=0x00388c70 sz=34900 vsz=34900 perm=m-r-x n
ame=7.__TEXT.__gcc_except_tab__TEXT
idx=08 vaddr=0x1003914d0 paddr=0x003914d0 sz=25128 vsz=25128 perm=m-r-x n
ame=8.__TEXT.__const
idx=09 vaddr=0x100397700 paddr=0x00397700 sz=11554 vsz=11554 perm=m-r-x n
ame=9.__TEXT.__swift3_typeref__TEXT
idx=10 vaddr=0x10039a430 paddr=0x0039a430 sz=3455 vsz=3455 perm=m-r-x nam
e=10.__TEXT.__swift3_reflstr__TEXT
idx=11 vaddr=0x10039b1b0 paddr=0x0039b1b0 sz=3340 vsz=3340 perm=m-r-x nam
e=11.__TEXT.__swift3_fieldmd__TEXT
idx=12 vaddr=0x10039bec0 paddr=0x0039bec0 sz=448 vsz=448 perm=m-r-x name=
```

Task - 5 mins

- On your Ubuntu VM, head under the directory **Desktop/DVIA-Decrypted-Binary**
- Run the commands for **rabin2** mentioned in the earlier slides
- Check any similar options provided in **jtool2**

Touch ID/ Face ID

- Electronic fingerprint or face recognition feature
- 1 in 50000 chance that someone else can unlock your phone
- After 5 unsuccessful attempts, it is required to enter Passcode
- Can be implemented in Apps as well for authentication
- Can be bypassed when using insecure APIs (canEvaluatePolicy:)
- Data protected using Secure Enclave, and never leaves the device
- Data Stored using a mathematical representation in the Secure Enclave, and hence can’t be reverse engineered

Secure enclave

- Separate co-processor independent from the AP (Application processor)
- Responsible for all cryptographic operations for Data Protection key management Includes a hardware-based key manager
- Runs its own OS called **SEPOS**, and has its own kernel, drivers, services, apps etc
- Secure boot mechanism and software update mechanism independent from the AP
- Maintains integrity even if the Kernel is compromised
- SE creates the key, stores it and performs operations with it without ever leaving it. Much better to store key there than storing in the keychain
- No plain text key in memory like keychain

Setting up an iOS pen testing environment

What do you need?

- A jailbroken iOS device (or Corellium)
- Ubuntu VM
- Some tools that need to be installed on your jailbroken device
- Some tools installed on your Ubuntu VM
- Set up the device by running the script at http://damnvulnerableiosapp.com/setupdevice.sh (Make sure to launch Cydia first if doing this for the first time in a device)

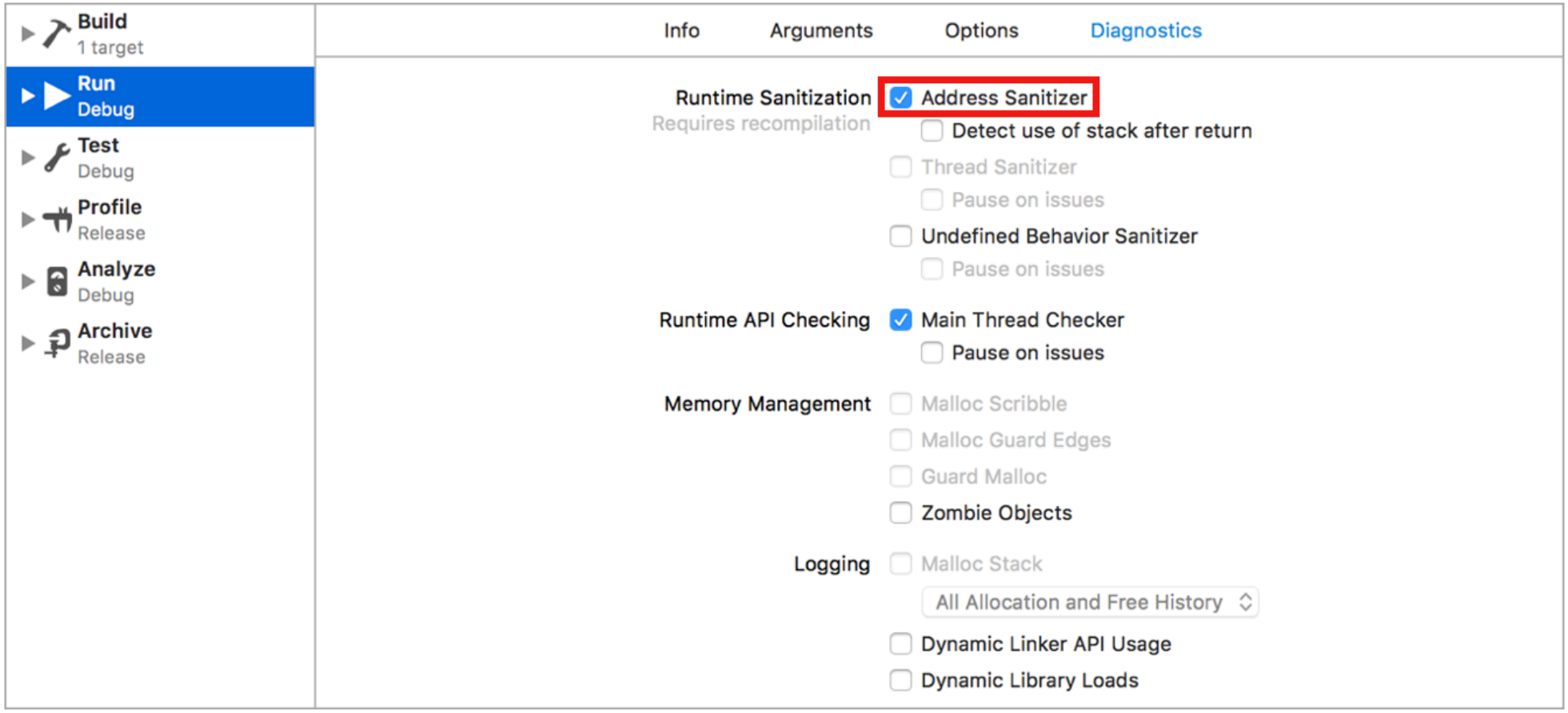
setupdevice.sh

```
#!/bin/sh

……..

echo "deb https://cydia.akemi.ai/ ./" >> /var/mobile/Library/Caches/com.saurik.Cydia/sources.list
echo "deb https://build.frida.re ./" >> /var/mobile/Library/Caches/com.saurik.Cydia/sources.list

apt-get -y update
apt-get -y install unzip
apt-get -y install mobilesubstrate
apt-get -y install python3
apt-get -y install lldb

apt-get -y --allow-unauthenticated install re.frida.server
apt-get -y --allow-unauthenticated install net.angelxwind.appsyncunified
apt-get -y --allow-unauthenticated install com.linusyang.appinst
apt-get -y --allow-unauthenticated install com.cannathea.afc2d-arm64

wget https://damnvulnerableiosapp.com/t.zip
unzip t.zip
cd iOS-Device-Tools/

cp jtool2 /usr/bin/ && cp keychain_dumper /usr/bin/ && cp debugserver /usr/bin/ && cp binbag /usr/bin/ && chmod +x /usr/bin/binbag && cp rop /usr/bin/ && cp vuln /usr/bin/ &&
&& chmod +x /usr/bin/keychain_dumper && chmod +x /usr/bin/ustack


appinst DamnVulnerableiOSApp.ipa
appinst DVIA-v2-swift.ipa

frida-server -l 0.0.0.0 &

killall -9 backboardd
```

Address Sanitizer - Xcode

- The Address Sanitizer feature in Xcode can be used to do an initial check for the following vulnerabilities in the iOS application
  - Use of Deallocated Memory
  - Deallocation of Deallocated Memory
  - Deallocation of Nonallocated Memory
  - Use of Stack Memory After Function Return
  - Use of Out-of-Scope Stack Memory
  - Overflow and Underflow of Buffers
  - Overflow of C++ Containers

Enabling Address Sanitizer - Xcode



https://developer.apple.com/documentation/code_diagnostics/address_sanitizer/enabling_the_address_sanitizer

Task - 5 mins

- Look at the different options provided by Address Sanitizer
- Recommended reading - https://medium.com/@cristianarielbarril/finding-runtime-bugs-using-xcode-9-tools-43c0fa61654d

Homework - Day 2 (Part 1)

- Look at the following video to understand more about Address Sanitizer https://developer.apple.com/videos/play/wwdc2015/413/
- Read about the following bug - https://awakened1712.github.io/hacking/hacking-whatsapp-gif-rce/

Jailbreaking for iOS app testing

- Remove limitations imposed by Apple
- Provides root level access to the device
- Install custom applications that don’t meet Apple’s compliance requirements
- Gives ability to run self signed apps on the device.
- All code must be signed. In case of jailbroken devices, it can be used to run self signed applications (non CoreTrust devices).

Disclaimer

- Having a jailbroken device is not necessary for testing iOS Apps https://medium.com/securing/pentesting-ios-apps-without-jailbreak-91809d23f64e
- However, it makes the job easier and it allows us to use some automation tools much more easily
- Ideally, it is good to have one Jailbroken device and one non-jailbroken device with the latest iOS version

Task - 10 mins

- SSH to your **Corellium** device
- Explore the directory structure of the device.
- What is the difference between logging in as **mobile** user and **root** user ?
- See a list of the processes running on the device
- Find the location where the **DVIA** app is installed
- Find all the **sqlite** and **db** files located on the device
- Find location of **iMessage** database and **Contacts** database (Might not work on a virtual device)

Solution

ps aux to view all processes

find / -name *.db to view all database files

find / -name *.sqlite to view all sqlite files

Run DVIA and run the command **ps aux | grep DamnVulnerableIOSApp** to view location of the binary

Solution

```
iPhone:~ root# find  / -name *.db
/usr/libexec/topicsmap.db
/System/Library/PreinstalledAssetsV2/RequiredByOs/com_apple_MobileAsset_VoiceServices_VoiceRes
ources/a1ea4eb1d9b37f3adc953fe0cc16d4140f1811ee.asset/AssetData/pron_dict.db
/System/Library/PreinstalledAssetsV2/RequiredByOs/com_apple_MobileAsset_VoiceServices_VoiceRes
ources/a1ea4eb1d9b37f3adc953fe0cc16d4140f1811ee.asset/AssetData/phon_map.db
/System/Library/Frameworks/CallKit.framework/CallDirectoryTemplate.db
/System/Library/Frameworks/CoreTelephony.framework/Support/lasdgsm.db
/System/Library/Frameworks/CoreTelephony.framework/Support/lasdcdma.db
/System/Library/Frameworks/CoreTelephony.framework/Support/lasdscdma.db
/System/Library/Frameworks/CoreTelephony.framework/Support/lasdlte.db
/System/Library/Frameworks/CoreTelephony.framework/Support/lasdumts.db
/System/Library/Frameworks/WebKit.framework/HTTPSUpgradeList.db
/System/Library/Frameworks/CoreLocation.framework/Support/timezone.db
/System/Library/Frameworks/CoreLocation.framework/Support/factory_minimum.db
/System/Library/Frameworks/CoreLocation.framework/Support/factory.db
/System/Library/Frameworks/CoreLocation.framework/factory_minimum.db
/System/Library/PrivateFrameworks/AppSupport.framework/de.lproj/Localizable_Places.db
/System/Library/PrivateFrameworks/AppSupport.framework/he.lproj/Localizable_Places.db
```

Solution

```
iPhone:/private/var/wireless/Library/Databases root# ls
CellularUsage.db  DataUsage.sqlite  DataUsage.sqlite-shm  DataUsage.sqlite-wal
iPhone:/private/var/wireless/Library/Databases root# sqlite3 DataUsage.sqlite
SQLite version 3.24.0 2018-06-04 19:24:41
Enter ".help" for usage hints.
sqlite> .tables
ZCHECKUPEVENT   ZEVENTSCENE     ZPROCESS        Z_METADATA
ZDEMOLIVEUSAGE  ZLIVEUSAGE      ZTSHOOTINGDATA  Z_MODELCACHE
ZEVENT          ZPEER           ZWIFIDATA       Z_PRIMARYKEY
sqlite> select * from ZWIFIDATA
   ...> ;
sqlite> ;
```

Important directories

Logs, preferences, profiles

- /private/var/installd/Library/Logs/MobileInstallation/
- /private/var/log/
- /private/var/preferences/
- /private/var/root/Library/Preferences/
- /private/var/mobile/Library/Logs/
- /private/var/mobile/Library/Preferences/
- /private/var/mobile/Library/UserConfigurationProfiles/

System databases

- /private/var/wireless/Library/Databases/CellularUsage.db
- /private/var/wireless/Library/Databases/DataUsage.db
- /private/var/root/Library/Caches/locationd/consolidated.db
- /private/var/mobile/Media/Downloads/downloads.28.sqlitedb
- /private/var/mobile/Library/ApplePushService/aps.db
- /private/var/mobile/Library/FrontBoard/applicationState.db
- /private/var/mobile/Library/TCC/TCC.db

Call log and iMessage/SMS (temporary databases)

- /private/var/mobile/Library/CallHistoryDB/CallHistoryTemp.storedata
- /private/var/mobile/Library/SMS/sms-temp.db

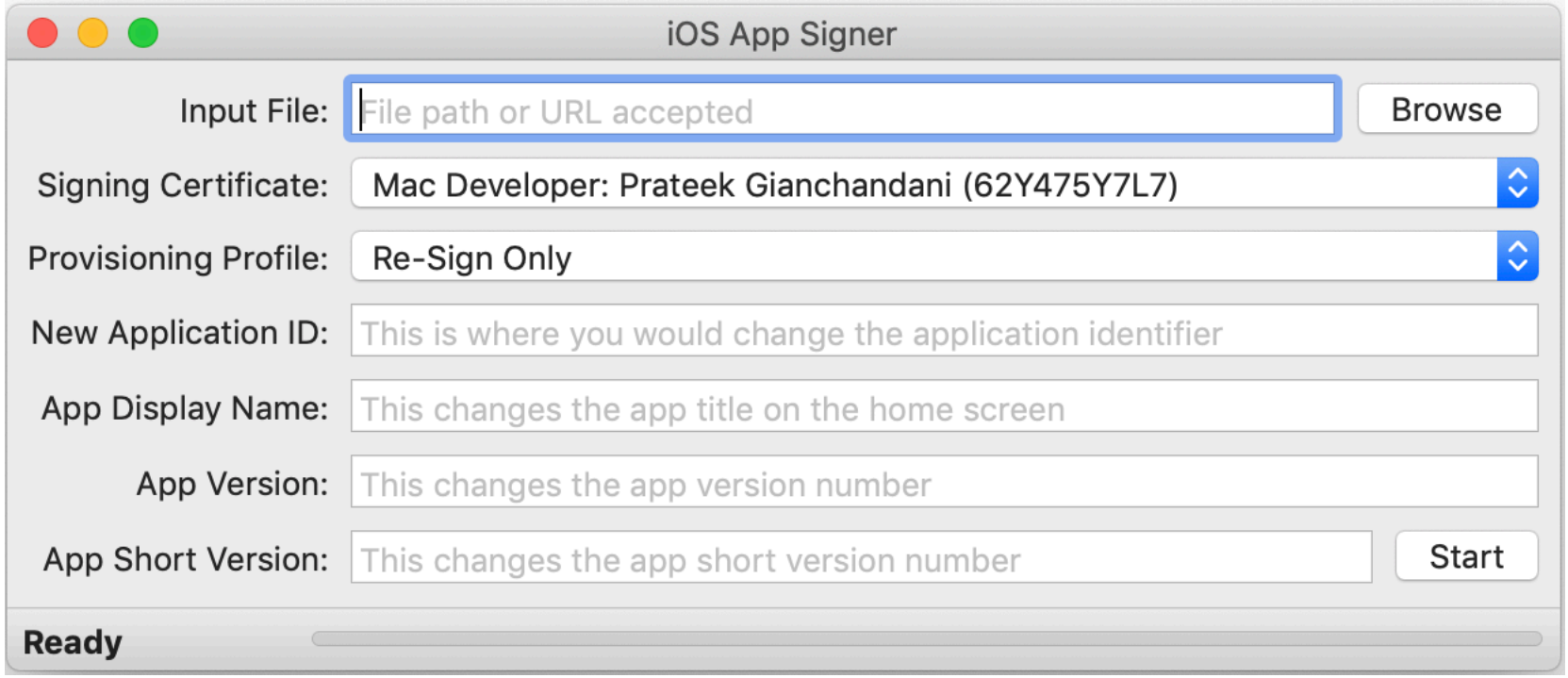
Voice mail

- /private/var/mobile/Library/Voicemail/voicemail.db

Source: https://blog.elcomsoft.com/2019/11/ios-device-acquisition-with-checkra1n-jailbreak/

Signing apps - iOS App Signer

- Can be done with iOS App Signer

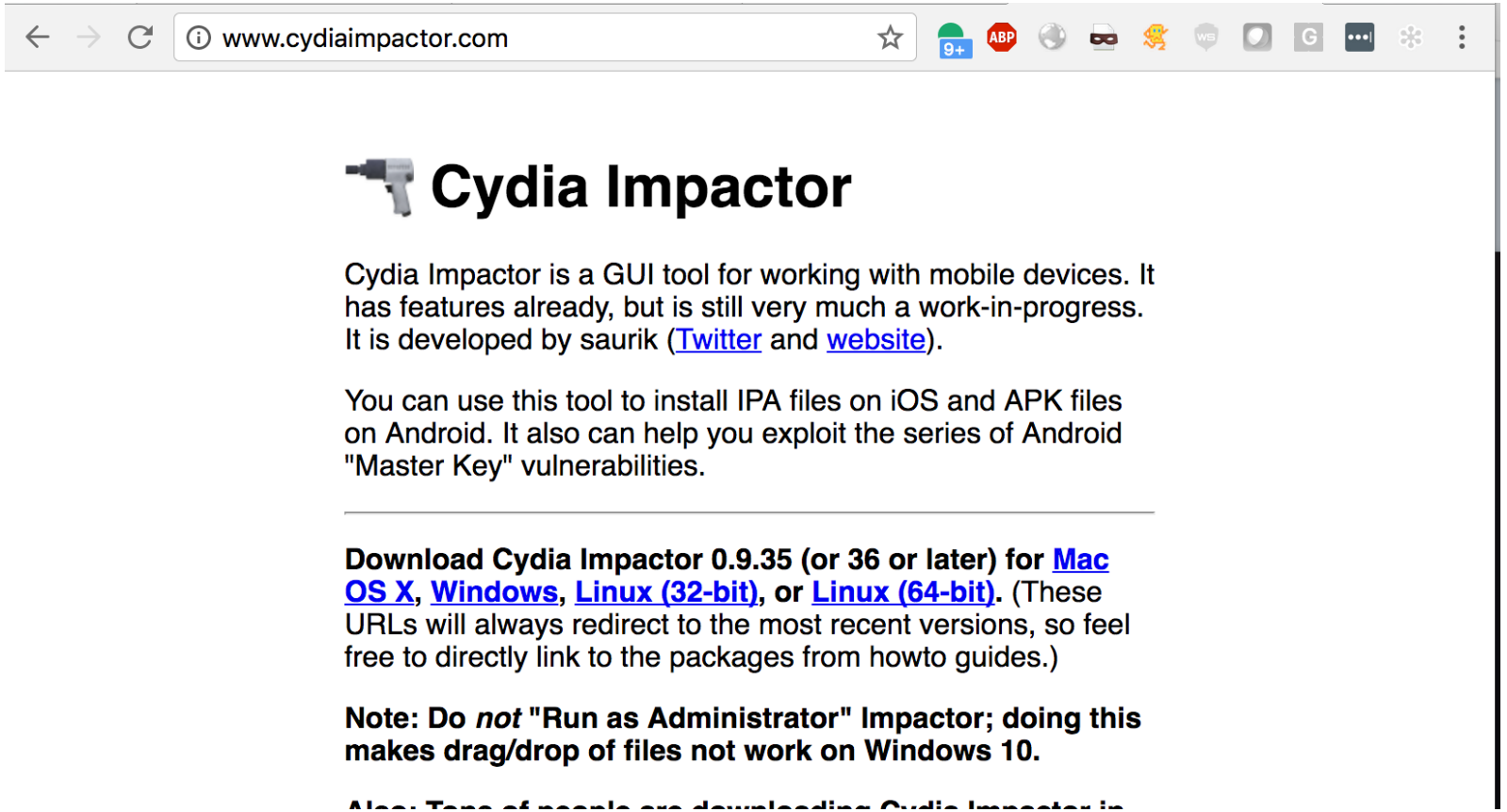


Signing apps - codesign using Apple certs

```
prateekg147@Prateek kernel_symbols % security find-identity -v
  1) 05D30ED26134E6851BF413E54A814C3D7B156E51 "iPhone Developer: Pra
teek Gianchandani (62Y475Y7L7)"
  2) 4CC09ABFE88C745842009EC4352572B09FAED9BC "Apple Development: Pr
ateek Gianchandani (62Y475Y7L7)"
     2 valid identities found
prateekg147@Prateek kernel_symbols % codesign -v -fs "4CC09ABFE88C74
5842009EC4352572B09FAED9BC" DVIA-v2.app
DVIA-v2.app: replacing existing signature
DVIA-v2.app: signed app bundle with Mach-O thin (arm64) [com.highalt
itudehacks.DVIAswiftv2]
prateekg147@Prateek kernel_symbols %
```

Sideloading apps - Cydia Impactor

- Requires a valid Apple Developer account
- Need to trust the developer explicitly once the application is installed

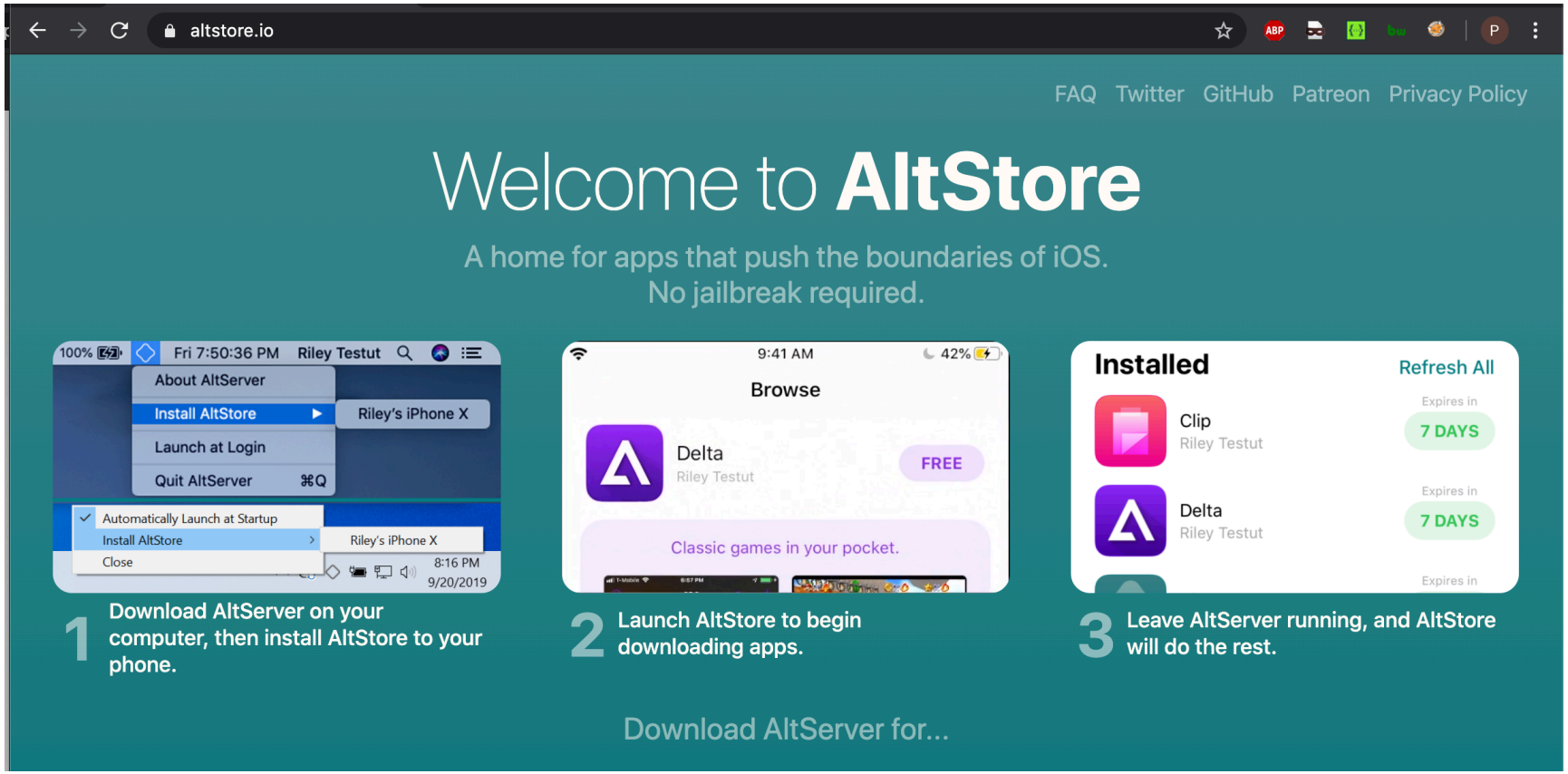


Sideloading apps - Xcode

- Requires a valid Apple Developer account
- Connect your device to your Mac and launch Xcode
- In Xcode go to **Window** -> **Devices and Simulators**
- Then just drag and drop the file into the apps section

Sideloading apps - Altstore (altstore.io)

- Doesn't require a developer account



Sideloading apps - appinst

- Add the source https://cydia.akemi.ai/ in Cydia
- Install **AppSync unified**

  apt-get -y --allow-unauthenticated install net.angelxwind.appsyncunified
- Install **appinst**

  apt-get -y --allow-unauthenticated install com.linusyang.appinst
- Use **appinst** to install apps

  appinst DVIA-v2-swift.ipa

Sideloading apps - manual method

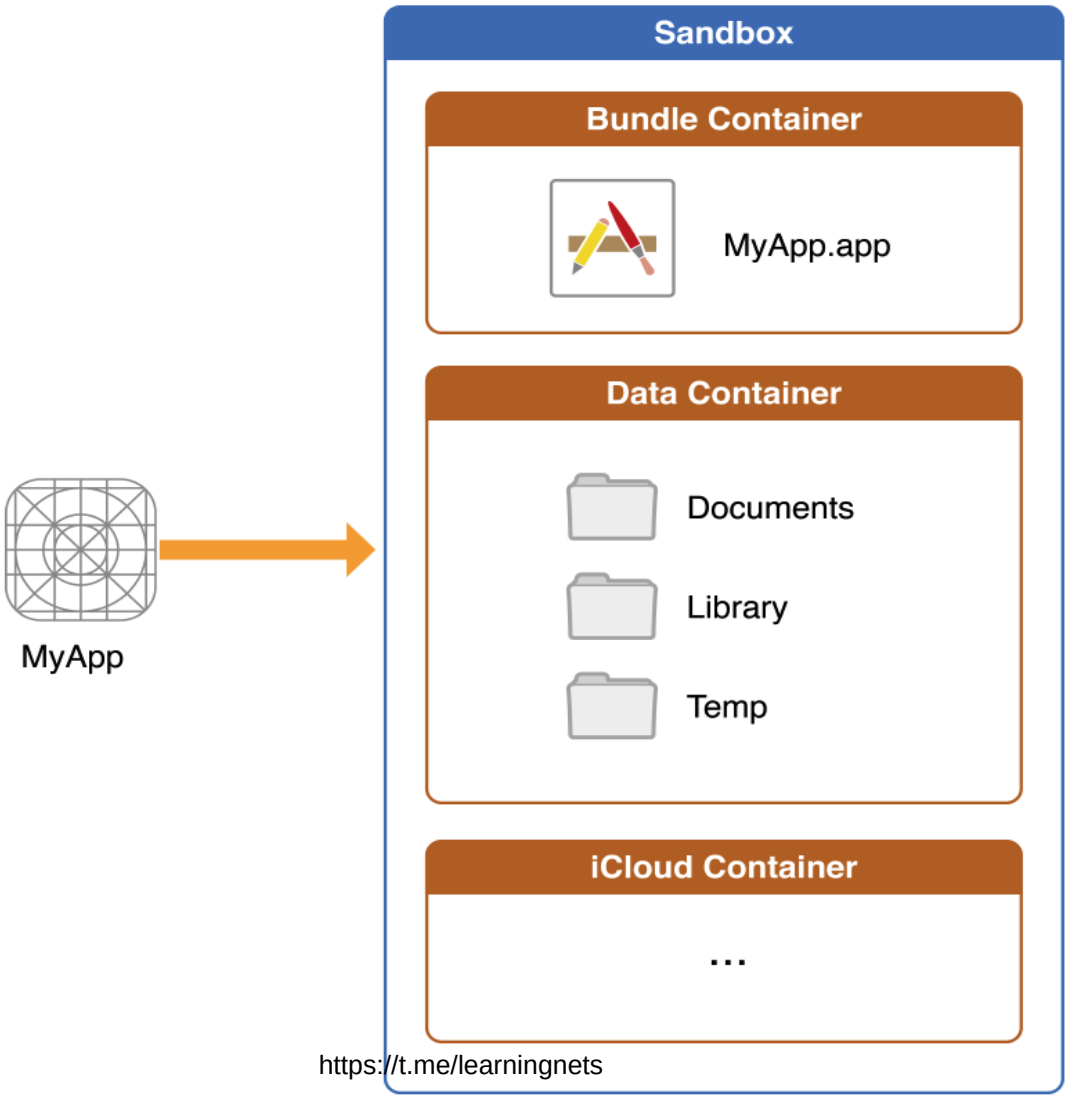
- Unzip the **ipa** file by Changing the extension to **.zip** and then expanding it and looking inside the **Payload** folder
- From inside the **Payload** folder, Copy the **.app** folder to the location **/Applications** in the device (**scp -r AntiDebug.app/ root@10.11.1.1:/Applications**)
- Give executable permissions to the app binary, for e.g **chmod +x /Applications/AnitiDebug.app/AntiDebug**
- On the device, run the command **uicache** to refresh the interface
- The app should now show up on the device

Task - 5 mins

- Install **AntiDebug.ipa** from **Day2** material using the manual method as discussed in the previous slide

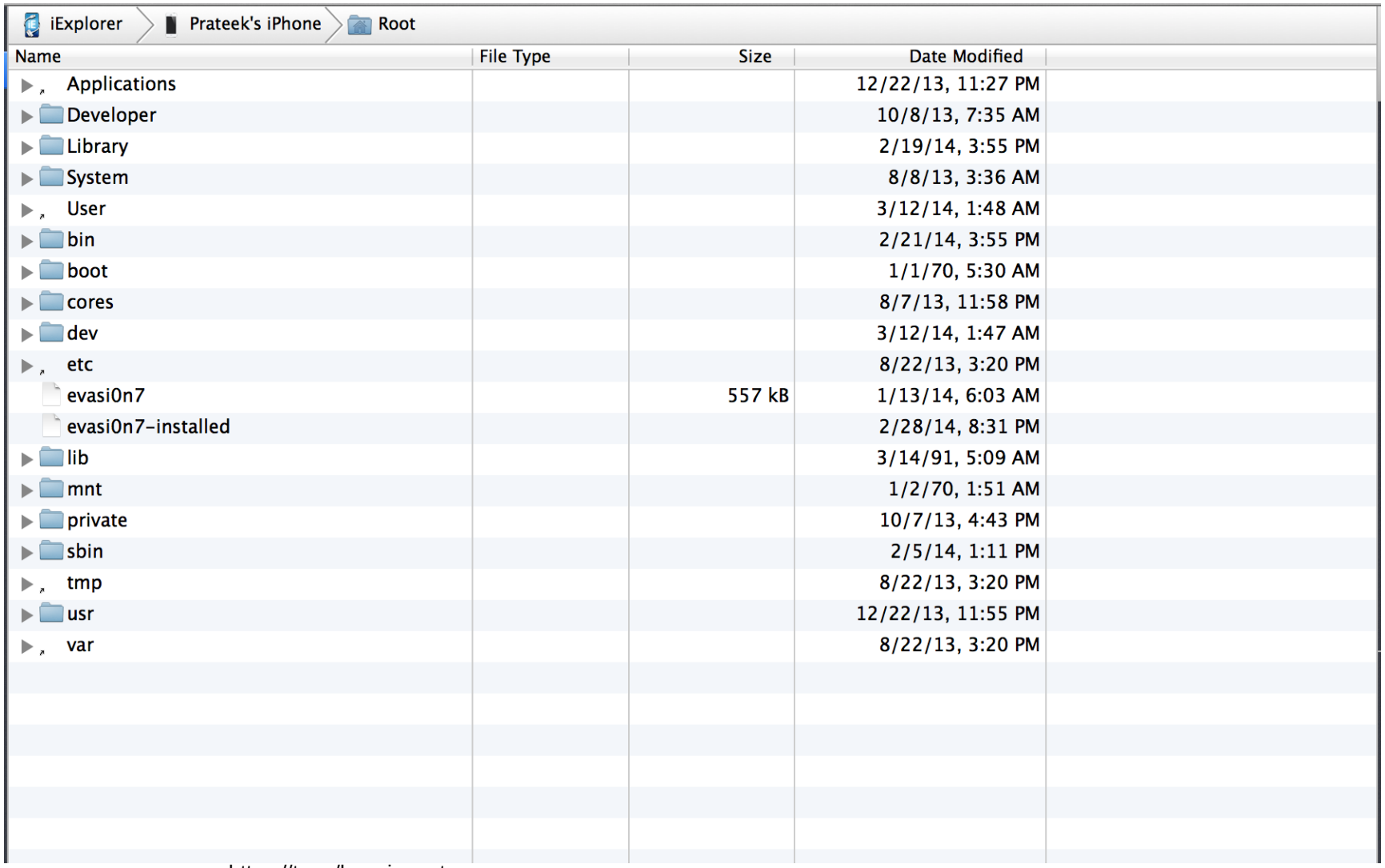
iOS File system

Ref: File system programming guide



Understanding the iOS filesystem

- Download a good file explorer utility like iExplorer or iFunbox



Understanding the iOS filesystem

- **/bin**—Contains essential command-line binaries. Typically, you execute these binaries from command-line scripts.
- **/dev**—Contains essential device files, such as mount points for attached hardware.
- **/etc**—Contains host-specific configuration files.
- **/sbin**—Contains essential system binaries.
- **/tmp**—Contains temporary files created by apps and the system.
- **/usr**—Contains non-essential command-line binaries, libraries, header files, and other data.
- **/var**—Contains log files and other files whose content is variable. (Log files are typically viewed using the Console app.)

Source: https://developer.apple.com/library/archive/documentation/FileManagement/Conceptual/FileSystemProgrammingGuide/FileSystemOverview/FileSystemOverview.html#//apple_ref/doc/uid/TP40010672-CH2-SW

iOS partitions

```
iPhone:/iOS-Device-Tools root# mount
/dev/disk0s1s1 on / (apfs, local, nosuid, journaled, noatime)
devfs on /dev (devfs, local, nosuid, nobrowse)
/dev/disk0s1s3 on /private/xarts (apfs, local, nodev, nosuid, journaled, noatime, nobrows
e)
/dev/disk0s1s2 on /private/var (apfs, local, nodev, nosuid, journaled, noatime, protect)
/dev/disk0s1s4 on /private/var/wireless/baseband_data (apfs, local, nodev, nosuid, journa
led, noatime, nobrowse)
/dev/disk0s1s5 on /private/var/MobileSoftwareUpdate (apfs, local, nodev, nosuid, journale
d, noatime, nobrowse)
iPhone:/iOS-Device-Tools root#
```

Why is the root partition not read only ?

Important - App Bundle vs Container

- **APP BUNDLE** - Contains the App binaries and all the resources required to run the app. Folder name has the **.app** extension
- **APP CONTAINER -** Contains all the **data** created by the app. We will check these folders for local data storage vulnerabilities.

Location of iOS apps in device

- Downloaded applications bundles reside in **/var/Containers/Bundle/Application/<ID>/**. Here you can find the ***.app** file with all the resources included.

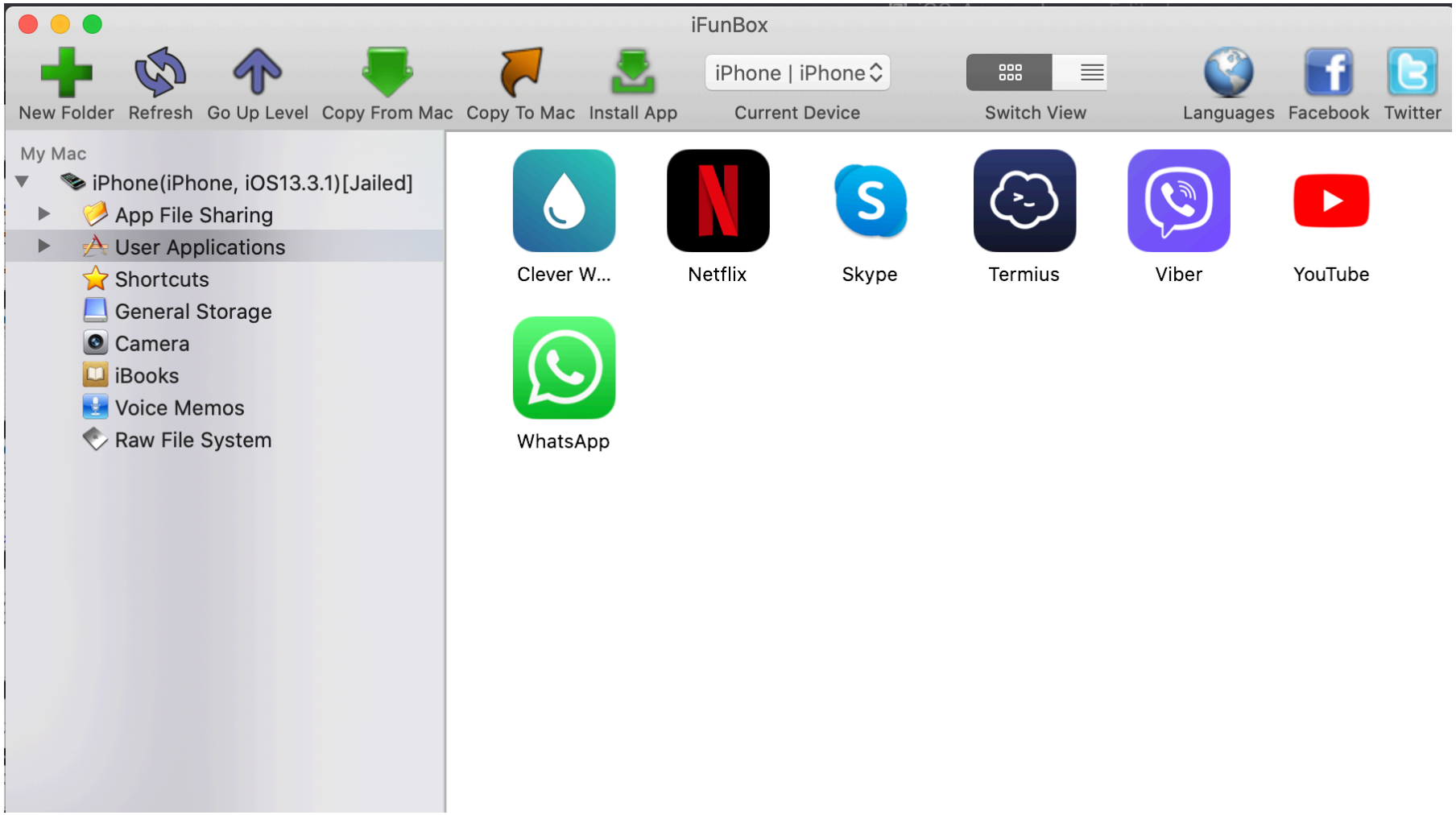
```
sh-5.0# ps aux | grep Damn
mobile           2114   0.0  1.7  4893232  34352   ??  Ss    1:32AM   0:03.11 /var/containers/Bundle/Application/A6A563
6D-01AF-47AE-B77D-99F4D18462B0/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
root             2168   0.0  0.0  4198688    688   ??  R+    2:37AM   0:00.00 grep Damn
sh-5.0#
```

- The **/Documents** and **/Library** data reside in **/var/mobile/Containers/Data/Application/<ID>/**. Here you can found application data, such as *.sqlite dbs or plist files.
- **Bundle vs Data -> Can be a bit Confusing**

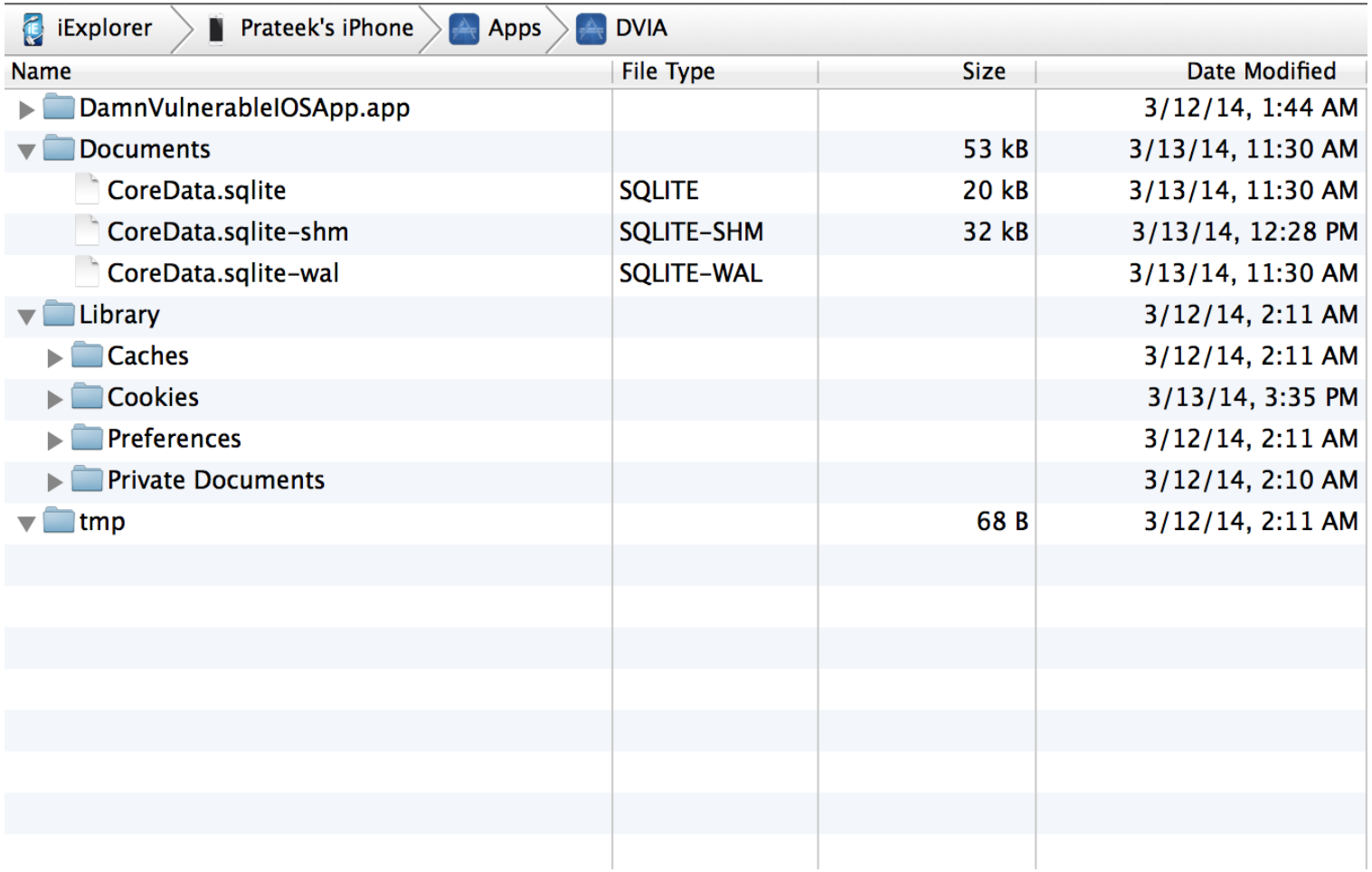
Location of iOS apps in device

- Many applications have different extensions (widgets). Their shared data can be found in **/var/mobile/Containers/Shared/AppGroup/<ID>.**
- Every application runs in its own environment known as the application sandbox, thereby preventing it to access resources from other applications. Such apps are said to be **“containerized”**

Exploring device using iFunBox



This is what a typical application directory looks like



This is what a typical shared application directory looks like

Containers			24.11.14, 11:09
Bundle			24.11.14, 11:09
Data			24.11.14, 11:09
Shared			24.11.14, 11:09
AppGroup			24.04.15, 9:17
0F0299CF-4687-4CA3-AB8D-...		185 B	16.03.15, 17:57
25AFE9E5-7475-4893-8AA8-B...			25.11.14, 12:19
Library			25.11.14, 12:19
appData			31.01.15, 9:58
CMAppData.sqlite	SQLITE	4.5 MB	31.01.15, 11:18
CMAppData.sqlite-shm	SQLITE-SHM	32 kB	31.01.15, 0:10
CMAppData.sqlite-wal	SQLITE-WAL	1.5 MB	31.01.15, 11:18
CMAppFTSData.sqlite	SQLITE	12.4 MB	31.01.15, 0:10
CMAppFTSData.sqlite-shm	SQLITE-SHM	32 kB	31.01.15, 19:05
CMAppFTSData.sqlite-wal	SQLITE-WAL	1.7 MB	31.01.15, 10:07
CMUserPreference.sqlite	SQLITE	24 kB	27.01.15, 19:27
CMUserPreference.sqlite-...	SQLITE-SHM	32 kB	31.01.15, 0:10
CMUserPreference.sqlite-wal	SQLITE-WAL	24 kB	27.01.15, 19:27
CloudMagic1.log	LOG	204 kB	31.01.15, 9:58
CloudMagic2.log	LOG	155 kB	31.01.15, 11:18
Payments.log	LOG		25.11.14, 12:20
appCache			27.01.15, 19:32
cardData			27.01.15, 19:28

Task - 5 mins

- Spend a few mins exploring the filesystem of the device
- Using the information from the previous slides, find the location where the data for **DVIA** and **DVIA-v2** are stored
- Hint: DVIA run (**find / -name *com.highaltitudehacks.dvia***)
- Find similar command for **DVIA-v2**

Frida connection check

- On Corellium device, make sure you run the command **frida-server -l 0.0.0.0 &**
- It is likely it's already running
- Connect to the **WiFi-IP** using the command **frida-ps -H <IP address>**


```
mobile@ubuntu:~/Desktop$ frida-ps -H 10.11.0.1
 PID  Name
----  ----------------------------------------------
2040  Settings
1987  Spotlight
1984  ACCHWComponentAuthService
2035  AppPredictionWidget
2038  AppSSODaemon
  70  AppleCredentialManagerDaemon
 102  BlueTool
 848  CAReportingService
 835  CMFSyncAgent
2020  CacheDeleteAppContainerCaches
 807  CallHistorySyncHelper
  99  CloudKeychainProxy
  78  CommCenter
 116  CommCenterMobileHelper
 819  ContextService
```

Decrypting iOS apps

- Apps downloaded from the App Store are encrypted by default
- Need to dump the apps before they can be reverse engineered
- Frida-ios-dump: https://github.com/AloneMonkey/frida-ios-dump
- Copy the working version from your **Day2** Course material and move to it to your Ubuntu-VM

Frida-ios-dump

- Change the config file **dump.py** for Corellium and put the **Wifi-IP** for your device

```
import paramiko
from paramiko import SSHClient
from scp import SCPClient
from tqdm import tqdm
import traceback

IS_PY2 = sys.version_info[0] < 3
if IS_PY2:
    reload(sys)
    sys.setdefaultencoding('utf8')

script_dir = os.path.dirname(os.path.realpath(__fil

DUMP_JS = os.path.join(script_dir, 'dump.js')
```

```
User = 'root'
Password = 'alpine'
Host = '10.11.0.1'
Port = 22
KeyFileName = None
```

```
TEMP_DIR = tempfile.gettempdir()
PAYLOAD_DIR = 'Payload'
PAYLOAD_PATH = os.path.join(TEMP_DIR, PAYLOAD_DIR)
file_dict = {}

finished = threading.Event()
```

Frida-ios-dump

- Dump the app using **dump.py** script

```
mobile@ubuntu:~/Desktop/frida-ios-dump$ python3 dump.py -R 10.11.0.1 "DVIA"
Connect to remote device: 10.11.0.1 (id=socket@10.11.0.1)
Start the target app DVIA
Dumping DVIA to /tmp
start dump /var/containers/Bundle/Application/69811C66-35F0-47A0-8EB0-ABE41791CBC5/D
amnVulnerableIOSApp.app/DamnVulnerableIOSApp
DamnVulnerableIOSApp.fid: 100%|                                  | 2.19M/2.19M [00:0
7<00:00, 289kB/s]
57x57.png: 6.44MB [01:16, 88.2kB/s]

0.00B [00:00, ?B/s]Generating "DVIA.ipa"
0.00B [00:00, ?B/s]
mobile@ubuntu:~/Desktop/frida-ios-dump$
```

Task - 10 mins Dump decrypted IPA

- Copy the **frida-ios-dump** folder provided in **Day2 Material** and replace it with the one in your UbuntuVM under Desktop
- Connect your UbuntuVM to the VPN
- On your UbuntuVM, go to the folder **Desktop/frida-ios-dump**
- Edit the **dump.py** file to modify the username and password (most likely it's already setup)
- SSH to the corellium device and run the command **frida-server -l 0.0.0.0 &** to start the frida server (most likely it's already setup)
- Dump the decrypted **DVIA** app from the device by running the command **python3 dump.py -R 10.11.0.1 "DVIA"**
- Now try and dump **DVIA-v2**

Solution from Mac for YouTube - USB Device

```
rateekg147@Prateek frida-ios-dump % python3 dump.py YouTube -o YouTube.ipa
tart the target app YouTube
umping YouTube to /var/folders/_v/l9x_tq4n6c1_jxvc3v71rnn80000gn/T
frida-ios-dump]: Load widevine_cdm_secured_ios.framework success.
frida-ios-dump]: Module_Framework.framework has been loaded.
tart dump /private/var/containers/Bundle/Application/D72091B1-D009-4EEC-8E61-EE
pp/YouTube
.00B [00:00, ?B/s]Traceback (most recent call last):
 File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-pac
, line 383, in _on_message
   callback(message, data)
 File "dump.py", line 120, in on_message
   scp.get(scp_from, scp_to)
 File "/Library/Frameworks/Python.framework/Versions/3.8/lib/python3.8/site-pac
238, in get
```

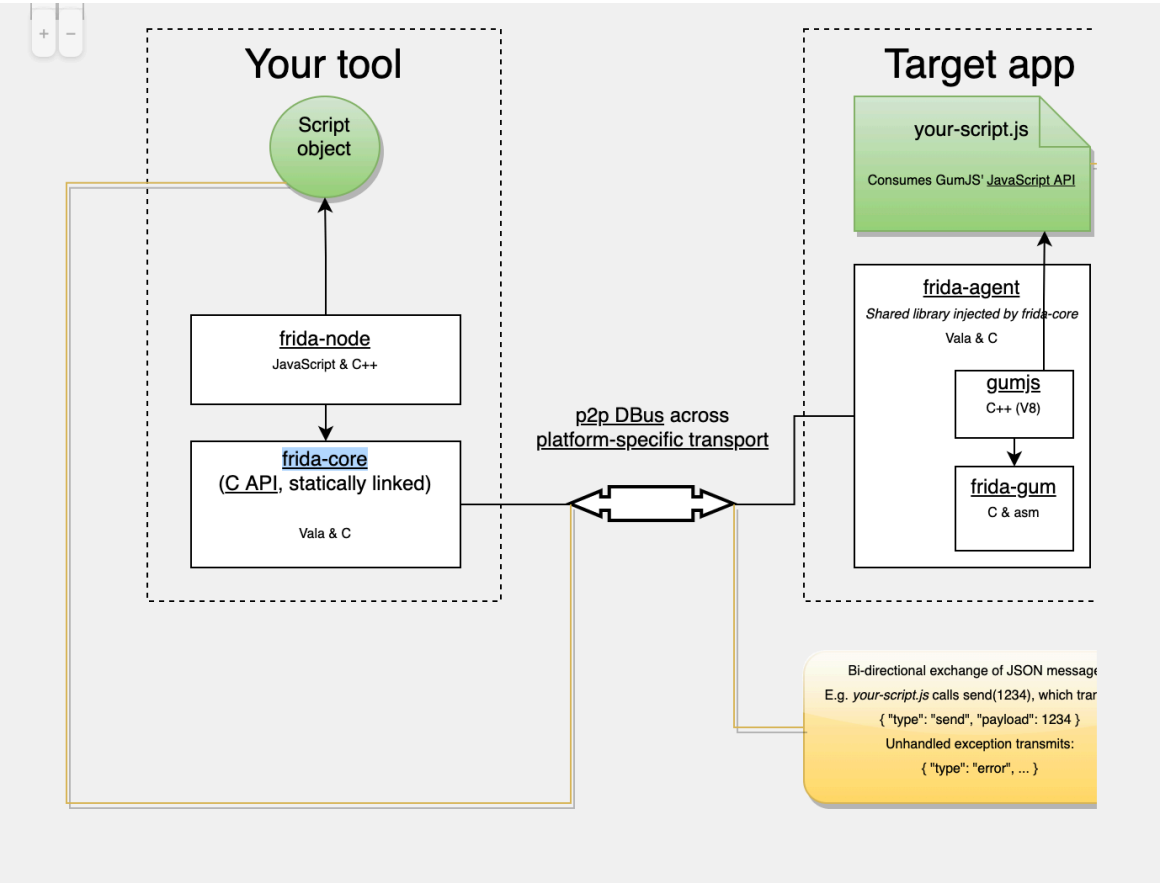
```
  File "/Library/Frameworks/Python.framework/Versions/
 388, in _recv_all
    raise SCPException(asunicode(msg[1:]))
scp.SCPException: scp: /private/var/containers/Bundle/
8C0/YouTube.app: No such file or directory
0.00B [00:00, ?B/s]
0.00B [00:00, ?B/s]Generating "YouTube.ipa"
'app'
0.00B [00:00, ?B/s]
prateekg147@Prateek frida-ios-dump %
```

Frida - Introduction

- Tool for reverse engineering and dynamic code instrumentation
- Works by injecting Quick JS (https://bellard.org/quickjs/) into running processes
- Useful for application information gathering, dynamic manipulation
- Core written in C, JS gets executed with full access to memory
- Can hook functions and even call native functions
- Can work in three modes

    1. Injected
    2. Embedded
    3. Preloaded

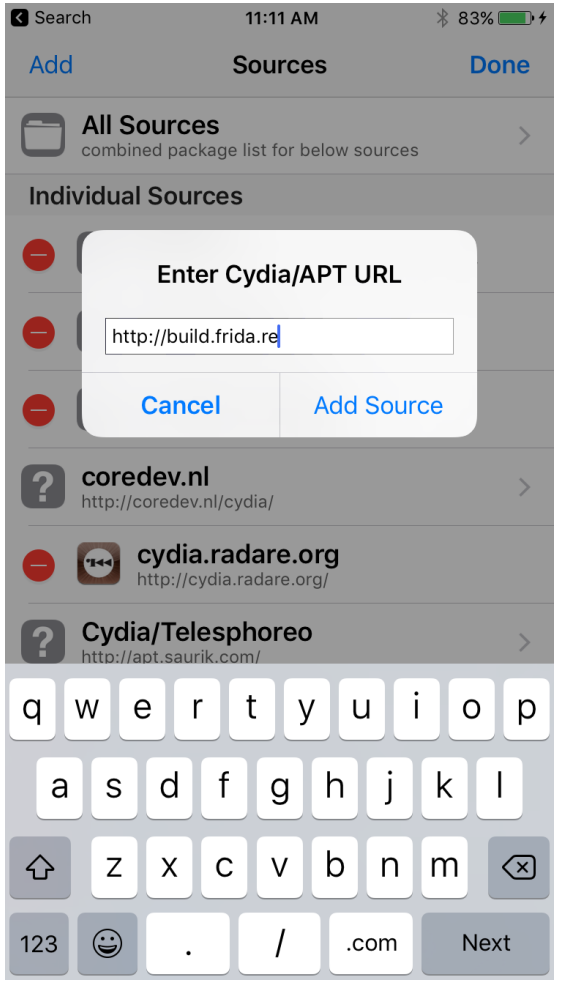
Frida - Introduction



- More details on how Frida works - https://frida.re/docs/hacking/
- Frida documentation - https://frida.re/docs/javascript-api/

Using FRIDA – iDevice Setup

- Launch Cydia.
- Manage -> Sources -> Edit -> Add
- Add https://build.frida.re to Cydia Sources
- Search and Install “Frida”




```
Ptk-iphone:~ root# ps aux | grep frida
root      1686   1.5  0.0   624224    200 s000  R+   11:37AM   0:00.01 grep frid
a
root      1266   0.0  0.1  1088192    812   ??  Ss   Sat01AM   0:00.37 /usr/sbin
/frida-server
Ptk-iphone:~ root#
```

Using FRIDA – Mac Setup

- pip install frida
- If Error -> Use -> sudo -H pip install frida --ignore-installed six
- Run `frida-ps -U` to confirm if Frida works

```
prateek:sudo pip install frida-tools
Collecting frida-tools
  Downloading https://files.pythonhosted.org/packages/70/e6/391
rida-tools-1.0.0.tar.gz
Requirement already satisfied: colorama<1.0.0,>=0.2.7 in /usr/l
Requirement already satisfied: frida<13.0.0,>=12.0.0 in /usr/lo
Requirement already satisfied: prompt-toolkit<2.0.0,>=0.57 in /
.0.15)
Requirement already satisfied: pygments<3.0.0,>=2.0.2 in /usr/l
Requirement already satisfied: wcwidth in /usr/local/lib/pythor
tools) (0.1.7)
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/pyt
da-tools) (1.11.0)
Installing collected packages: frida-tools
  Running setup.py install for frida-tools ... done
Successfully installed frida-tools-1.0.0
prateek:
```

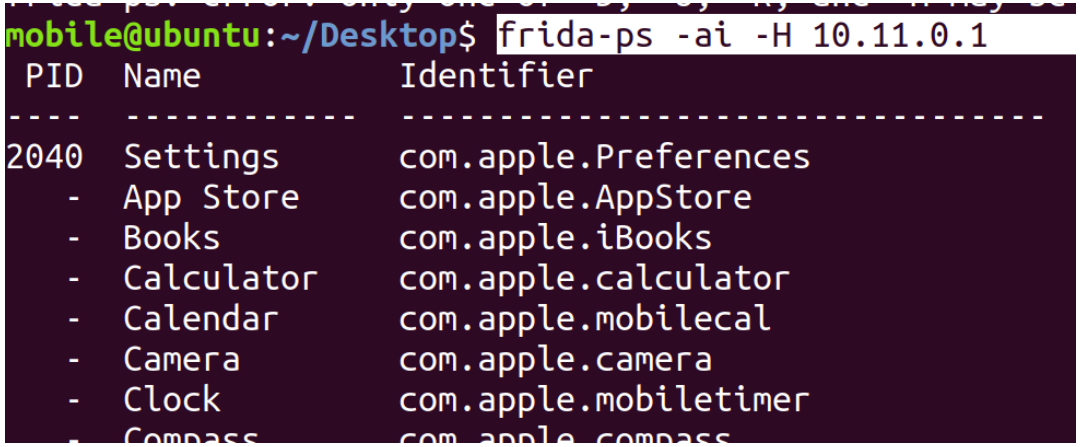
TASK

- Launch the DVIA Application
- Find the process-id of the DVIA application using Frida
- Find the list of all the applications installed on the device using Frida

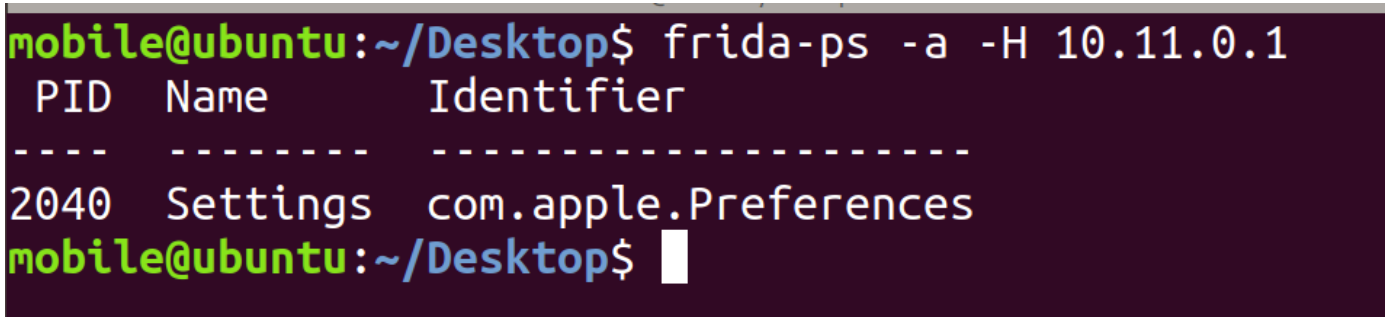
```
mobile@ubuntu:~/Desktop$ frida-ps -H 10.11.0.1  | more
 PID  Name
----  ----------------------------------------------
2040  Settings
1987  Spotlight
1984  ACCHWComponentAuthService
2035  AppPredictionWidget
2038  AppSSODaemon
  70  AppleCredentialManagerDaemon
 102  BlueTool
 848  CAReportingService
 835  CMFSyncAgent
2020  CacheDeleteAppContainerCaches
 807  CallHistorySyncHelper
  99  CloudKeychainProxy
  78  CommCenter
 116  CommCenterMobileHelper
 819  ContextService
 856  HeuristicInterpreter
 831  IDSRemoteURLConnectionAgent
 141  IMDPersistenceAgent
2010  LocalStorageFileProvider
2026  MobileBackupCacheDeleteService
```

TASK

- Find the list of all the applications installed on the device using Frida



- Find the list of all the running applications on the device



Tracing methods

```
mobile@ubuntu:~/Desktop$ frida-trace -m "-[BrokenCrypto* *]" DVIA -H 10.11.0.1
Instrumenting functions...
-[BrokenCryptographyVC readArticleTapped:]: Auto-generated handler at "/home/mobile/
Desktop/__handlers__/__BrokenCryptographyVC_readArtic_-7f2d22eb.js"
-[BrokenCryptographyVC initWithNibName:bundle:]: Auto-generated handler at "/home/mo
bile/Desktop/__handlers__/__BrokenCryptographyVC_initWithN_0a574c96.js"
-[BrokenCryptographyVC viewDidLoad]: Auto-generated handler at "/home/mobile/Desktop
/__handlers__/__BrokenCryptographyVC_viewDidLoad_.js"
-[BrokenCryptographyVC didReceiveMemoryWarning]: Auto-generated handler at "/home/mo
bile/Desktop/__handlers__/__BrokenCryptographyVC_didReceiv_-cfce8d7.js"
-[BrokenCryptographyDetailsVC checkUserState]: Auto-generated handler at "/home/mobi
le/Desktop/__handlers__/__BrokenCryptographyDetailsVC_ch_05d63159.js"
-[BrokenCryptographyDetailsVC firstUserView]: Auto-generated handler at "/home/mobil
e/Desktop/__handlers__/__BrokenCryptographyDetailsVC_fi_0051b514.js"
-[BrokenCryptographyDetailsVC returningUserTextField]: Auto-generated handler at "/h
ome/mobile/Desktop/__handlers__/__BrokenCryptographyDetailsVC_re_7155c44d.js"
-[BrokenCryptographyDetailsVC loggedInLabel]: Auto-generated handler at "/home/mobil
```

```
Started tracing 23 functions. Press Ctrl+C to stop.
           /* TID 0x403 */
  4415 ms  -[BrokenCryptographyVC viewDidLoad]
  6092 ms  -[BrokenCryptographyVC readArticleTapped:0x102d20500]
 27350 ms  -[BrokenCryptographyDetailsVC setFirstUserView:0x102d86ce0]
 27350 ms  -[BrokenCryptographyDetailsVC setLoggedInLabel:0x102d86f60]
 27350 ms  -[BrokenCryptographyDetailsVC setPasswordTextField:0x102da6840]
 27350 ms  -[BrokenCryptographyDetailsVC setReturningUserLabel:0x102d610a0]
 27350 ms  -[BrokenCryptographyDetailsVC setReturningUserTextField:0x102d67830
 27350 ms  -[BrokenCryptographyDetailsVC setReturningUserView:0x102d610a0]
 27350 ms  -[BrokenCryptographyDetailsVC viewDidLoad]
 27350 ms     | -[BrokenCryptographyDetailsVC checkUserState]
 31946 ms  -[BrokenCryptographyDetailsVC textFieldShouldReturn:0x102da6840]
 31946 ms     | -[BrokenCryptographyDetailsVC passwordTextField]
 31961 ms     | -[BrokenCryptographyDetailsVC passwordTextField]
 31984 ms     | -[BrokenCryptographyDetailsVC firstUserView]
```

Tracing methods

```
mobile@ubuntu:~/Desktop$ frida-trace -i *Crypto* DVIA -H 10.11.0.1
Instrumenting functions...
MRCreateCryptoError: Auto-generated handler at "/home/mobile/Desktop/__handl
diaRemote/MRCreateCryptoError.js"
_MRCryptoPairingMessageProtobufReadFrom: Auto-generated handler at "/home/mo
ktop/__handlers__/MediaRemote/_MRCryptoPairingMessageProtobufReadFrom.js"
CCCryptorFinal: Auto-generated handler at "/home/mobile/Desktop/__handlers__
onCrypto.dylib/CCCryptorFinal.js"
CCECCryptorComputeSharedSecret: Auto-generated handler at "/home/mobile/Desk
ndlers__/libcommonCrypto.dylib/CCECCryptorComputeSharedSecret.js"
```

```
VPNTunnelCryptoStopDataTraffic: Auto-generated handler at "/h
ndlers__/SystemConfiguration/VPNTunnelCryptoStopDataTraffic.j
VPNTunnelCryptoStartDataTraffic: Auto-generated handler at "/
andlers__/SystemConfiguration/VPNTunnelCryptoStartDataTraffic
VPNTunnelCryptoEnable: Auto-generated handler at "/home/mobil
SystemConfiguration/VPNTunnelCryptoEnable.js"
Started tracing 147 functions. Press Ctrl+C to stop.
Show Applications
```

Tracing methods

```
^Cmobile@ubuntu:~/Desktop$ frida-trace -I "*libcommonCrypto*" DVIA -H 10.11.0.1
Instrumenting functions...
CCDHRelease: Auto-generated handler at "/home/mobile/Desktop/__handlers__/libcommo
nCrypto.dylib/CCDHRelease.js"
CCECCryptorExportKey: Loaded handler at "/home/mobile/Desktop/__handlers__/libcomm
onCrypto.dylib/CCECCryptorExportKey.js"
CNEncoderFinal: Auto-generated handler at "/home/mobile/Desktop/__handlers__/libco
mmonCrypto.dylib/CNEncoderFinal.js"
CCDigestBlockSize: Auto-generated handler at "/home/mobile/Desktop/__handlers__/li
bcommonCrypto.dylib/CCDigestBlockSize.js"
CCCryptorGetOutputLength: Loaded handler at "/home/mobile/Desktop/__handlers__/lib
commonCrypto.dylib/CCCryptorGetOutputLength.js"
CCBigNumClear: Auto-generated handler at "/home/mobile/Desktop/__handlers__/libcom
monCrypto.dylib/CCBigNumClear.js"
CCECGetKeyType: Auto-generated handler at "/home/mobile/Desktop/__handlers__/libco
```

```
Started tracing 208 functions. Press Ctrl+C to stop.
           /* TID 0xa07 */
 21608 ms  CC_MD5()
 21608 ms     | CCDigest()
 21608 ms  CNEncode()
 21608 ms     | CNEncoderCreate()
 21608 ms     | CNEncoderUpdate()
 21608 ms     | CNEncoderFinal()
 21609 ms     | CNEncoderRelease()
 21619 ms  CC_MD5()
 21619 ms     | CCDigest()
 21619 ms  CNEncode()
 21619 ms     | CNEncoderCreate()
 21619 ms     | CNEncoderUpdate()
 21619 ms     | CNEncoderFinal()
 21619 ms     | CNEncoderRelease()
 21620 ms  CC_MD5()
 21620 ms     | CCDigest()
 21620 ms  CNEncode()
 21620 ms     | CNEncoderCreate()
 21620 ms     | CNEncoderUpdate()
 21620 ms     | CNEncoderFinal()
 21620 ms     | CNEncoderRelease()
           /* TID 0x1303 */
 22919 ms  CCHmac(algorithm=0x0, key=0x170470691, keyLength=0x29, data=0x100541871, da
16e0b2b04)
           /* TID 0xa07 */
 30042 ms  CC_MD5()
 30042 ms     | CCDigest()
 30043 ms  CNEncode()
```

Attach to the application

```
mobile@ubuntu:~/Desktop$ frida DVIA -H 10.11.0.1
     ____
    / _  |   Frida 12.7.11 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/

[Remote::DVIA]-> ObjC.available
true
[Remote::DVIA]-> ObjC.classes
```

Javascript APIs

```
[iPhone::DVIA-v2]-> Process.platform
"darwin"
[iPhone::DVIA-v2]-> Process.arch
"arm64"
[iPhone::DVIA-v2]-> Process.pageSize
16384
[iPhone::DVIA-v2]-> Process.isDebuggerAttached()
false
[iPhone::DVIA-v2]-> Process.getCurrentThreadId()
46099
[iPhone::DVIA-v2]-> Process.codeSigningPolicy
"optional"
[iPhone::DVIA-v2]->
```

Using FRIDA – Dump Class Information

```
for (var classname in ObjC.classes)
{
        if (ObjC.classes.hasOwnProperty(classname))
        {
                console.log(classname);
        }
}
```

Using FRIDA – Dump Class Information

```
[iPhone::DVIA]-> function listClasses()
                 {
                 if (ObjC.available)
                                 {
                                     for (var className in ObjC.classes)
                                     {
                                         if (ObjC.classes.hasOwnProperty(className))
                                         {
                                             console.log(className);
                                         }
                                     }
                                 }
                                 else
                                 {
                                     console.log("Objective-C Runtime is not available!");
                                 }
                 }
undefined
[iPhone::DVIA]-> listClasses()
FigIrisAutoTrimmerMotionSampleExport
NSLeafProxy
Object
__NSGenericDeallocHandler
_NSZombie_
__NSMessageBuilder
NSProxy
WBBadgeLabel
WBTimeLabel
```

Using FRIDA – Dump Methods list

```
for (var classname in ObjC.classes)
{
if (ObjC.classes.hasOwnProperty(classname))
        {
                console.log("ClassName: "+classname);
                var methodslist = ObjC.classes[classname].$ownMethods;
                for (var i = 0; i < methodslist.length; i++)
                        {
                                console.log("\t Method: "+methodslist[i]);
                        }
        }
}
```

Using FRIDA – Dump Methods list

```
⇒  cat list-methods-all.js
console.log("List all available methods:")

function listAllMethods() {
if (ObjC.available)
{
        for (var classname in ObjC.classes)
        {
                if (ObjC.classes.hasOwnProperty(classname))
                {
                        console.log("ClassName: "+classname);
                        var methodslist = ObjC.classes[classname].$ownMethods;
                        for (var i = 0; i < methodslist.length; i++)
                        {
                                console.log("\t Method: "+methodslist[i]);
                        }
                }
        }
}
else
{
        console.log("Something went wrong");
}
}setTimeout(listAllMethods,0);
console.log("End of script");
```

Using FRIDA – Dump Specific Class Methods

- **Example for DVIA App (Make sure app is running on foreground)**

```
var classname = "JailbreakDetectionVC";
console.log("ClassName: "+classname);
var methodslist = ObjC.classes[classname].$ownMethods;
for (var i = 0; i < methodslist.length; i++)
{
        console.log("\t Method: "+methodslist[i]);
}
```

Using FRIDA – Dump Specific Class Methods

```
mobile@ubuntu:~/Desktop$ frida DVIA -H 10.11.0.2
     ____
    / _  |   Frida 12.7.11 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at http://www.frida.re/docs/home/

[Remote::DVIA]->     var classname = "JailbreakDetectionVC";
_ _ console.log("ClassName: "+classname);
_ _ var methodslist = ObjC.classes[classname].$ownMethods;
_ _ for (var i = 0; i < methodslist.length; i++)
_ _ {
_ _ _^Iconsole.log("\t Method: "+methodslist[i]);
_ _ }
ClassName: JailbreakDetectionVC
         Method: - isJailbroken
         Method: - readArticleTapped:
         Method: - jailbreakTest1Tapped:
         Method: - jailbreakTest2Tapped:
         Method: - initWithNibName:bundle:
         Method: - viewDidLoad
         Method: - didReceiveMemoryWarning
undefined
[Remote::DVIA]->
```

Using FRIDA – Dump Specific Class Methods - Function

```
function findMethodsFromClass(classname){
var methodslist = ObjC.classes[classname].$ownMethods;
    for (var i = 0; i < methodslist.length; i++)
    {
        console.log("\t Method: "+methodslist[i]);
    }
}
```

Using FRIDA – Dump Specific class methods

```
                                    }
[iPhone::DVIA]-> function findMethodsFromClass(className)
                                    {
                                    var methods = ObjC.classes[className].$ownMethods;

                                    return methods;
                                    }
undefined
[iPhone::DVIA]-> findMethodsFromClass("BrokenCryptographyDetailsVC")
[
    "- checkUserState",
    "- firstUserView",
    "- returningUserTextField",
    "- loggedInLabel",
    "- returningUserLabel",
    "- encryptAndSaveInKeychainTapped:",
    "- setFirstUserView:",
    "- returningUserView",
    "- setReturningUserView:",
    "- setReturningUserTextField:",
    "- setReturningUserLabel:",
    "- setLoggedInLabel:",
    "- .cxx_destruct",
```

Using FRIDA – Show Return Value

- Applicable for **DVIA-v2** App - Jailbreak Test 2
- Frida-trace command **frida-trace -m "+[JailbreakDet* *]" DVIA-v2 -H 10.11.0.1**

```
var classname = "JailbreakDetection";
    var functionname= "isJailbroken";
    var hook = ObjC.classes[classname][functionname];
    Interceptor.attach(hook.implementation, {
        onLeave: function(returnvalue) {
          console.log("ClassName: " + classname);
          console.log("FunctionName: " + functionname);
          console.log("\tReturn Type: " + typeof returnvalue);
          console.log("\tReturn Value: " + returnvalue);

        }  });
```

Using FRIDA - Show Return Value

```
[iOS Device::DVIA-v2]-> var classname = "JailbreakDetection";
                        var functionname= "isJailbroken";
                        var hook = ObjC.classes[classname][func
tionname];
                        Interceptor.attach(hook.implementation,
 {
                              onLeave: function(returnvalue) {
                                console.log("ClassName: " + cla
ssname);
                                console.log("FunctionName: " +
functionname);
                                console.log("\tReturn Type: " +
 typeof returnvalue);
                                console.log("\tReturn Value: "
+ returnvalue);
                        ^I   newretvalue = ptr("0x0");
                        ^I   returnvalue.replace(newretvalue);
                             }});


{}
[iOS Device::DVIA-v2]-> ClassName: JailbreakDetection
FunctionName: isJailbroken
        Return Type: object
        Return Value: 0x1
```

- Applicable for DVIA-v2 App - Jailbreak Test 2

Task - 20 mins

- Execute all the command shown in the previous slides using Frida

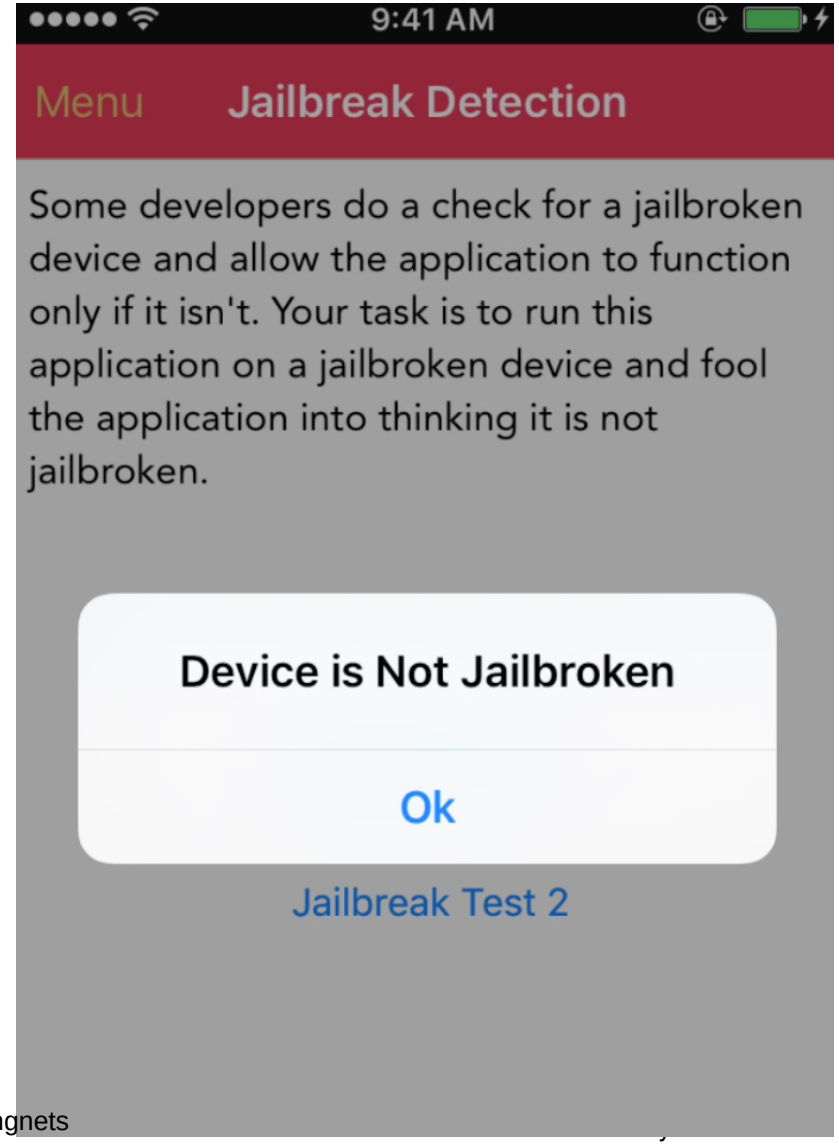
Bypass Root Detection in DVIA-v2 - Jailbreak Test 2

```
newreturnvalue = ptr("0x0");
returnvalue.replace(newreturnvalue );
```

Bypass Root Detection in DVIA-v2 - Jailbreak Test 2

```
[iPhone::DVIA-v2]-> ^Iif (ObjC.available){
                    ^I^Ivar classname = "JailbreakDetection";
                    ^I^Ivar methodname= "isJailbroken";
                    ^I^Ivar hook = ObjC.classes[classname][methodname];
                    ^I^IInterceptor.attach(hook.implementation, {
                    ^I^I  onLeave: function(retvalue) {
                    ^I^I    // args[0] is self
                    ^I^I    // args[1] is selector
                    ^I^I    // args[2] is the return value
                    ^I^I^Inewretvalue = ptr("0x0");
                    ^I^I^Iretvalue.replace(newretvalue);
                    ^I^I    console.log("\nNewReturnValue:"
                    ^I^I        + retvalue + "\"]");
                    ^I^I  }
                    ^I^I});
                    ^I^I}
{}
[iPhone::DVIA-v2]->
NewReturnValue:0x0"]

NewReturnValue:0x0"]
```

Bypass Root Detection - Change Return Value



Bypass Root Detection – Frida Alternatives

- Alternatively make use of below tools based on Frida:
- NEEDLE – https://github.com/mwrlabs/needle
- APPMon - https://github.com/dpnishant/appmon
- Objection - https://github.com/sensepost/objection

Bypass Login Check in DVIA-v2

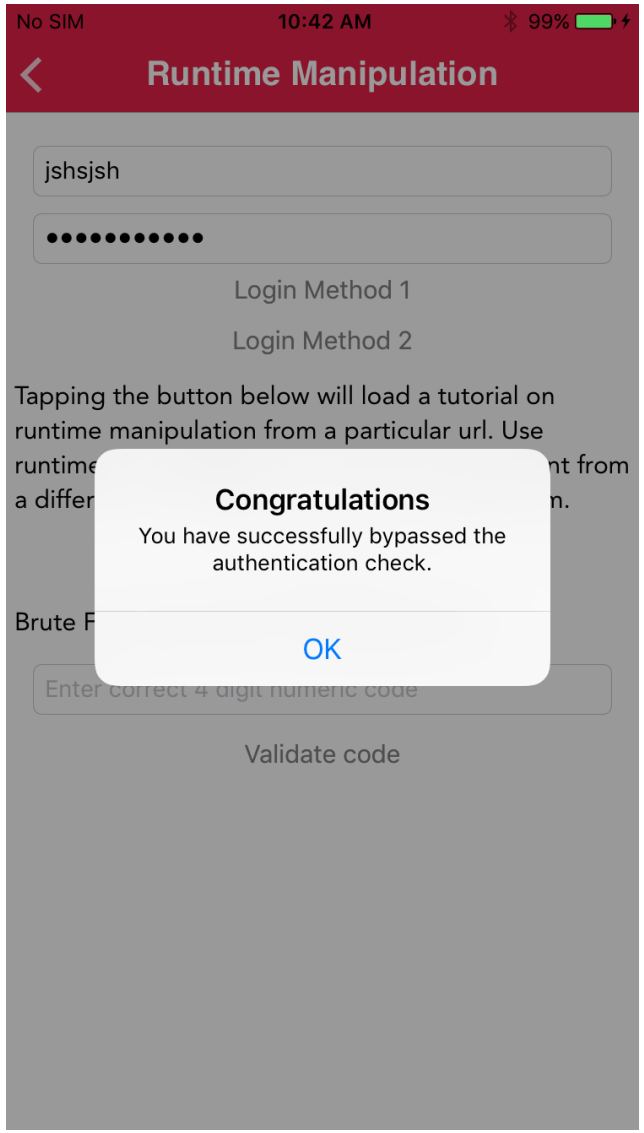
- Using **frida-trace -m "+[LoginValidate* *]" DVIA-v2 -H 10.11.0.1**, We can trace the method being called

```
if (ObjC.available){
var classname = "LoginValidate";
var methodname= "isLoginValidated";
var hook = ObjC.classes[classname][methodname];
Interceptor.attach(hook.implementation,
 { onLeave: function(retvalue)
{
newretvalue = ptr("0x1");
retvalue.replace(newretvalue);
console.log("\nNewReturnValue:" + retvalue + "\"]"); } });
}
```

Bypass Login Check in DVIA-v2

```
[iPhone::DVIA-v2]-> if (ObjC.available){
                    ^Ivar classname = "LoginValidate";
                    ^Ivar methodname= "isLoginValidated";
                    ^Ivar hook = ObjC.classes[classname][methodname];
                    ^IInterceptor.attach(hook.implementation, {
                    ^I  onLeave: function(retvalue) {
                    ^I    // args[0] is self
                    ^I    // args[1] is selector
                    ^I    // args[2] is the return value
                    ^I^Inewretvalue = ptr("0x1");
                    ^I^Iretvalue.replace(newretvalue);
                    ^I    console.log("\nNewReturnValue:"
                    ^I        + retvalue + "\"]");
                    ^I  }
                    ^I});
                    ^I}
                    ^I
{}
[iPhone::DVIA-v2]->
NewReturnValue:0x1"]

NewReturnValue:0x1"]
```

Bypass Login Check in DVIA-v2



Task - 15 mins

- Open **DVIA**, go to **Runtime Manipulation -> Login Method 1**
- Enter any username and password and try to Bypass the login check

Solution

```
if (ObjC.available){
var classname = “RuntimeManipulationDetailsVC";
var methodname= “- isLoginValidated";
var hook = ObjC.classes[classname][methodname]; Interceptor.attach(hook.implementation,
 { onLeave: function(retvalue)
{ // args[0] is self
 // args[1] is selector
// args[2] is the return value
 newretvalue = ptr("0x1");
retvalue.replace(newretvalue);
console.log("\nNewReturnValue:" + retvalue + "\"]"); } });
}
```

Frida- Touch/Face ID Bypass

There are two ways of implementing Touch/Face ID authentication:

- Through Apple's Local Authentication APIs.
- Through access control on the underlying system keychain.

Frida- Touch/Face ID Bypass - evaluatePolicy function

Evaluates the specified policy.

Declaration

```
- (void)evaluatePolicy:(LAPolicy)policy
      localizedReason:(NSString *)localizedReason
                reply:(void (^)(BOOL success, NSError *error))reply;
```

Parameters

policy
The policy to evaluate. For possible values, see `LAPolicy`.

localizedReason
The app-provided reason for requesting authentication, which displays in the authentication dialog presented to the user.

reply
A closure that is executed when policy evaluation finishes. This is evaluated on a private queue internal to the framework in an unspecified threading context. You must not call `canEvaluatePolicy:error:` in this block, because doing so could lead to deadlock.

success
YES if policy evaluation succeeded, otherwise NO.

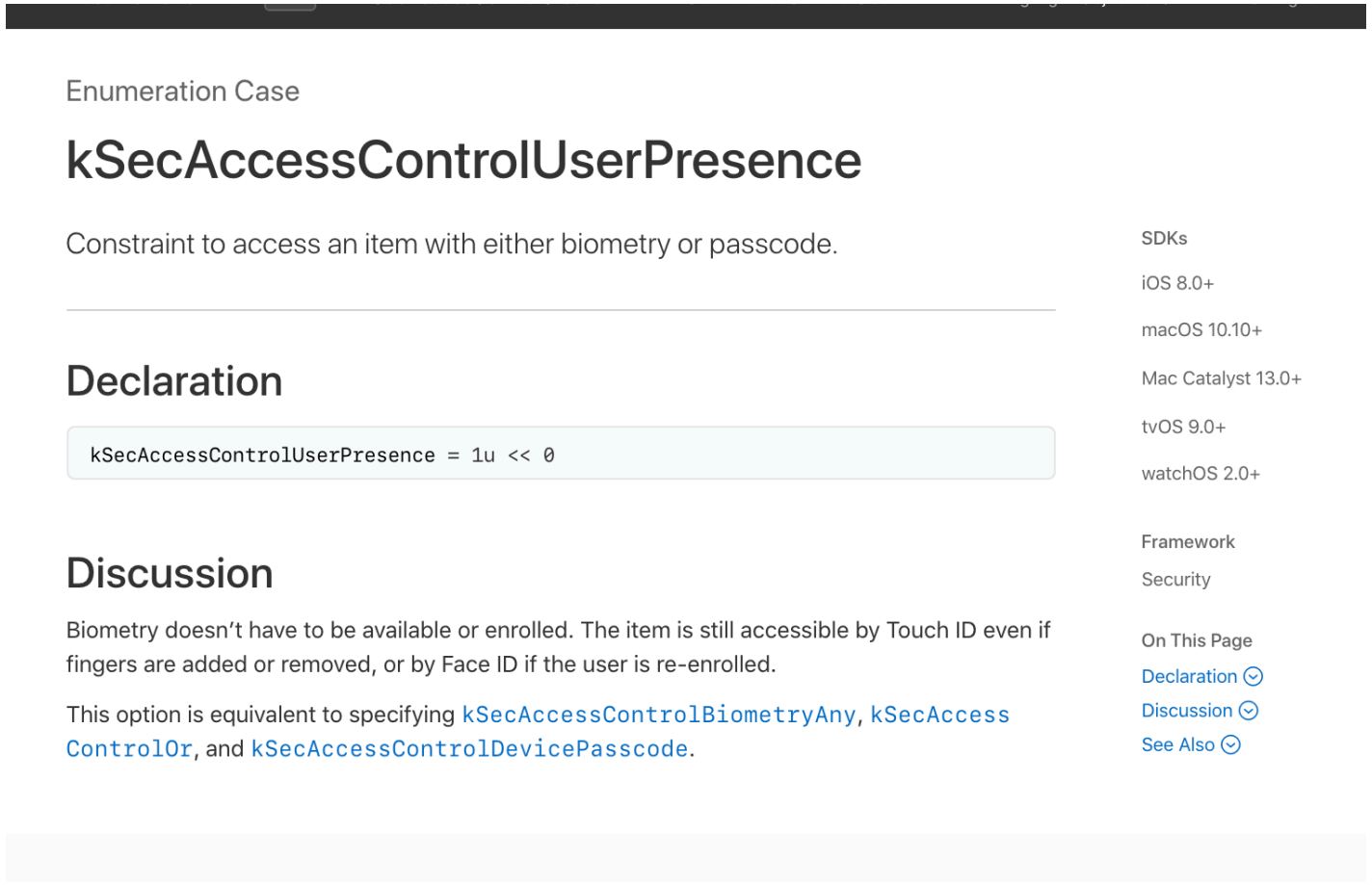
error
`nil` if policy evaluation succeeded, an error object that should be presented to the user otherwise. See `LAError` for possible error codes

- App checks for response and returns a **YES** or **NO**
- Can be hooked to return **YES**

Frida- Touch/Face ID Bypass script

```
if(ObjC.available) {
var hook = ObjC.classes.LAContext["- evaluatePolicy:localizedReason:reply:"];
Interceptor.attach(hook.implementation, {
onEnter: function(args) {
send("Hooking Touch Id..")
var block = new ObjC.Block(args[4]);
const appCallback = block.implementation;
block.implementation = function (error, value) {
const result = appCallback(1, null);
return result;
};
},
});
```

Frida- Touch/Face ID secure implementation



Enumeration Case

kSecAccessControlUserPresence

Constraint to access an item with either biometry or passcode.

Declaration

```
kSecAccessControlUserPresence = 1u << 0
```

Discussion

Biometry doesn't have to be available or enrolled. The item is still accessible by Touch ID even if fingers are added or removed, or by Face ID if the user is re-enrolled.

This option is equivalent to specifying `kSecAccessControlBiometryAny`, `kSecAccessControlOr`, and `kSecAccessControlDevicePasscode`.

SDKs

iOS 8.0+

macOS 10.10+

Mac Catalyst 13.0+

tvOS 9.0+

watchOS 2.0+

Framework

Security

On This Page

Declaration

Discussion

See Also

- Data is stored in the Keychain with a specific access control that requires biometric authentication
- Without successful authentication, data can't be retrieved

Frida- Touch/Face ID secure implementation

- Store the data in the Keychain
- The underlying system keychain is offering an **access control** on the keychain using the **TouchID** authenticator, so when users attempt to retrieve a secure item stored there will have first to be verified using the TouchID successfully.
- Thus, an adversary can not hook the function and tamper the response, because he does not know the stored value that is protected in the keychain.
- It should be noted that this value must be important for the rest of the application authentication or business logic (similar to the user provided PIN).

Frida – Dumping iOS Memory

- Fridump – A universal memory dumper using Frida
- https://github.com/Nightbringer21/fridump
- **python fridump.py -u -s -r DVIA**

Frida – Bypass SSL Pinning Example

```
var servertrusthook = ObjC.classes.OWSHTTPSecurityPolicy["- evaluateServerTrust:forDomain:

 Interceptor.attach(servertrusthook.implementation, {
onLeave: function(returnvalue) {
 returnvalue.replace(ptr(1));
}
});
```

Protecting against Frida and other dylibs

```
inline void scanForInjection() __attribute__((always_inline));

void scanForInjection()
{
   uint32_t count = _dyld_image_count();

   char* evilLibs[] =
   {
      "Substrate", "cycript" , “FridaGadget”, “SSLKillSwitch2”
   };

   for(uint32_t i = 0; i < count; i++)
   {
      const char *dyld = _dyld_get_image_name(i);
      int slength = strlen(dyld);
      int j;
      for(j = slength - 1; j>= 0; --j)
         if(dyld[j] == '/') break;

      char *name = strndup(dyld + ++j, slength - j);

      for(int x=0; x < sizeof(evilLibs) / sizeof(char*); x++)
      {
         if(strstr(name, evilLibs[x]) || strstr(dyld, evilLibs[x]))
                                                fprintf(stderr,"Found injected library matching string: \
                                                      %s", evilLibs[x]);
                                                }

                                                free(name);
                                             }
                        }
//Source: The Mobile Application Hackers’ Handbook
```

Task - 10 mins

- Execute all the previous command using Frida
- Use Frida to trace each file access from within your application (Hint: https://github.com/iddoeldor/frida-snippets#file-access)
- Write a Frida script to take a screenshot of the app
- Check out the other Frida snippets here and try them out https://github.com/iddoeldor/frida-snippets#file-access
- Check out the scripts at https://codeshare.frida.re/

Debuggers

- Allows to hook into the application during runtime and analyse each instruction
- **LLDB** is the debugger used for iOS device since a few years now
- Inspect each method call, set breakpoints, modify value of registers
- Can be used to modify registers during runtime
- Good knowledge of ARM assembly is preferred
- LLDB to GDB command map http://lldb.llvm.org/lldb-gdb.html
- Useful LLDB commands https://www.dropbox.com/s/9sv67e7f2repbpb/lldb-commands-map.png?dl=0

LLDB setup

- Run “**ps aux**” to find the list of running processes on the device
- Run **debugserver** on the device and run **lldb** on your host machine

```
iPhone:~ root# debugserver -a "DVIA-v2" *:1234
debugserver-@(#)PROGRAM:LLDB  PROJECT:lldb-900.3.98
 for arm64.
Attaching to process DVIA-v2...
Listening to port 1234 for a connection from *...
```

```
Prateek:Tools-Device prateekg147$ lldb
(lldb) platform select remote-ios
  Platform: remote-ios
 Connected: no
  SDK Path: "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/8.1 (12B410)"
 SDK Roots: [ 0] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/(null) ((null))"
 SDK Roots: [ 1] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/7.0.4 (11B554a)"
 SDK Roots: [ 2] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/7.1 (11D167)"
 SDK Roots: [ 3] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/8.1 (12B410)"
 SDK Roots: [ 4] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/8.1 (12B411)"
 SDK Roots: [ 5] "/Users/prateekg147/Library/Developer/Xcode/iOS DeviceSupport/8.1.2 (12B440)"
(lldb) process connect connect://192.168.8.32
error: invalid host:port specification: '192.168.8.32'
(lldb) process connect connect://192.168.8.32:1234
```

Reading symbols from binary

```
(lldb) target create --arch arm Twitter
Current executable set to 'Twitter' (armv7).
(lldb)
```

```
(lldb) b -[T1CommerceOfferHowWorksView setHiddenObserver:]
Breakpoint 1: where = Twitter`-[T1CommerceOfferHowWorksView setHiddenObserver:], address = 0x0000b078
(lldb)
```

- Make sure to set the specific architecture for which the binary is compiled

```
(lldb) target create --arch arm64 Twitter
Current executable set to 'Twitter' (arm64).
(lldb) b -[T1CommerceOfferHowWorksView setHiddenObserver:]
Breakpoint 1: where = Twitter`-[T1CommerceOfferHowWorksView setHiddenObserver:], address = 0x0000000100008d20
(lldb)
```

Some basic commands

- **continue** - Resume process (Use **process interrupt** to pause execution)
- **image dump symtab** - Loads the symbols
- **process interrupt** - stops the process
- **register read** - Reads all registers
- **register read <register>** - Reads a particular register
- **register write <register> <value>** - Write value into a particular register
- **image list** - Identifies main executable and loaded libraries
- **b <method>**- Sets breakpoint on a specific method
- **br s -a <Address>** - Sets breakpoint on a specific address
- **image dump sections** - Dump sections (Can be used to calculate ASLR slide)

Lldb on Corellium

```
iPhone:~ root# lldb
(lldb) attach DVIA-v2
Process 831 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = signal SIGSTOP
    frame #0: 0x00000018f9fe5f4 libsystem_kernel.dylib`mach_msg_trap + 8
libsystem_kernel.dylib`mach_msg_trap:
->  0x18f9fe5f4 <+8>: ret

libsystem_kernel.dylib`mach_msg_overwrite_trap:
    0x18f9fe5f8 <+0>: mov    x16, #-0x20
    0x18f9fe5fc <+4>: svc    #0x80
    0x18f9fe600 <+8>: ret

Executable module set to "/var/containers/Bundle/Application/1F90EE26-EE3E-4EAB-B512-AC9ECFD4E311/DVIA-v2.app/DVIA-v2".
Architecture set to: arm64-apple-ios-.
(lldb)
```

Finding ASLR slide

Slid address of TEXT section

```
Architecture set to: arm64-apple-ios-.
(lldb)
(lldb) image dump sections DVIA-v2
Sections for '/var/containers/Bundle/Application/1F90EE26-EE3E-4EAB-B512-AC9ECFD4E311/DVIA-v2.app/D
VIA-v2' (arm64):
  SectID     Type             Load Address                             Perm File Off.  File Size  F
lags      Section Name
  ---------- ---------------- ---------------------------------------  ---- ---------- ---------- -
--------- --------------------------
  0x00000100 container        [0x0000000000000000-0x0000000100000000)* ---  0x00000000 0x00000000 0
x00000000 DVIA-v2.__PAGEZERO
  0x00000200 container        [0x00000001043e4000-0x0000000104788000)  r-x  0x00000000 0x003a4000 0
x00000000 DVIA-v2.__TEXT
  0x00000001 code             [0x00000001043e8ab8-0x000000010470c59c)  r-x  0x00004ab8 0x00323ae4 0
x80000400 DVIA-v2.__TEXT.__text
  0x00000002 code             [0x000000010470c59c-0x000000010470e9e4)  r-x  0x0032859c 0x00002448 0
x80000408 DVIA-v2.__TEXT.__stubs
  0x00000003 code             [0x000000010470e9e4-0x0000000104710d00)  r-x  0x0032a9e4 0x0000231c 0
x80000400 DVIA-v2.__TEXT.__stub_helper
```

```
                        ;
                        ; MachO Header
                        ;
                    __mh_execute_header:
0000000100000000        struct __macho_header64 {                       ; DATA XREF=-[YapDatabaseCloudKitTransa
                            0xfeedfacf,                     // mach magic number identifier
                            0x100000c,                      // cpu specifier
                            0x0,                            // machine specifier
                            MH_EXECUTE,                     // type of file
                            65,                             // number of load commands
                            7112,                           // the size of all the load commands
                            MH_NOUNDEFS|MH_DYLDLINK|MH_TWOLEVEL|MH_WEAK_DEFINES|MH_BINDS_TO_WEAK|MH_PIE, // flags
                            0x0                             // reserved
                        }
                        ;
                        ; Load Command 0
                        ;
0000000100000020        struct __macho_segment_command_64 {
                            LC_SEGMENT_64,                  // LC_SEGMENT_64
                            0x48,                           // includes sizeof section_64 structs
```

Unslid address of TEXT section in Binary

Finding ASLR slide

```
prateek:: python
Python 3.8.6 (default, Oct  8 2020, 14:06:32)
[Clang 12.0.0 (clang-1200.0.32.2)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 0x00000001043e4000 - 0x00000001043e4000
0
>>> 0x00000001043e4000 - 0x0000000100000000
71188480
>>> hex(71188480)
'0x43e4000'
>>>
```

- For this run, the slide is **0x43e4000**
- While setting breakpoints for any function, make sure to add the slide to the **unslid** address obtained from the binary

Bypassing JB Detection 1 challenge

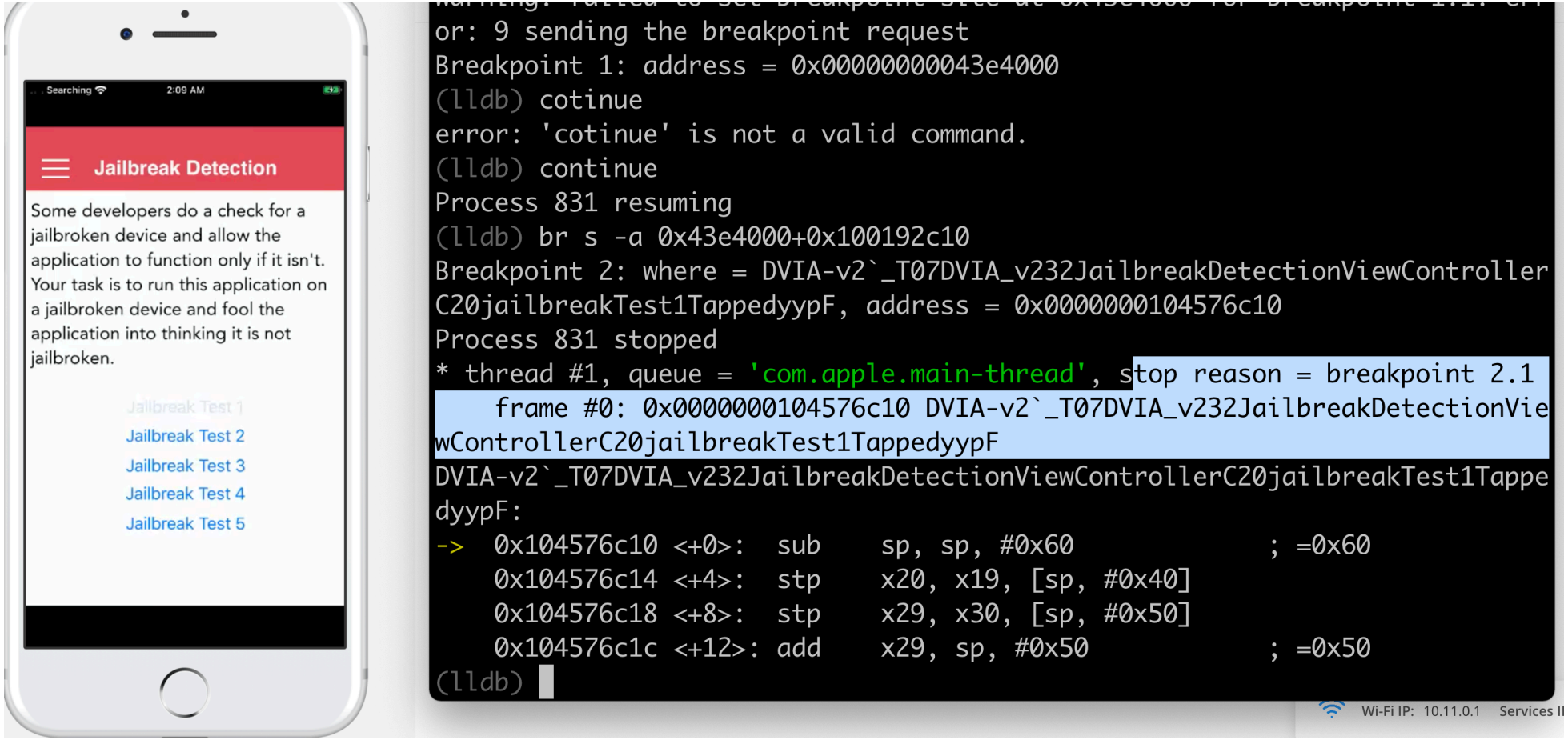
```
                        __T07DVIA_v232JailbreakDetectionViewControllerC20jailbreakTest1TappedyypF:       // DVIA
0000000100192c10            sub         sp, sp, #0x60                                 ; CODE XREF=-[_TtC7DVIA_v232Ja
0000000100192c14            stp         x20, x19, [sp, #0x40]
0000000100192c18            stp         x29, x30, [sp, #0x50]
0000000100192c1c            add         x29, sp, #0x50
0000000100192c20            stur        x0, [x29, var_18]
0000000100192c24            stur        x20, [x29, var_20]
0000000100192c28            str         x20, [sp, #0x50 + var_28]
0000000100192c2c            str         x0, [sp, #0x50 + var_30]
0000000100192c30            bl          __T07DVIA_v213DVIAUtilitiesCMa                ; type metadata accessor for D
0000000100192c34            adrp        x20, #_AVCaptureDeviceTypeBuiltInWideAngleCamera_1003a4000 ; 0x1003a4708@P
0000000100192c38            ldr         x20, [x20, #0x708]                            ; 0x1003a4708@PAGEOFF, _swift_
0000000100192c3c            ldr         x30, [x0, #0x58]
0000000100192c40            ldr         x8, [sp, #0x50 + var_28]
```

- Slid address of the function is **0x43e4000+0x100192c10**
- We are interested in all branch calls from that function

```
(lldb) continue
Process 831 resuming
(lldb) br s -a 0x43e4000+0x100192c10
Breakpoint 2: where = DVIA-v2`_T07DVIA_v232JailbreakDetectionViewControllerC20jailbre
akTest1TappedyypF, address = 0x0000000104576c10
(lldb)
```

Bypassing JB Detection 1 challenge

Click on **Jailbreak Detection 1,** the breakpoint will hit



Bypassing JB Detection 1 challenge

Keep pressing **n** until the next instruction is **blr x9**, then read out the register **x9**

Skip this one and keep pressing **n** until you reach the next **blr x9** instruction

```
    0x104576c68 <+88>: str    w0, [sp, #0xc]
(lldb) n
Process 831 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = instruction step over
    frame #0: 0x0000000104576c60 DVIA-v2`_T07DVIA_v232JailbreakDetectionViewControllerC20jailbre
akTest1TappedyypF + 80
DVIA-v2`_T07DVIA_v232JailbreakDetectionViewControllerC20jailbreakTest1TappedyypF:
->  0x104576c60 <+80>: blr    x9
    0x104576c64 <+84>: ldr    x8, [sp, #0x28]
    0x104576c68 <+88>: str    w0, [sp, #0xc]
    0x104576c6c <+92>: mov    x0, x8
(lldb) register read x9
      x9 = 0x00000001045789a8  DVIA-v2`_T07DVIA_v232JailbreakDetectionViewControllerC12isJailbro
kenSbyF
(lldb) register read w0
      w0 = 0x048301c8
(lldb) register read x0
```

Bypassing JB Detection 1 challenge

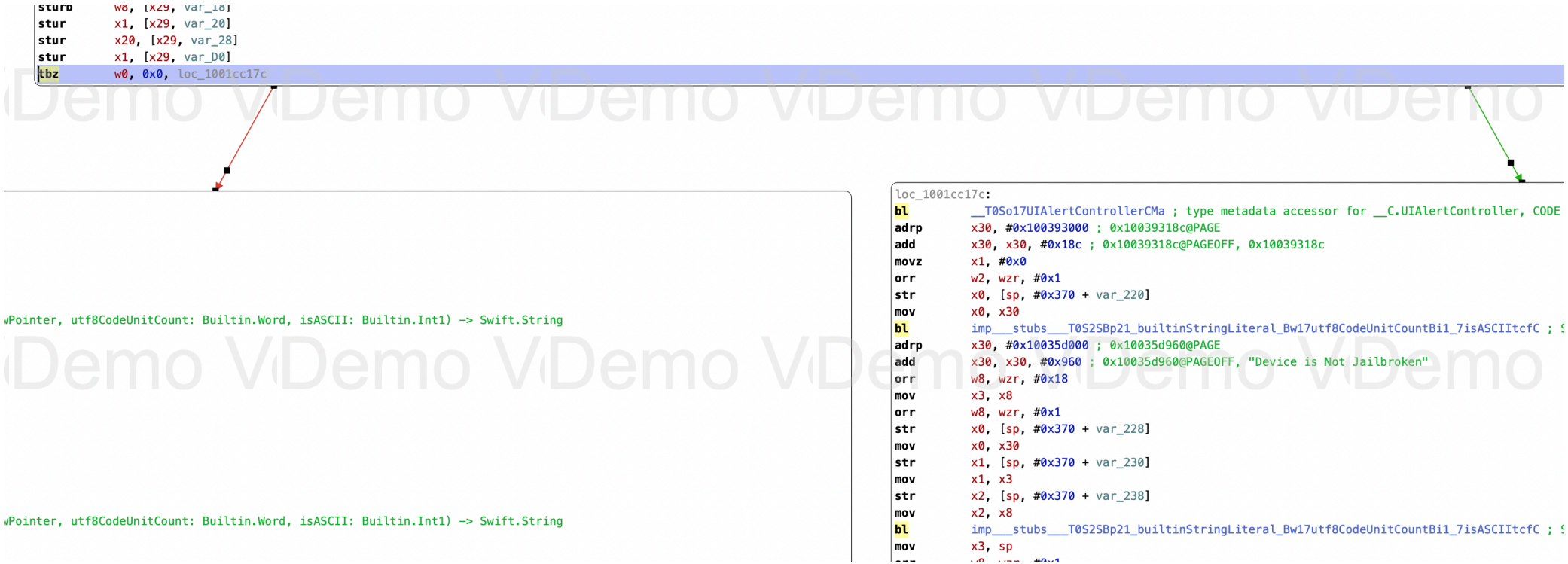
Here is the next **blr x9** instruction, read the value of x9 with the command **register read x9**

Let's reverse this function

```
roy_boxed_opaque_existential_0
(lldb) n
Process 831 stopped
* thread #1, queue = 'com.apple.main-thread', stop reason = instruction
 step over
    frame #0: 0x0000000104576c94 DVIA-v2`_T07DVIA_v232JailbreakDetectio
nViewControllerC20jailbreakTest1TappedyypF + 132
DVIA-v2`_T07DVIA_v232JailbreakDetectionViewControllerC20jailbreakTest1T
appedyypF:
->  0x104576c94 <+132>: blr    x9
    0x104576c98 <+136>: ldr    x0, [sp, #0x20]
    0x104576c9c <+140>: bl     0x10456e1ec               ; __swift_dest
roy_boxed_opaque_existential_0
    0x104576ca0 <+144>: ldp    x29, x30, [sp, #0x50]
(lldb) register read x9
      x9 = 0x00000001045afdac  DVIA-v2`_T07DVIA_v213DVIAUtilitiesC9show
AlertySb28forJailbreakTestIsJailbroken_So16UIViewControllerC04viewL0tFZ
(lldb)
```

Bypassing JB Detection 1 challenge

```
__T07DVIA_v213DVIAUtilitiesC9showAlertySb28forJailbreakTestIsJailbroken_So16UIViewControllerC04viewL0tFZ:        // static
DVIA_v2.DVIAUtilities.showAlert(forJailbreakTestIsJailbroken: Swift.Bool, viewController: __C.UIViewController) -> ()
```



TBZ - Test bit and branch if zero to a label at a PC-relative offset, without affecting the condition flags, and with a hint that this is not a subroutine call or return.

Bypassing JB Detection 1 challenge

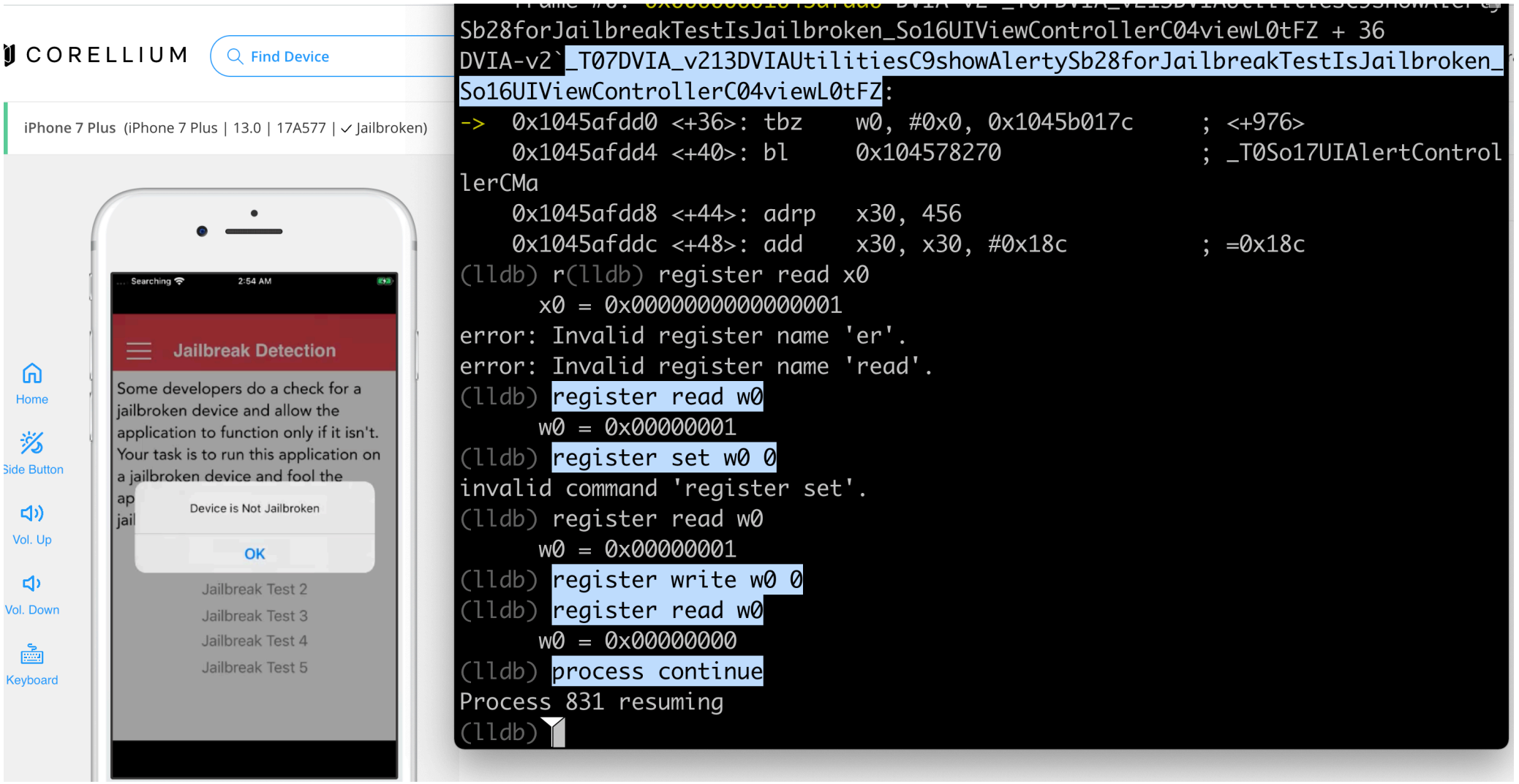
```
28forJailbreakTestIsJailbroken_So16UIViewControllerC0
(lldb) br s -a 0x43e4000+0x1001cbdd0
Breakpoint 5: where = DVIA-v2`_T07DVIA_v213DVIAUtilit
breakTestIsJailbroken_So16UIViewControllerC04viewL0tF
01045afdd0
(lldb) process continue

Process 831 resuming
(lldb)
error: Process is running.  Use 'process interrupt' t
Process 831 stopped
```

```
00000001001cbdcc        stur    x1, [x29, var_D0]
00000001001cbdd0        tbz     w0, 0x0, loc_1001cc17c
00000001001cbdd4        bl      __T0So17UIAlertControllerCMa
00000001001cbdd8        adrp    x30, #0x100393000
00000001001cbddc        add     x30, x30, #0x18c
00000001001cbde0        movz    x1, #0x0
00000001001cbde4        orr     w2, wzr, #0x1
00000001001cbde8        stur    x0, [x29, var_D8]
00000001001cbdec        mov     x0, x30
00000001001cbdf0        bl      imp___stubs___T0S2SBp21_builtinStringLi
00000001001cbdf4        adrp    x30, #0x10035f000
```

Let’s set a breakpoint on the **tbz** instruction

Once the breakpoint hit at that instruction, change the argument **w0** from **1** to **0**

JB Detection 1 challenge - Solved



Task - 25 mins

- Use **lldb** to connect to **DVIA-v2** on **Corellium**
- Bypass **Jailbreak detection challenge 1** in DVIA-v2 using **lldb** as discussed in previous slides
- Recommended reading -
- https://www.citadel.sh/blog/dvia-v2-jailbreak-detection-solutions
- https://hackcatml.tistory.com/47

Protection against debuggers

- Add this C code snippet in your application

```
void detectdebugger()
{
    //Check if app was started by someone else other than launchd
    if (getppid() != 1){
        exit(0);
    }

    //ptrace - deny any other app to attach to the process
    void* handle = dlopen(0, RTLD_GLOBAL | RTLD_NOW);
    ptrace_ptr_t ptrace_ptr = dlsym(handle, "ptrace");
    ptrace_ptr(PT_DENY_ATTACH, 0, 0, 0);
    dlclose(handle);

    //sysctl method
    int name[4];
    struct kinfo_proc info;
    size_t info_size = sizeof(info);
    info.kp_proc.p_flag = 0;
    name[0] = CTL_KERN;
    name[1] = KERN_PROC;
    name[2] = KERN_PROC_PID;
    name[3] = getpid();
    if (sysctl(name, 4, &info, &info_size, NULL, 0) == -1) {
        //Program is being debugged
        exit(0);
    }
}
```

Bypass ?

- How do you bypass this mitigation ?
- Can we do it using Frida ? Is there any specific option in Frida we need to use in order to implement this ?
- Can you use Frida to patch C calls ?

Binary Code Changes in main function (0x1000065d0)

```
                        ; endp

                        ; =============== B E G I N N I N G   O F   P R O C E D U R E ===============

                        ; Variables:
                        ;    saved_fp: 0
                        ;    var_10: -16
                        ;    var_20: -32


                    EntryPoint:
00000001000065d0            stp         x22, x21, [sp, #-0x30]!
00000001000065d4            stp         x20, x19, [sp, #0x10]
00000001000065d8            stp         x29, x30, [sp, #0x20]
00000001000065dc            add         x29, sp, #0x20
00000001000065e0            mov         x19, x1
00000001000065e4            mov         x20, x0
00000001000065e8            bl          imp___stubs__objc_autoreleasePoolPush       ; objc_autoreleasePoolPush
00000001000065ec            mov         x21, x0
00000001000065f0            bl          sub_100006580                               ; sub_100006580
00000001000065f4            nop
00000001000065f8            ldr         x0, =__objc_class_AppDelegate_class         ; __objc_class_AppDelegate_class
00000001000065fc            nop
0000000100006600            ldr         x1, =aClass                                 ; "class",@selector(class)
0000000100006604            bl          imp___stubs__objc_msgSend                   ; objc_msgSend
0000000100006608            bl          imp___stubs__NSStringFromClass              ; NSStringFromClass
000000010000660c            mov         x29, x29
0000000100006610            bl          imp___stubs__objc_retainAutoreleasedReturnValue ; objc_retainAutoreleasedReturnValue
0000000100006614            mov         x22, x0
0000000100006618            mov         x0, x20
000000010000661c            mov         x1, x19
0000000100006620            movz        x2, #0x0
0000000100006624            mov         x3, x22
0000000100006628            bl          imp___stubs__UIApplicationMain              ; UIApplicationMain
000000010000662c            mov         x19, x0
0000000100006630            mov         x0, x22
0000000100006634            bl          imp___stubs__objc_release                   ; objc_release
0000000100006638            mov         x0, x21
000000010000663c            bl          imp___stubs__objc_autoreleasePoolPop        ; objc_autoreleasePoolPop
0000000100006640            mov         x0, x19
0000000100006644            ldp         x29, x30, [sp, #0x20]
0000000100006648            ldp         x20, x19, [sp, #0x10]
000000010000664c            ldp         x22, x21, [sp], #0x30
0000000100006650            ret
                        ; endp
```

Reversing the anti debugging function (0x100006580)

```
; =============== B E G I N N I N G   O F   P R O C E D U R E ===============

; Variables:
;    saved_fp: 0
;    var_10: -16


                sub_100006580:
0000000100006580        stp     x20, x19, [sp, #-0x20]!         ; CODE XREF=EntryPoint+32
0000000100006584        stp     x29, x30, [sp, #0x10]
0000000100006588        add     x29, sp, #0x10
000000010000658c        movz    w1, #0xa
0000000100006590        movz    x0, #0x0                        ; argument "path" for method imp___stubs__dlopen
0000000100006594        bl      imp___stubs__dlopen             ; dlopen
0000000100006598        mov     x19, x0
000000010000659c        adr     x1, #0x100007e47                ; "ptrace"
00000001000065a0        nop
00000001000065a4        bl      imp___stubs__dlsym              ; dlsym
00000001000065a8        mov     x8, x0
00000001000065ac        orr     w0, wzr, #0x1f
00000001000065b0        movz    w1, #0x0
00000001000065b4        movz    w3, #0x0
00000001000065b8        movz    x2, #0x0
00000001000065bc        blr     x8
00000001000065c0        mov     x0, x19
00000001000065c4        ldp     x29, x30, [sp, #0x10]
00000001000065c8        ldp     x20, x19, [sp], #0x20
00000001000065cc        b       imp___stubs__dlclose            ; dlclose
                ; endp
```

Replacing the anti debug function

- Actual address will be always different because of slide, but relative address between functions would be same.
- relative address is (sub - main) -> (0x100006580 - 0x1000065d0) = -0x50
- Function at a particular pointer can be hooked and replaced with Javascript function using **NativeCallback** in frida https://frida.re/docs/javascript-api/#nativecallback

NativeCallback

- `new NativeCallback(func, returnType, argTypes[, abi])`: create a new NativeCallback implemented by the JavaScript function `func`, where `returnType` specifies the return type, and the `argTypes` array specifies the argument types. You may also specify the abi if not system default. See `NativeFunction` for details about supported types and abis. Note that the returned object is also a `NativePointer`, and can thus be passed to `Interceptor#replace`. When using the resulting callback with Interceptor.replace(), `func` will be invoked with `this` bound to an object with some useful properties, just like the one in Interceptor.attach().

Replacing the anti debug function

```
var mainAddr = Module.findExportByName(null, 'main')
var subAddr = mainAddr.sub(0x50);
console.log("subAddr is " + subAddr.toString(16));
var debugfuncAddr = ptr(subAddr);
Interceptor.replace(debugfuncAddr, new NativeCallback(function (argc, argv){
console.log("Bypassing anti debug function!!");
return;
}, 'void', ['int', 'pointer']));
```

Frida - Bypass ptrace (recommended reading)

https://la0s.github.io/2019/03/07/anti_ptrace/

```
var f = Module.findExportByName('dyld',"dlsym");
Interceptor.attach(f, {
    onEnter: function (args) {
        this.is_common_path = false;
        var arg = Memory.readUtf8String(args[1]);
            if (arg.indexOf("ptrace") > -1) {
                this.is_common_path = true;
                //return -1;
            }
    },
    onLeave: function (retval) {
               if (this.is_common_path){
                   console.log("ptrace str: " + retval); //打印返回的ptrace地址：0x18bad1078
                   //retval.replace();
               }
            }
});


var ptracePtr = ptr("0x18bad1078"); //ptrace函数地址
var OriginPtrace = new NativeFunction(ptracePtr, 'void', ["pointer", "pointer", "pointer", "point
Interceptor.replace(ptracePtr, new NativeCallback(function(arg1, arg2, arg3, arg4) {
    if (arg1 == 31) {
        console.log("Hook ptrace Bypass!!!");
        return 0;
    }
    else {
        OriginPtrace(arg1, arg2, arg3, arg4);
    }
},'int', ['int', 'pointer']));
```

Frida - Manipulating registers

- Using Frida also, you can also hook a particular address and modify the value of registers

```
 Interceptor.attach(functionAddress, {
  onEnter: function(args) {
        if(this.context.x0 == 0x01){
          this.context.x0=0x00
          console.log("Register x0 changed from 1 to 0");
      }
  },
});
```

Task - 15 mins

- Using your knowledge of Frida and ARM - Solve **Jailbreak Test 1** challenge in **DVIA-v2**
- **Hint: Look for all branch calls**

Jailbreak Test 1 - DVIA-v2

```
                        __T07DVIA_v232JailbreakDetectionViewControllerC20jailbreakTest1TappedyypF:       // DVIA_v2.JailbreakDetectionViewController.jailbreakTest1Tapped(
0000000100192c10            sub         sp, sp, #0x60                                 ; CODE XREF=-[_TtC7DVIA_v232JailbreakDetectionViewController jailbreakTest1Tapped:]+72
0000000100192c14            stp         x20, x19, [sp, #0x40]
0000000100192c18            stp         x29, x30, [sp, #0x50]
0000000100192c1c            add         x29, sp, #0x50
0000000100192c20            stur        x0, [x29, var_18]
0000000100192c24            stur        x20, [x29, var_20]
0000000100192c28            str         x20, [sp, #0x50 + var_28]
0000000100192c2c            str         x0, [sp, #0x50 + var_30]
0000000100192c30            bl          __T07DVIA_v213DVIAUtilitiesCMa                ; type metadata accessor for DVIA_v2.DVIAUtilities
0000000100192c34            adrp        x20, #_AVCaptureDeviceTypeBuiltInWideAngleCamera_1003a4000 ; 0x1003a4708@PAGE
0000000100192c38            ldr         x20, [x20, #0x708]                            ; 0x1003a4708@PAGEOFF, _swift_isaMask_1003a4708,_swift_isaMask
0000000100192c3c            ldr         x30, [x0, #0x58]
0000000100192c40            ldr         x8, [sp, #0x50 + var_28]
0000000100192c44            ldr         x9, [x8]
0000000100192c48            ldr         x20, [x20]                                    ; _swift_isaMask
0000000100192c4c            and         x9, x9, x20
0000000100192c50            ldr         x9, [x9, #0x80]
0000000100192c54            mov         x20, x8
0000000100192c58            str         x0, [sp, #0x50 + var_38]
0000000100192c5c            str         x30, [sp, #0x50 + var_40]
0000000100192c60            blr         x9
0000000100192c64            ldr         x8, [sp, #0x50 + var_28]
0000000100192c68            str         w0, [sp, #0x50 + var_44]
0000000100192c6c            mov         x0, x8
0000000100192c70            bl          imp___stubs__objc_retain                      ; objc_retain
0000000100192c74            ldr         x8, [sp, #0x50 + var_28]
0000000100192c78            ldr         x9, [sp, #0x50 + var_40]
0000000100192c7c            ldr         w10, [sp, #0x50 + var_44]
0000000100192c80            and         w11, w10, #0x1
0000000100192c84            str         x0, [sp, #0x50 + var_50]
0000000100192c88            mov         x0, x11
0000000100192c8c            mov         x1, x8
0000000100192c90            ldr         x20, [sp, #0x50 + var_38]
0000000100192c94            blr         x9
0000000100192c98            ldr         x0, [sp, #0x50 + var_30]
0000000100192c9c            bl          ___swift_destroy_boxed_opaque_existential_0 ; ___swift_destroy_boxed_opaque_existential_0
0000000100192ca0            ldp         x29, x30, [sp, #0x50]
0000000100192ca4            ldp         x20, x19, [sp, #0x40]
0000000100192ca8            add         sp, sp, #0x60
0000000100192cac            ret
                          ; endp
```

Solution

```
var targetModule = "DVIA-v2";
var addr = ptr(0x192c94);
var moduleBase = Module.getBaseAddress(targetModule);
var targetAddress = moduleBase.add(addr);
  Interceptor.attach(targetAddress, {
      onEnter: function(args) {
          if(this.context.x0 == 0x01){
             this.context.x0=0x00
             console.log("Modified x0 Register value from 1 to 0");
        }
      },
  });
```

Solution

Task - 30 mins

- Bypass the rest of the jailbreak detection challenges in DVIA-v2
- Solution: https://philkeeble.com/ios/reverse-engineering/iOS-Bypass-Jailbreak/

Insecure Local Data Storage

- It is important to securely store local data on the device.
- In the event of a lost or compromised device, the data can be accessed by an attacker.

There are many ways of storing data locally on an iOS device.

Some of these techniques are:

- Plist
- NSUserDefaults
- CoreData (Sqlite)
- Keychain

Plist

- Data stored in plist files is stored unencrypted in the application sandbox.
- Property lists, data is stored as key/value pair.
- Generally used to store user settings, or information about the application.
- Most often, developers make the mistake of storing confidential data in Plist files.

Plist

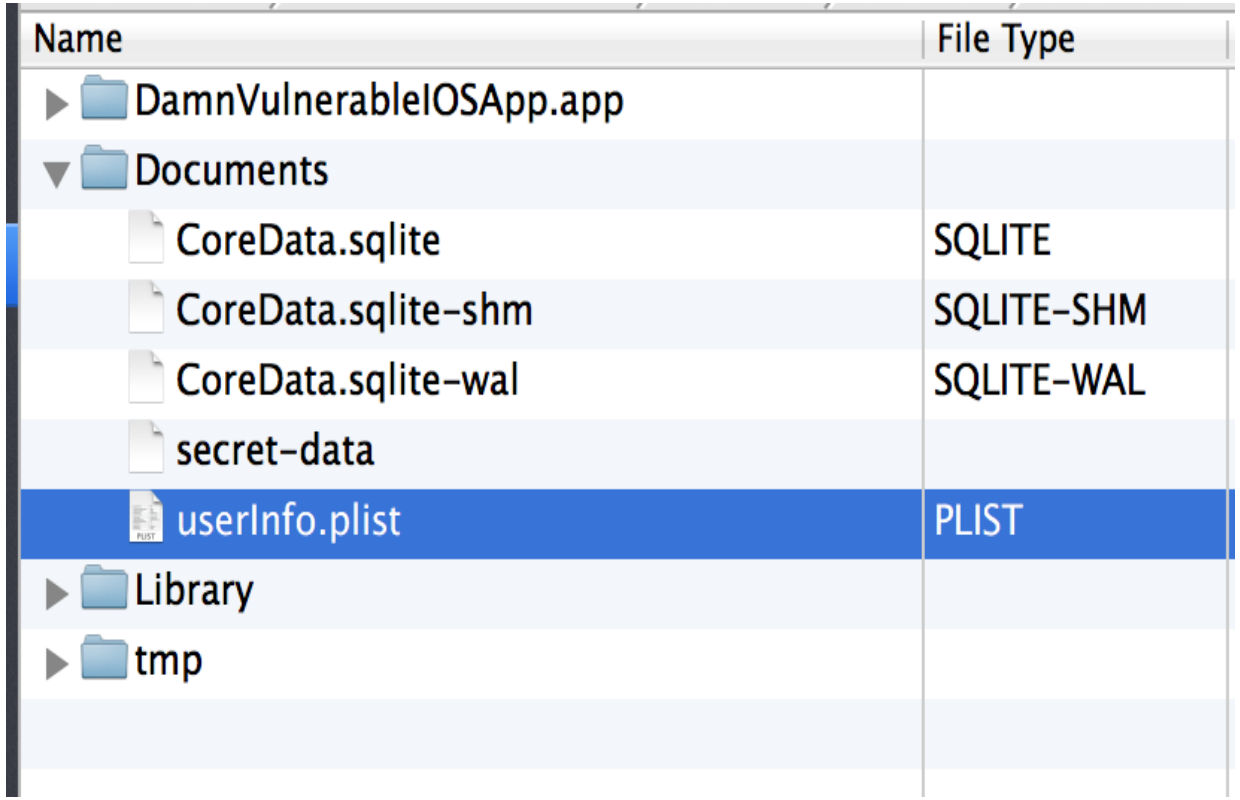
- Sample code for storing data in plist files.

```
NSArray *paths = NSSearchPathForDirectoriesInDomains(NSDocumentDirectory,NSUserDomainMask,YES);
NSString *documentsDirectory = [paths objectAtIndex:0];
NSString *filePath = [documentsDirectory stringByAppendingString:@"/userInfo.plist"];
NSMutableDictionary* plist = [[NSMutableDictionary alloc] init];
[plist setValue:self.usernameTextField.text forKey:@"username"];
[plist setValue:self.passwordTextField.text forKey:@"password"];
[plist writeToFile:filePath atomically:YES];
[DamnVulnerableAppUtilities showAlertWithMessage:@"Data saved in Plist"];
```

- Location in file system is **/private/var/mobile/Containers/Data/Application/<APP ID>/Documents/userInfo.plist**

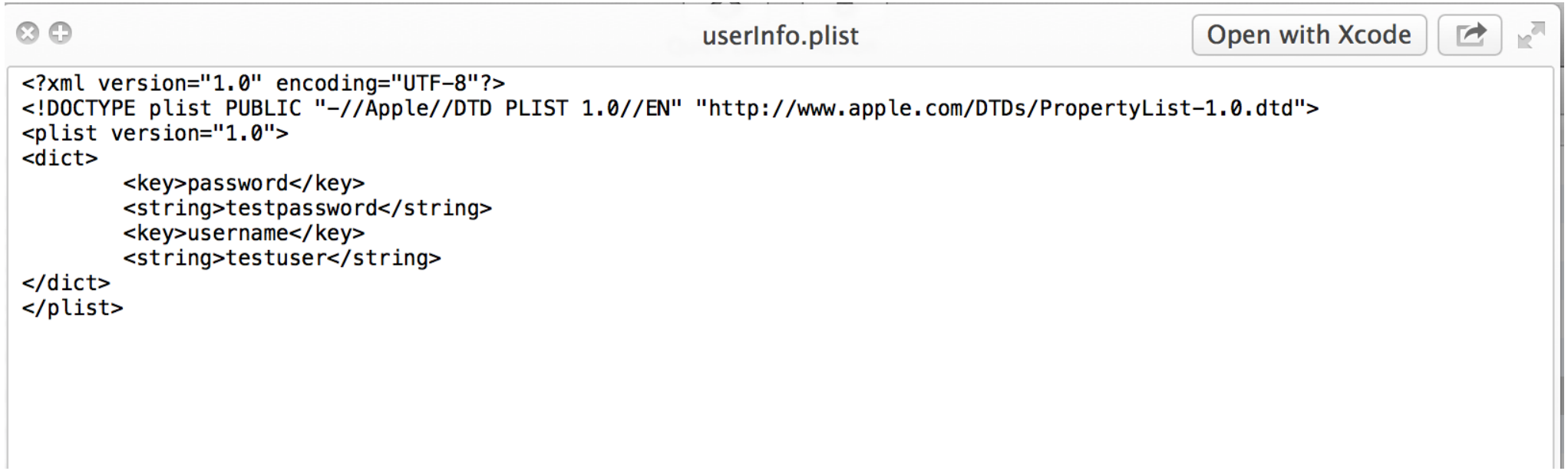
Plist

- These files can be easily found using any simple file explorer utility like iExplorer in the application folder.



Plist

- On inspecting these files, you can find the information being saved in the plist file.



Plist

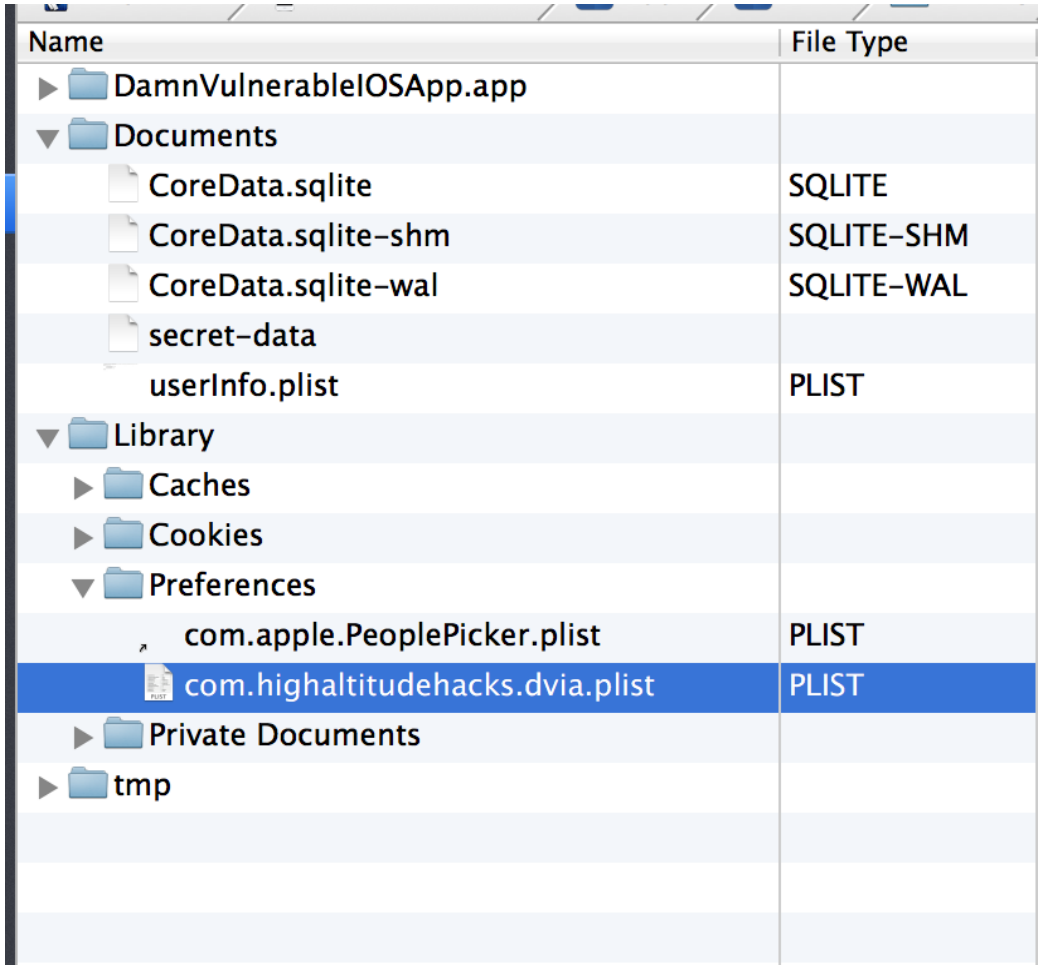
- Do not use plist files to store confidential information like username/ passwords.
- Do not store session ID’s , important properties etc in a plist file.
- Plist files should only be used to store information that is not important, for e.g, a list of image names, the last launch date of the application etc.

NSUserDefaults

- Used for storing properties, objects that can persist even after an application restart.
- Information is saved unencrypted inside the application sandbox in a plist file with the name [BUNDLE_ID].plist inside the folder Library -> preferences .
- Developers make a common mistake of storing critical data using NSUserDefaults.

NSUserDefaults

- All the information stored using NSUserDefaults can be found inside the file [BUNDLE_ID].plist inside the folder Library -> Preferences.
- Found in **/private/var/mobile/Library/Preferences/com.highaltitudehacks.dvia.plist** for iOS 13.



NSUserDefaults

- All the key/value pairs stored using NSUserDefaults can be found in this file.

com.highaltitudehacks.dvia.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyLi
<plist version="1.0">
<dict>
        <key>DemoValue</key>
        <string>important info </string>
        <key>WebDatabaseDirectory</key>
        <string>/var/mobile/Applications/33EF9650-E655-4E7D-BB4C-DEE63E00C5B8/Library/Cach
        <key>WebKitDiskImageCacheSavedCacheDirectory</key>
        <string></string>
        <key>WebKitLocalStorageDatabasePathPreferenceKey</key>
        <string>/var/mobile/Applications/33EF9650-E655-4E7D-BB4C-DEE63E00C5B8/Library/Cach
        <key>WebKitOfflineWebApplicationCacheEnabled</key>
        <true/>
        <key>WebKitShrinksStandaloneImagesToFit</key>
        <true/>
        <key>loggedIn</key>
        <true/>
</dict>
</plist>
```

Core Data

- Core Data framework is used to store persistent data, manage relationships between objects etc.
- Information is again saved unencrypted on the device in .db or .sqlite files.
- An attacker can gather information about Core data objects by using a sqlite client.

Core Data

- Navigate to your application directory and look for files with the extension .db or .sqlite.
- Use an sqlite client to access these files.

```
Prateeks-iPhone:~ root# cd /var/mobile/Applications/
Prateeks-iPhone:/var/mobile/Applications root# cd 33EF9650-E655-4E7D-BB4C-DEE63E00C5B8
Prateeks-iPhone:/var/mobile/Applications/33EF9650-E655-4E7D-BB4C-DEE63E00C5B8 root# cd Documents/
Prateeks-iPhone:/var/mobile/Applications/33EF9650-E655-4E7D-BB4C-DEE63E00C5B8/Documents root# sqlite3 CoreData.sqlite
SQLite version 3.7.13
Enter ".help" for instructions
sqlite>
```

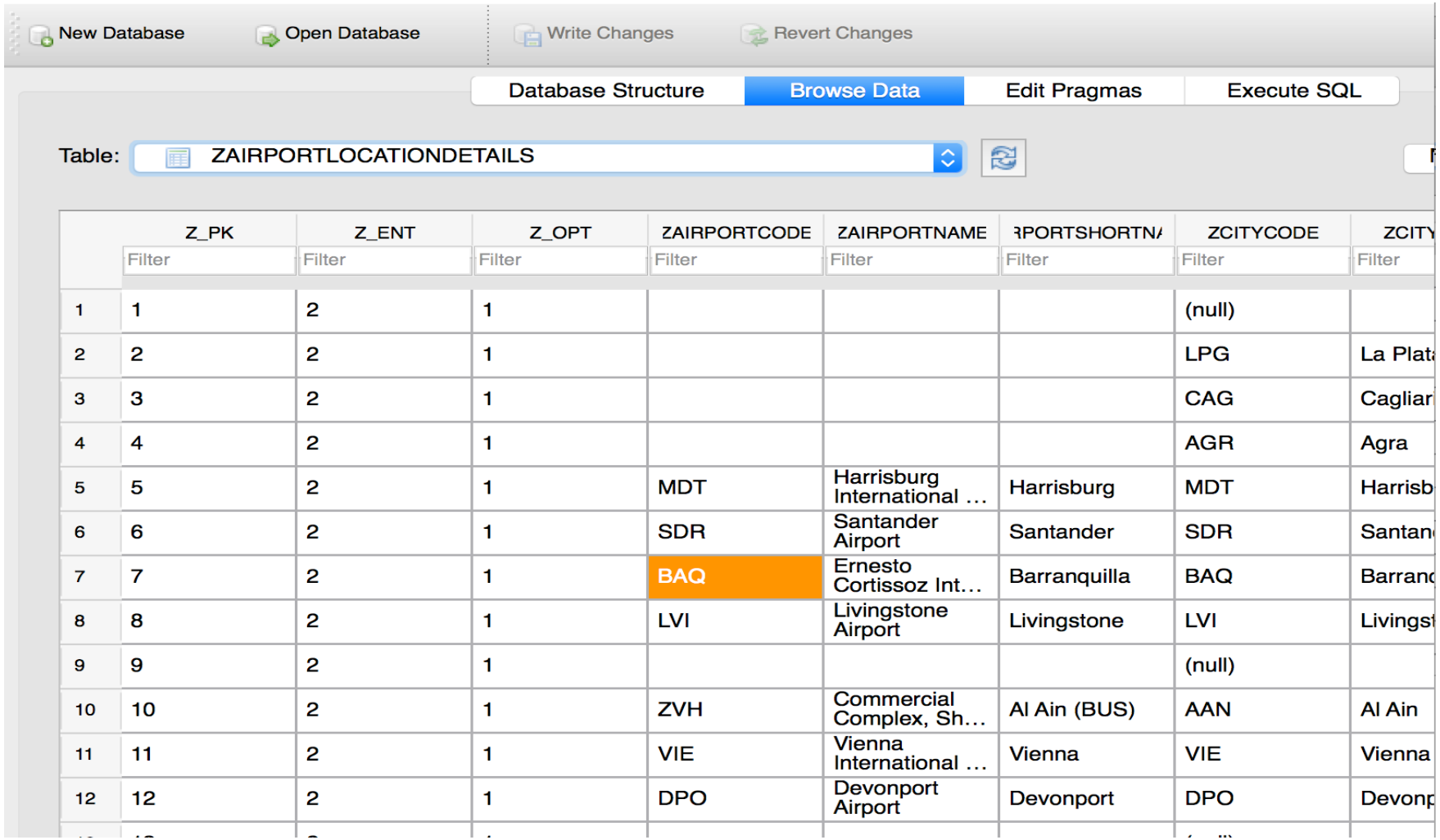
Core Data

- Navigate to your application directory and look for files with the extension .db or .sqlite.
- Use an sqlite client to access these files.
- You can dump information from the tables in the database using the commands as shown in the image below.

```
Prateeks-iPhone:/var/mobile/Applications/33EF9650-E655-4E7D-BB4C-DEE63E00C5B8/Documents root# sqlite3 CoreData.sqlite
SQLite version 3.7.13
Enter ".help" for instructions
sqlite> .tables
ZUSER          Z_METADATA     Z_PRIMARYKEY
sqlite> .headers on
sqlite> select * from ZUSER;
Z_PK|Z_ENT|Z_OPT|ZEMAIL|ZNAME|ZPASSWORD|ZPHONE
1|1|1|baggage@hhsha.com|jags|hahgahaha|515455454
2|1|1|busisi@hahs.com|jagsjshs|ksisisis|5154554545
sqlite>
```

Core Data

- Sqlite browser can be used for quick analysis of sqlite databases



	Z_PK	Z_ENT	Z_OPT	ZAIRPORTCODE	ZAIRPORTNAME	RPORTSHORTNA	ZCITYCODE	ZCITY
	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
1	1	2	1				(null)	
2	2	2	1				LPG	La Plat
3	3	2	1				CAG	Cagliar
4	4	2	1				AGR	Agra
5	5	2	1	MDT	Harrisburg International ...	Harrisburg	MDT	Harrisb
6	6	2	1	SDR	Santander Airport	Santander	SDR	Santan
7	7	2	1	BAQ	Ernesto Cortissoz Int...	Barranquilla	BAQ	Barranc
8	8	2	1	LVI	Livingstone Airport	Livingstone	LVI	Livingst
9	9	2	1				(null)	
10	10	2	1	ZVH	Commercial Complex, Sh...	Al Ain (BUS)	AAN	Al Ain
11	11	2	1	VIE	Vienna International ...	Vienna	VIE	Vienna
12	12	2	1	DPO	Devonport Airport	Devonport	DPO	Devonp

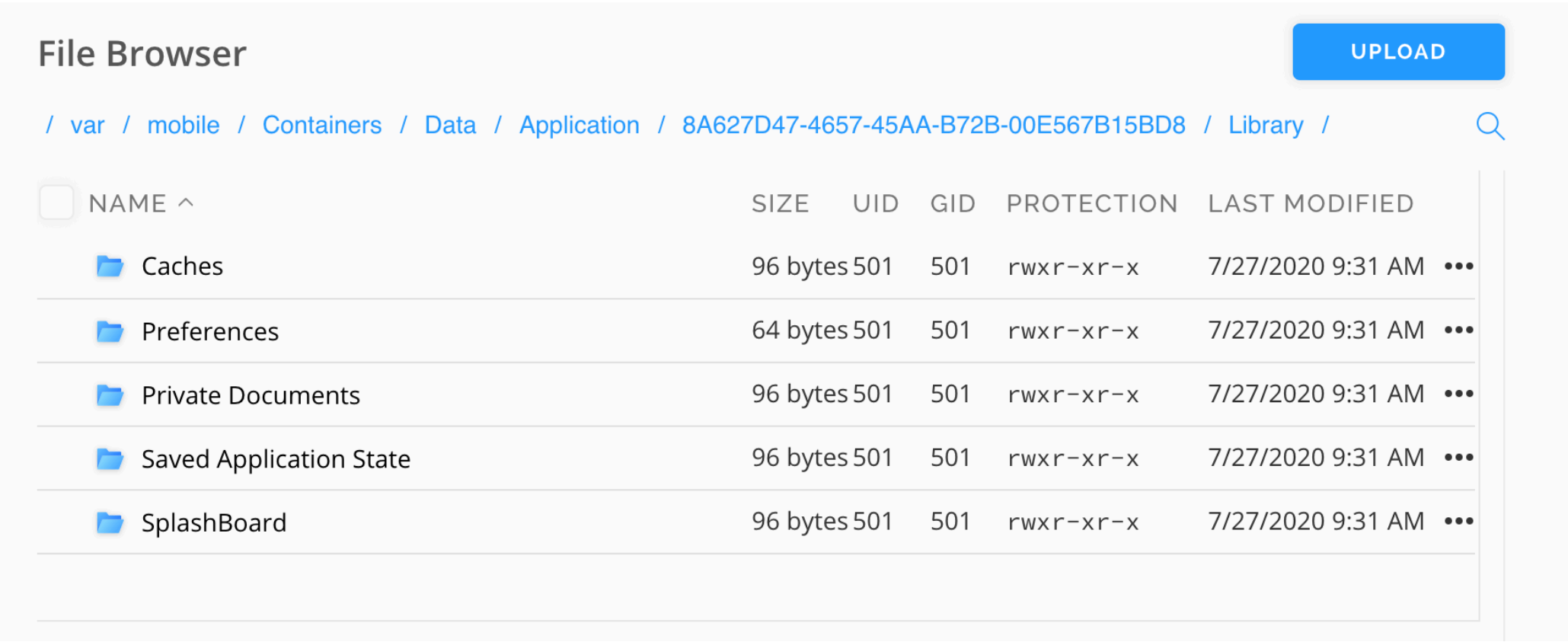
Core Data

- Core data framework should not be used to store confidential information as the information is stored unencrypted on the device.
- If you want to use some confidential informaiton, encrypt it before saving locally or use some wrappers over core data that store encrypted information on the device.

Task - 10 mins

- Start the Corellium VM
- Run the **DVIA** app and fill data under all sections in the Insecure Local data storage section
- Find the place where the application is storing data
- (**find / -name *com.highaltitudehacks.dvia***)
- Solve the challenges in Insecure Data Storage for Plist, NSUserDefaults and CoreData
- Use public frida scripts, for e.g this one to read NSUserDefaults https://github.com/noobpk/frida-ios-hook/blob/master/frida-scripts/read-nsuserdefaults.js

Solution



Solution - UserDefaults with Frida

```
[Remote::DVIA]-> console.warn("[*] Started: Read NSUserDefaults PLIST file");
if (ObjC.available)
{
    try
    {
        var NSUserDefaults = ObjC.classes.NSUserDefaults;
        var NSDictionary = NSUserDefaults.alloc().init().dictionaryRepresentation();
        console.log(NSDictionary.toString())
    }
    catch(err)
    {
        console.warn("[!] Exception: " + err.message);
    }
}
else
{
    console.warn("Objective-C Runtime is not available!");
}
console.warn("[*] Completed: Read NSUserDefaults PLIST file");
timeout was reached
```

Solution - UserDefaults with Frida

```
    CarCapabilities =     {
        CarCapabilitiesDefaultIdentifier =         {
            CRCapabilitiesDisabledFeatureKey = 0;
            CRCapabilitiesUserInterfaceStyleKey = 2;
            CapabilitiesDashboardRoundedCornersKey = "{{0, 0}, {0, 0}}";
            CapabilitiesNowPlayingAlbumArtKey = 2;
            CapabilitiesViewAreaInsetKey = "{{0, 0}, {0, 0}}";
        };
    };
    DemoValue = secretvalue;
    NSAllowsDefaultLineBreakStrategy = 1;
    NSInterfaceStyle = macintosh;
    NSLanguages =     (
        en
    );
    WebDatabaseDirectory = "/var/mobile/Containers/Data/Application/4311C309-D54F-4B93-8FEA-EBD2DEB6217D/Library/Caches";
    WebKitLocalStorageDatabasePathPreferenceKey = "/var/mobile/Containers/Data/Application/4311C309-D54F-4B93-8FEA-EBD2DEB6217D/Library/Ca
ches";
    WebKitOfflineWebApplicationCacheEnabled = 1;
    WebKitShrinksStandaloneImagesToFit = 1;
    "com.apple.content-rating.AppRating" = 1000;
    "com.apple.content-rating.ExplicitBooksAllowed" = 1;
    "com.apple.content-rating.ExplicitMusicPodcastsAllowed" = 1;
    "com.apple.content-rating.MovieRating" = 1000;
    "com.apple.content-rating.TVShowRating" = 1000;
    loggedIn = 1;
}
[*] Completed: Read NSUserDefaults PLIST file
```

Keychain

- It is the most secure way of storing information locally on the device.
- Used by applications store sensitive information like session ID, authentication tokens, encryption keys etc.
- Common database for all apps in a sqlite file. Data is stored outside of the application sandbox.
- Currently, information stored in the keychain can only be dumped from a jailbroken device using a tool named Keychain Dumper (https://github.com/ptoomey3/Keychain-Dumper)
- Downside is that data stays even after application uninstall

Keychain

Can be updated by using four main operations

- SecItemAdd
- SecItemUpdate
- SecItemCopyMatching
- SecItemDelete

Data Protection for Keychain items can be configured by setting the **kSecAttrAccessible** attribute key

Keychain accessibility values

Accessibility Values

Values you use with the `kSecAttrAccessible` attribute key, listed from most to least restrictive.

`let kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly: CFString`

The data in the keychain can only be accessed when the device is unlocked. Only available if a passcode is set on the device.

`let kSecAttrAccessibleWhenUnlockedThisDeviceOnly: CFString`

The data in the keychain item can be accessed only while the device is unlocked by the user.

`let kSecAttrAccessibleWhenUnlocked: CFString`

The data in the keychain item can be accessed only while the device is unlocked by the user.

`let kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly: CFString`

The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

`let kSecAttrAccessibleAfterFirstUnlock: CFString`

The data in the keychain item cannot be accessed after a restart until the device has been unlocked once by the user.

Source: https://developer.apple.com/documentation/security/keychain_services/keychain_items/item_attribute_keys_and_values#1679100

Touch ID to fetch Keychain items

Can be implemented by using **LocalAuthentication** or **Security** framework

Recommended Reading: https://developer.apple.com/documentation/localauthentication/accessing_keychain_items_with_face_id_or_touch_id

Keychain wrappers

- One line implementation.
- Access group allows applications to share keychain data.
- **KeychainItemWrapper *wrapper = [[KeychainItemWrapper alloc] initWithIdentifier:@“Identifier” accessGroup:nil];**
- The same access group has to be given from both the apps and both the app ID’s have to be mentioned in the plist file for both the applications.

Keychain wrappers

- Using Keychain is quite simple (especially with third-party wrappers). One of them is SSKeychain (https://github.com/soffes/sskeychain):
- [SSKeychain setPassword:@”secretkey” forService:@”DVIA” account:@”Admin”];
- NSString *pass = [SSKeychain passwordForService:@”DVIA” account:@”Admin”];

Storing Data in Keychain

- Using Keychain is quite simple (especially with third-party wrappers). One of them is SSKeychain (https://github.com/soffes/sskeychain):
- [SSKeychain setPassword:@”secretkey” forService:@”DVIA” account:@”Admin”];
- NSString *pass = [SSKeychain passwordForService:@”DVIA” account:@”Admin”];

Keychain dumper demo

Keychain information dumped for the application Damn Vulnerable iOS app can be clearly found in the image below.

```
Generic Password
----------------
Service: HighAltitudeHacks.com.DamnVulnerableIOSApp
Account: keychainValue
Entitlement Group: 9UBS947V73.HighAltitudeHacks.com.DamnVulnerableIOSApp
Label: (null)
Generic Field: (null)
Keychain Data: fes
```

Even though keychain is one of the most secure places to store information, consider adding an extra layer of encryption before saving data in the application to make the job for the attacker more difficult.

Keychain dumper in iOS 13

Need to sign with the following entitlements.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
        <dict>
                <key>keychain-access-groups</key>
                <array>
                        <string>*</string>
                </array>
                <key>platform-application</key> <true/>
        </dict>
</plist>
```

Build instructions at **https://github.com/ptoomey3/Keychain-Dumper**

Keychain dumper in iOS 13

```
Prateek:Keychain-Dumper-master prateekg147$ CER="55DC5C9
CD518016C" make codesign
codesign -fs "55DC5C92BA5AC3207CCC5C36FFB921DCD518016C" --entitlements entitleme
nts.xml keychain_dumper
keychain_dumper: replacing existing signature
Prateek:Keychain-Dumper-master prateekg147$ ./keychain_dumper
-bash: ./keychain_dumper: Bad CPU type in executable
```

```
Prateek:Keychain-Dumper-master prateekg147$ scp -P 2222 keychain_dumper
 root@localhost:/bin/
root@localhost's password:
keychain_dumper                               100%  207KB  14.5MB/s   0
0:00
```

Build instructions at **https://github.com/ptoomey3/Keychain-Dumper**

Task

- Start the Corellium VM
- Run the following command to dump the keychain:

  keychain_dumper
- Can you dump all the keychain contents for the whole device using Frida by hooking into only one app ?

keychain_dumper

- Explore the different options in Keychain Dumper

```
Prateeks-iphone:~/Keychain-Dumper-master root# ./keychain_dumper -h
Usage: keychain_dumper [-e]|[-h]|[-agnick]
<no flags>: Dump Password Keychain Items (Generic Password, Internet Passwords)
-a: Dump All Keychain Items (Generic Passwords, Internet Passwords, Identities, Certificates, and Keys
-e: Dump Entitlements
-g: Dump Generic Passwords
-n: Dump Internet Passwords
-i: Dump Identities
-c: Dump Certificates
-k: Dump Keys
Prateeks-iphone:~/Keychain-Dumper-master root#
```

Realm

- Realm is a popular third-party cross-platform mobile database built on a custom C++ core.
- Just like in all other cases information is stored unencrypted in the /Documents folder, in the *.realm files.
- Realm is a replacement for SQLite & Core Data
- An attacker can gather information using the default Realm Browser: https://github.com/realm/realm-cocoa/tree/master/tools/RealmBrowser

Firebase

- Another popular SDK used to create apps without the need to write backend code.
- Data is written in Firebase databases.
- In the past, there have been issues with misconfigured Firebase databases.
- **PROJECT_ID** key in the **GoogleService-Info.plist** contains the Project Name.
- One can then use FireBase scanner to scan for misconfigured databases.

python FirebaseScanner.py -f <commaSeperatedFirebaseProjectNames>

Apple Watch

- Apple Watch is the wearable device from Apple.
- Most of the Apple Watch apps are simply extensions of native iOS applications.

WATCH

Apple Watch

- The running app extension and containing app have no direct access to each other's containers, so they use a shared data directory for data syncing.
- Their shared data can be found in **/var/mobile/Containers/Shared/AppGroup/<ID>**
- So, all the Insecure Data Storage vulnerabilities can be applied to Apple Watch extensions.

Side Channel Data leakage

There are many different ways in which data can be leaked from the application without the awareness of the developer.

- Device Logs
- Application snapshots
- Pasteboard
- Keystroke logging
- Cookies

Device Logs

- Some developer use logs while debugging their applications but forget to remove them while releasing the application.
- To see the device logs while you are running an application, make sure that the device is connected to your computer.
- In Xcode, go to **Window -> Organizer -> Device -> Your Device -> Console**
- In Recent iOS versions , data is logged with the **os_log** command
- On your UbuntuVM, you can use the command **idevicesyslog** to view the logs from a USB connected device

Device Logs

```
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Notice>: MS:Notice: Loading: /Library/MobileSubstrate/DynamicLibraries/Accelerate.dylib
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Warning>: Accelerate: Reloading preferences, instantaneously! :)
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Notice>: MS:Notice: Loading: /Library/MobileSubstrate/DynamicLibraries/Veency.dylib
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Notice>: MS:Notice: Loading: /Library/MobileSubstrate/DynamicLibraries/WeeLoader.dylib
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Warning>: MS:Warning: message not found [BBServer _loadAllWeeAppSections]
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Warning>: MS:Warning: message not found [BBServer _loadAllDataProviderPluginBundles]
Mar 20 16:52:51 Prateeks-iPhone assistantd[1399] <Notice>: MS:Notice: Loading: /Library/MobileSubstrate/DynamicLibraries/libstatusbar.dylib
Mar 20 16:53:01 Prateeks-iPhone wifid[74] <Notice>: WiFi:[417007381.006427]: WiFiLocaleManagerCheckLocale: location service or wifi is off, set locale to default
Mar 20 16:53:01 Prateeks-iPhone wifid[74] <Notice>: WiFi:[417007381.007370]: new locale:  , locale:
Mar 20 16:53:09 Prateeks-iPhone DamnVulnerableIOSApp[1330] <Warning>: user saved: <Person:new:(null)> {
        email = "trfdd@the.com";
        name = trfdd;
        password = Gfddghh342;
        phone = 5087456980;
    }
Mar 20 16:53:11 Prateeks-iPhone DamnVulnerableIOSApp[1330] <Warning>: Saved user info: <Person:lxNA4r3ull:(null)> {
        email = "trfdd@the.com";
        name = trfdd;
        password = Gfddghh342;
        phone = 5087456980;
    }
```

- Device logs should only be enabled for **DEBUG** mode in the application, this will ensure that the logs are disabled when the application is downloaded from the App store and run on a user's device.

Disabling Device Logs

Add the following code in the Precompiled header file , usually with the name

PrefixHeader.pch

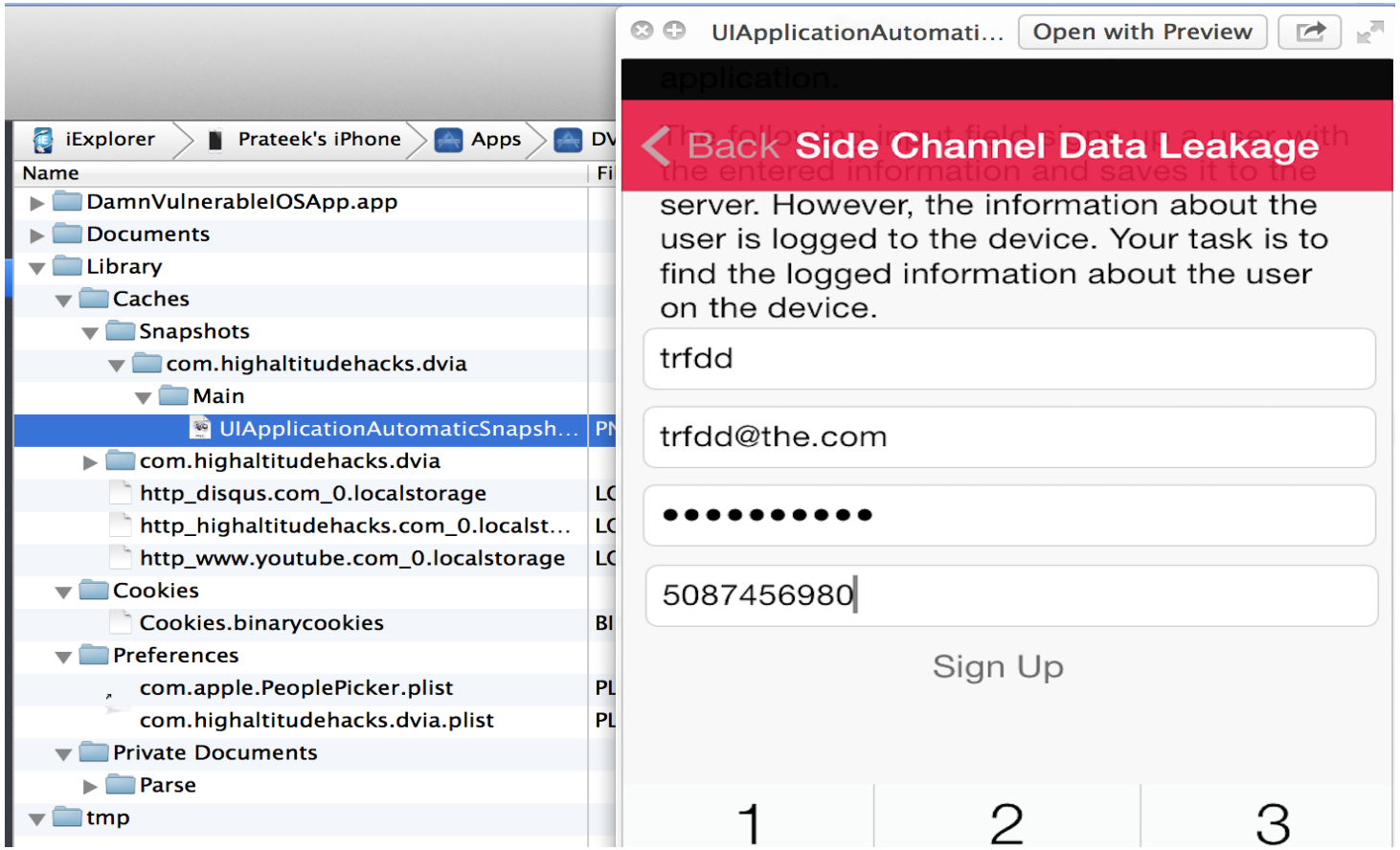
```
#ifdef DEBUG
# define NSLog (...) NSLog(__VA_ARGS__)
#else
# define NSLog (...)
#endif
```

Application Snapshots

- iOS by default takes a screenshot of your application when you take the application to background by pressing the home button.
- This screenshot is shown to the user when he opens the app again while the app is loaded in the background.
- Provides a seamless experience.
- The problem is that the screenshot is stored without any protection in the application folder.
- Sometimes, these screenshots can contain confidential information that might be leaked to an attacker.
- Location is **./private/var/mobile/Containers/Data/Application/<APP-ID>/Library/SplashBoard/Snapshots** for iOS 13

Application Snapshots

- The following image shows the application snapshot stored in the application folder

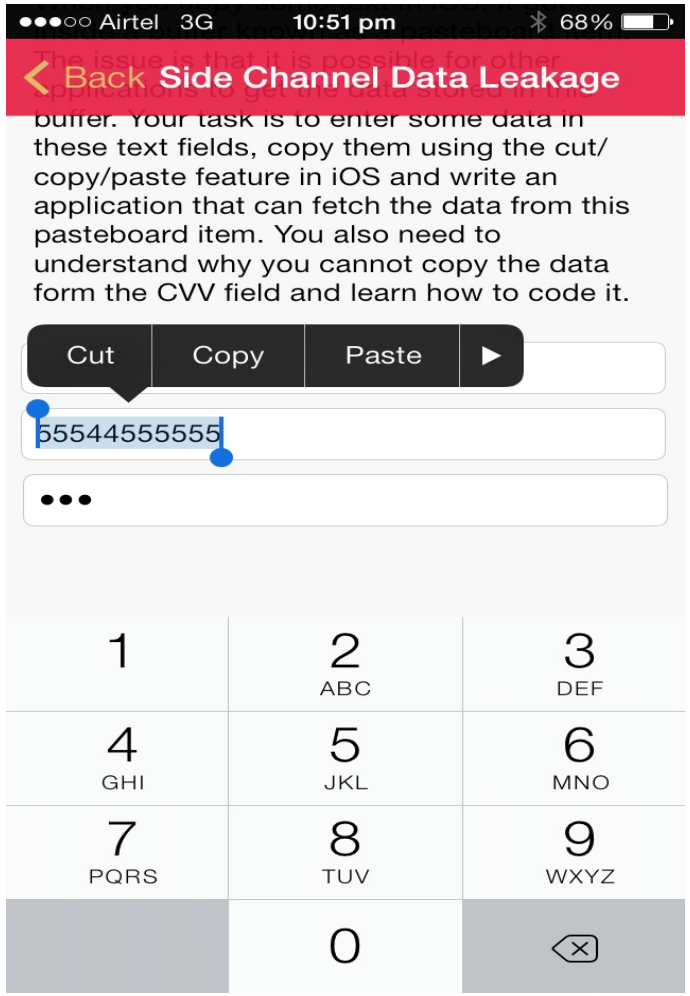


Pasteboard

- Data copied using the cut/copy features in iOS goes inside a buffer known inside a pasteboard item.
- **It is possible for other applications to access the content of this pasteboard.**
- If the pasteboard item contains some confidential information, it might lead to information leakage
- Fixed in iOS 14 - User sees each copy/paste notification

Pasteboard

- Data can be copied using the Copy feature in iOS.
- Once it is copied, it remains in the buffer.



Pasteboard

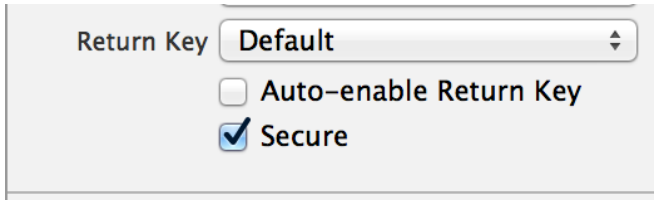
Using the following code in any app you can dump out the contents of the pasteboard.

[UIPasteboard generalPasteboard].items[0] -> Objective-C

UIPasteboard.general.string -> Swift

Pasteboard

- For text fields that might contain secure information, make sure the *Secure* property is set.



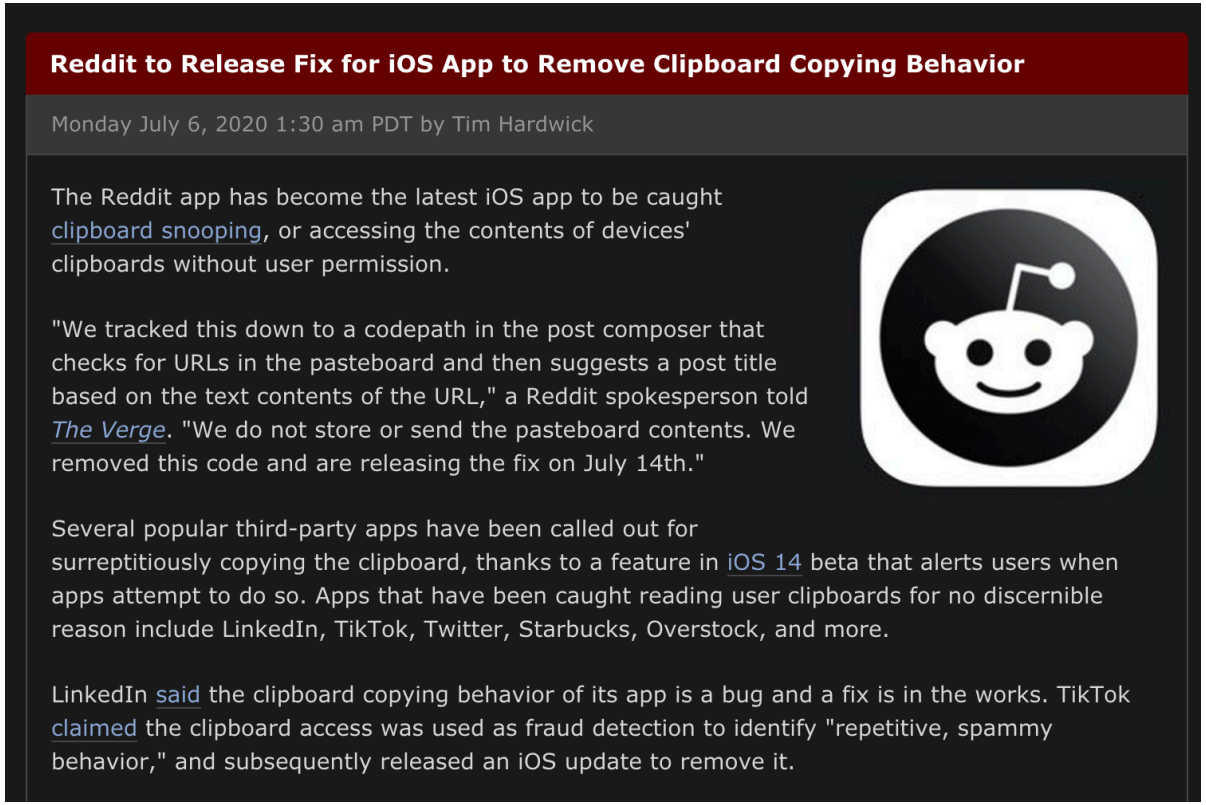
- Clear pasteboard contents when the application enters background.

```
[UIPasteboard generalPasteboard].items = nil;
```

- Use pasteboard with specific identifiers, this makes it difficult for other applications to fetch data from this pasteboard item.

```
[UIPasteboard pasteboardWithName:@"DVIA" create:YES];
```

Pasteboard



Reddit to Release Fix for iOS App to Remove Clipboard Copying Behavior

Monday July 6, 2020 1:30 am PDT by Tim Hardwick

The Reddit app has become the latest iOS app to be caught clipboard snooping, or accessing the contents of devices' clipboards without user permission.

"We tracked this down to a codepath in the post composer that checks for URLs in the pasteboard and then suggests a post title based on the text contents of the URL," a Reddit spokesperson told *The Verge*. "We do not store or send the pasteboard contents. We removed this code and are releasing the fix on July 14th."

Several popular third-party apps have been called out for surreptitiously copying the clipboard, thanks to a feature in iOS 14 beta that alerts users when apps attempt to do so. Apps that have been caught reading user clipboards for no discernible reason include LinkedIn, TikTok, Twitter, Starbucks, Overstock, and more.

LinkedIn said the clipboard copying behavior of its app is a bug and a fix is in the works. TikTok claimed the clipboard access was used as fraud detection to identify "repetitive, spammy behavior," and subsequently released an iOS update to remove it.

Source: https://www.macrumors.com/2020/07/06/reddit-app-clipboard-snooping-fix-coming/

Pasteboard

LinkedIn iOS Clipboard Copying Was Bug

LIKE DISCUSS

JUL 03, 2020 • 2 MIN READ

by

Alex Blewitt

FOLLOW

When iOS 14 (Beta) was released at WWDC, a number of new security features were added. One of them was the ability to find out if an app was interacting with the clipboard at unusual times; many apps have been indicated that they are copying the clipboard unnecessarily. This could be a problem if the user has recently copied a sensitive password or other personal application, leading some to question whether this was an intentional feature.

@DonCubed reported on twitter that the LinkedIn app was copying data on each keystroke, and thanks to the fact that it's possible to share clipboard data from a linked macOS system, it was able to copy data that had been used on the laptop as well:

> LinkedIn is copying the contents of my clipboard every keystroke. IOS 14 allows users to see each paste notification.

Source: https://www.infoq.com/news/2020/07/ios-clipboard-bug/

Keystroke logging

- iOS by default logs every input that you enter in any text field unless the secure flag is not set.
- This helps in autocorrecting the user later.
- All the keystroke logs can be easily fetched out from a device.
- These logs might contain information that is important.
- Logs remain stored on the device for a long time hence making it even more insecure.
- Logs are stored in a file with the extension .dat in the location **“/var/mobile/Library/Keyboard/“**

Keystroke logging

- The prefix of the file denotes the language in which the keystroke logs are stored.
- Here is how a part of the logs file look like.

```
en_GB-dynamic-text.dat  hi-dynamic-text.dat


s^@off^@for^@all^@of^@us^@about^@afternoon^@Sirf^@Kaam^@ka
ne^@and^@gave^@what^@away^@think^@told^@me^@mn^@gunk^@TFT^
ichever^@hrs^@ridiculously^@hard^@l^@be^@rubbing^@running^
```

Cookies

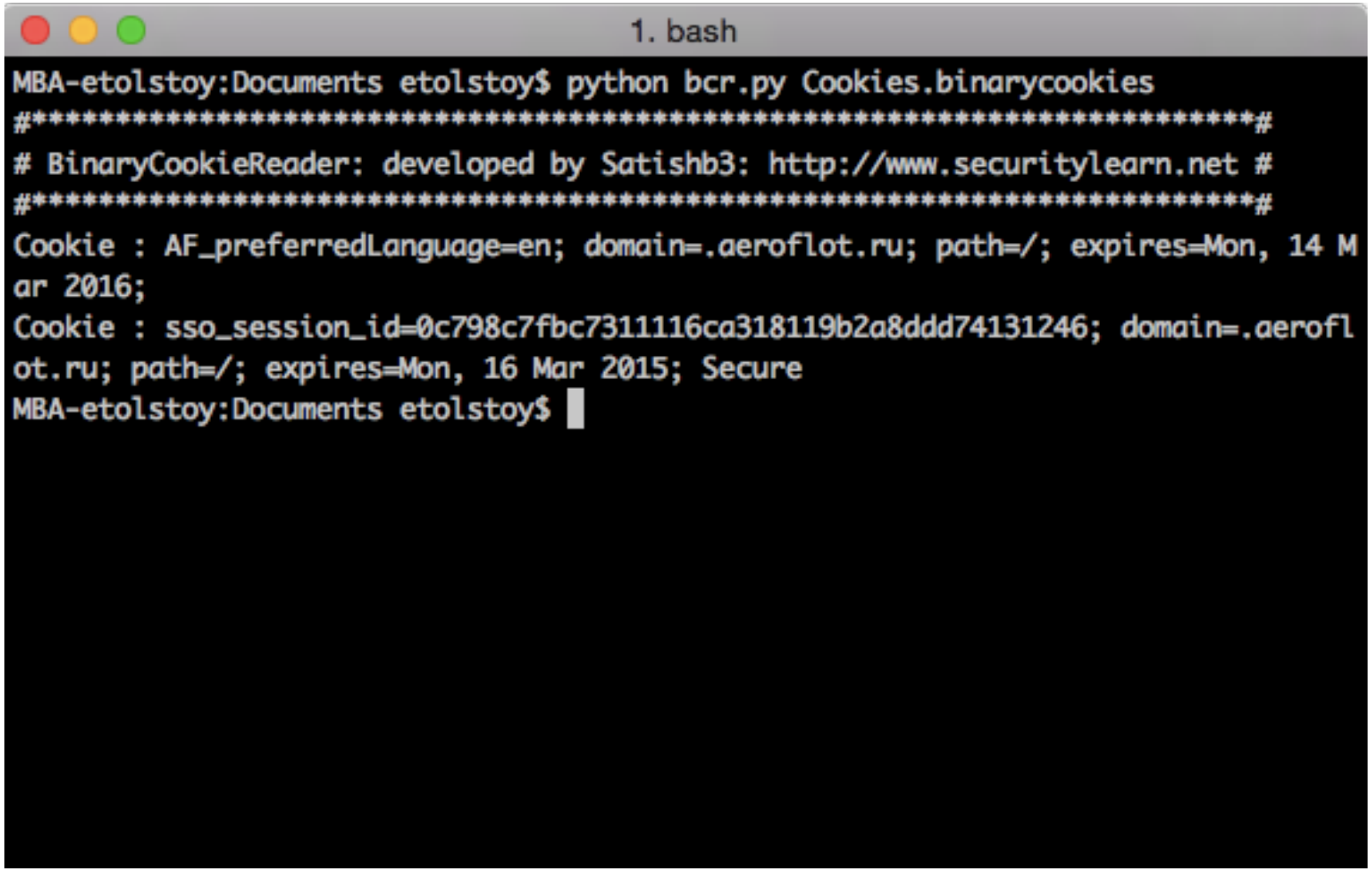
- Some applications create persistance cookies created while browsing web page and store them in **Cookies.binarycookies** file in application's home directory.
- The sample code for storing cookies:
- [[NSHTTPCookieStorage sharedHTTPCookieStorage] setCookie:cookie];
- The path to cookies is:
- **/User/Data/Application/<APP-ID>/Library/Cookies**

Cookies

- Revealing cookies is quite easy:
- Download the python script BinaryCookieReader.py from http://securitylearn.net/wp-content/uploads/tools/iOS/BinaryCookieReader.py
- Download the Cookies.binarycookies file from your device.
- Run binarycookiereader.py /file-path-to-cookies

Cookies

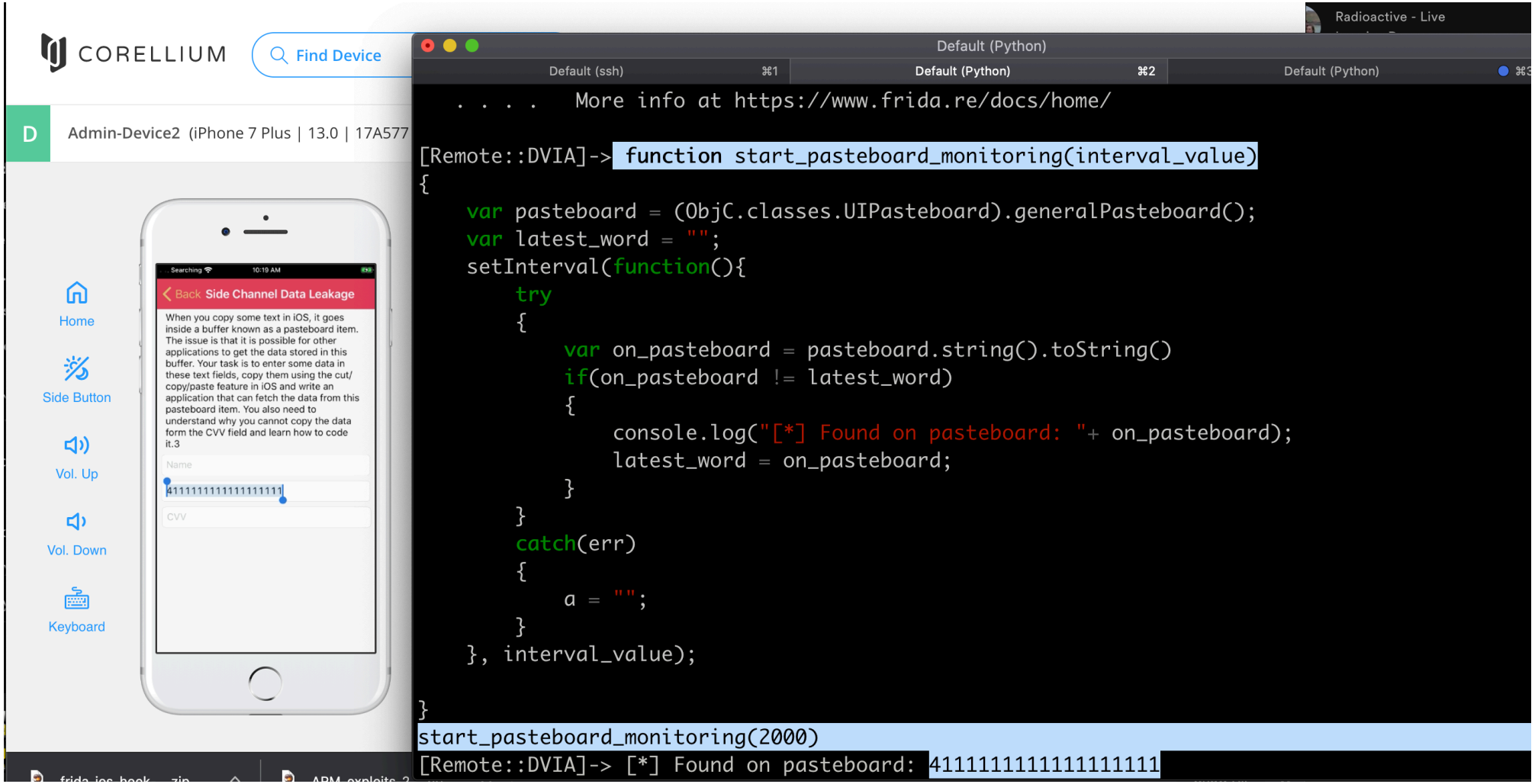
This is the example from Aeroflot mobile application:



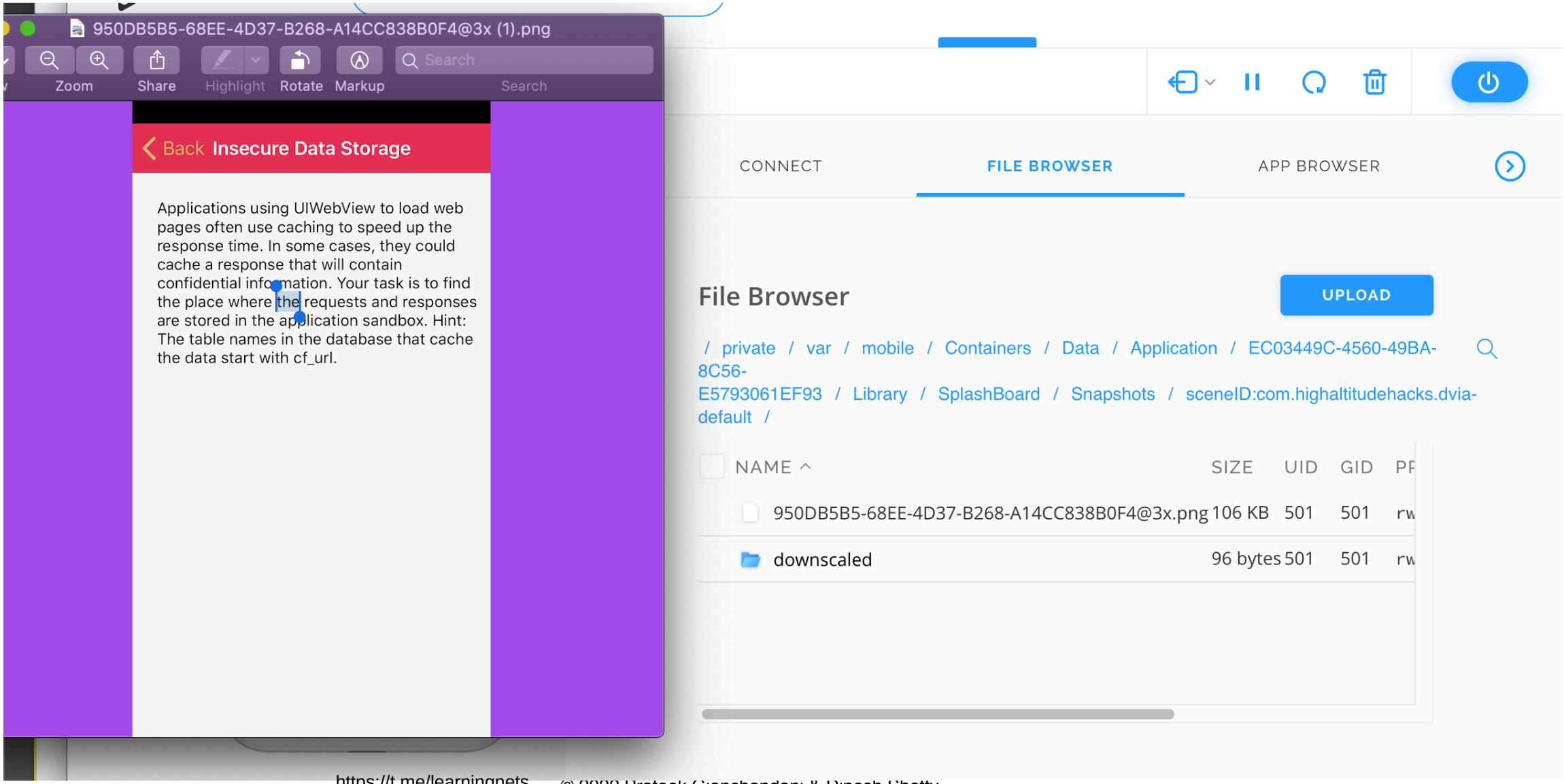
Task - 10 mins

- Dump the pasteboard contents using this Frida script https://github.com/noobpk/frida-ios-hook/blob/master/frida-scripts/pasteboard-monitoring.js
- Find the location of the Application Snapshot stored on the device for **DVIA** or **DVIA-v2**
- You can also Dump Binary Cookies using this Frida script https://github.com/noobpk/frida-ios-hook/blob/master/frida-scripts/show_binarycookies.js

Solution - Pasteboard



Solution - Application Snapshot

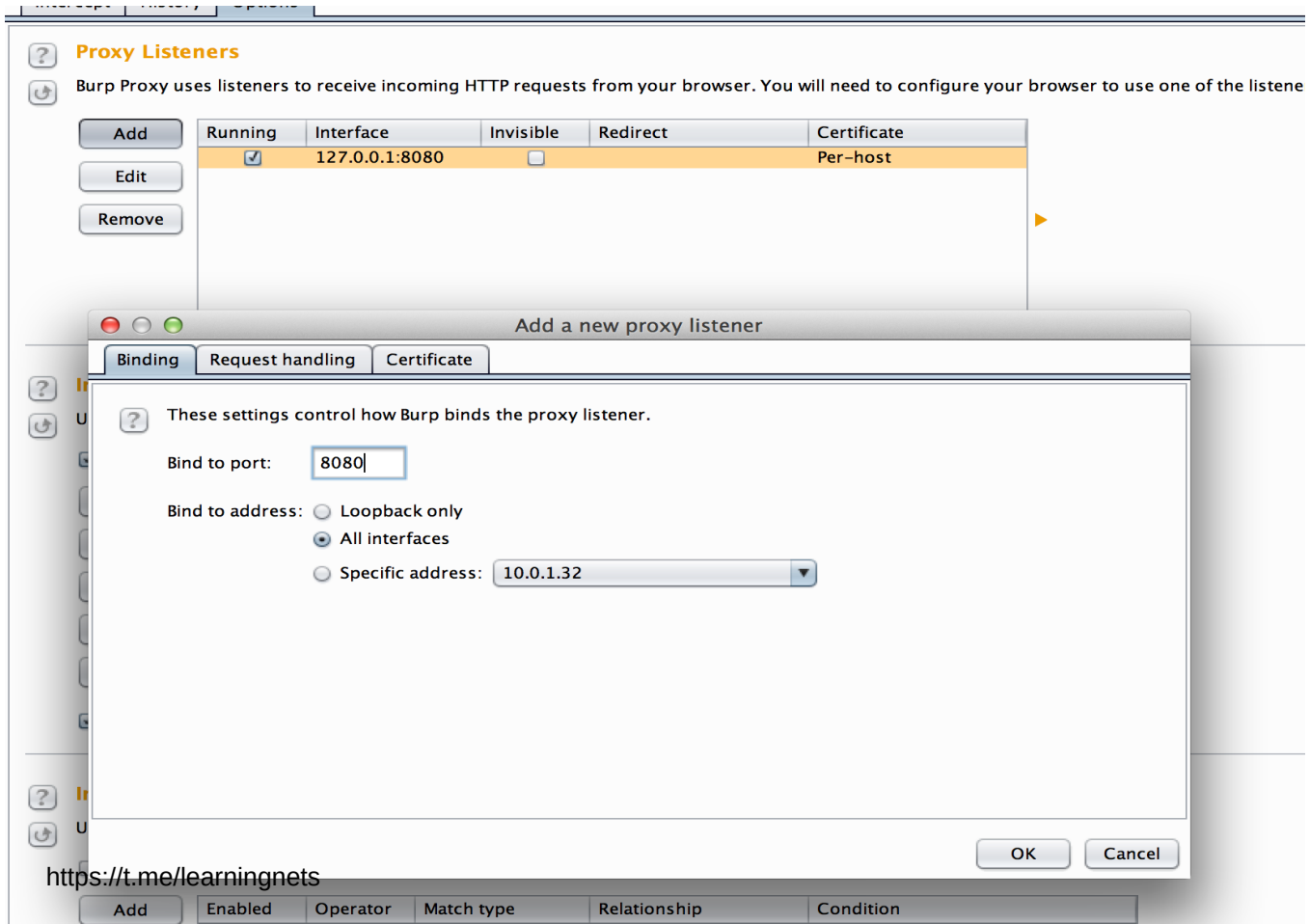


Analysing network traffic over HTTP/ HTTPs

- It is important to analyze the network traffic that flows between the client/server in an application.
- Look for credentials, authentication tokens, API keys being transmitted over unsecured http channel.
- Check for the entropy in Session ID's.
- Traffic can be analyzed using a simple proxy tool like Burp proxy.
- Try to manipulate the request/response using Burp and see how the client side application responds to it.

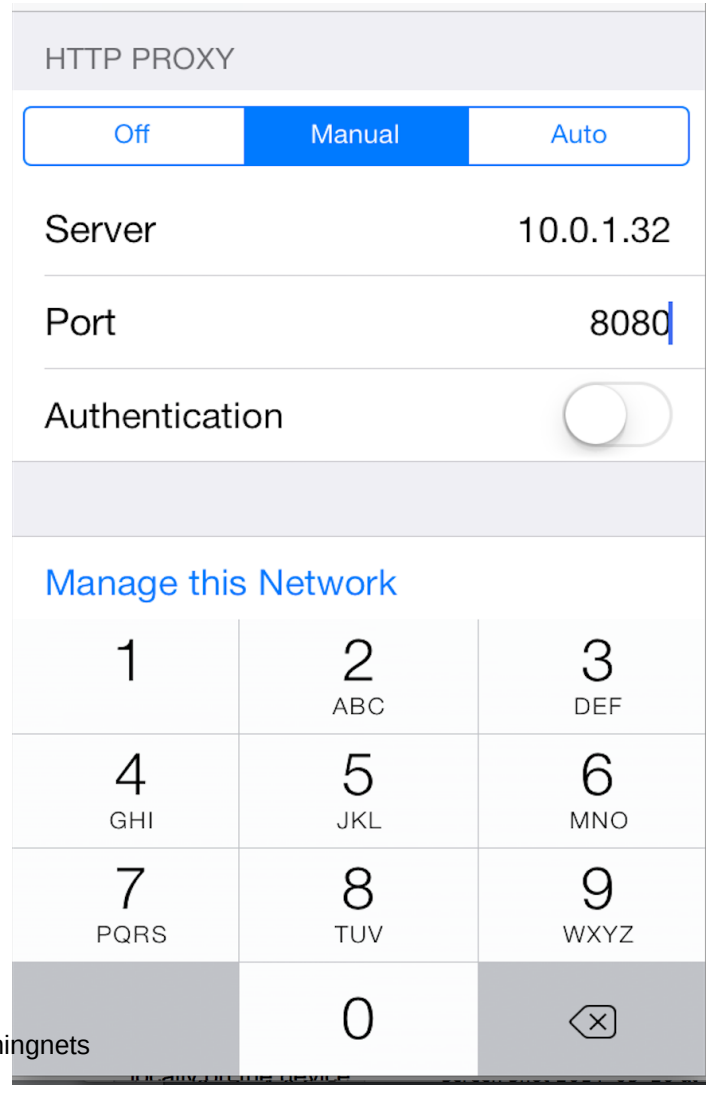
Analyzing traffic over HTTP

- Configure Burp Proxy to start listening for traffic. Make sure it is listening on all interfaces.



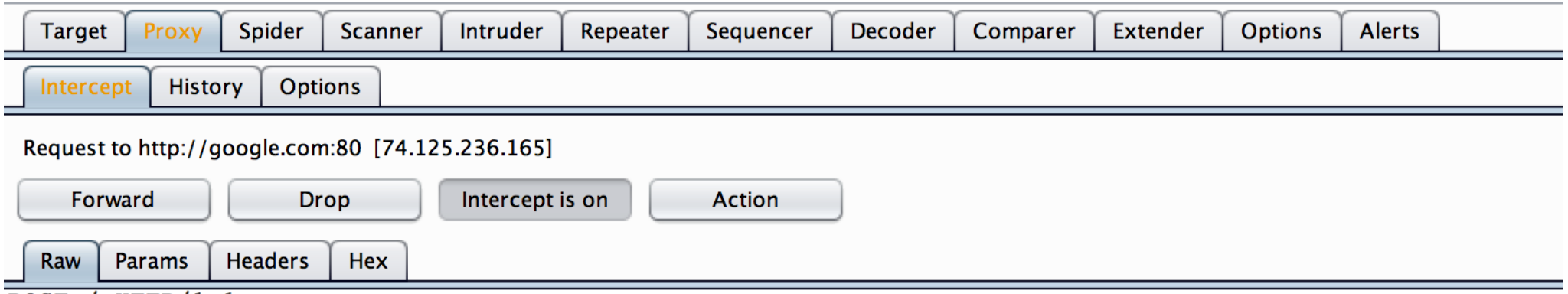
Analyzing traffic over HTTP

- Configure your iOS device to use your computer as a proxy.



Analyzing traffic over HTTP

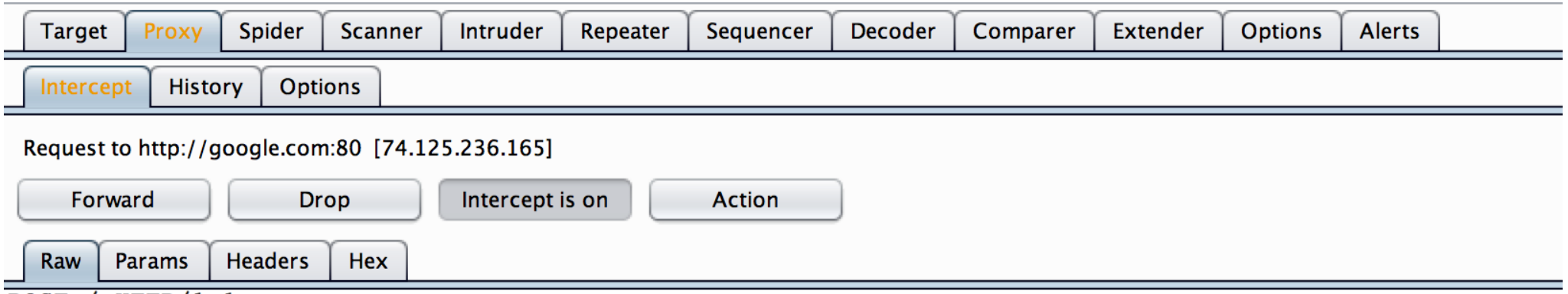
- You can now intercept the traffic as it goes to the server.




```
POST / HTTP/1.1
Host: google.com
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Cookie: NID=67=U23QDGVu1C14fYI58rj8_FBdCuDJpauRawCRSoe5mlXclpxyi8Me4sCRnUPSKiOPhnT8RkwrHNic1AyUygXry0z
PREF=ID=8f8b1380871e2351:FF=0:TM=1395305946:LM=1395305946:S=qeRBcprMxfEqHxF3
Accept-Language: en-us
Accept: */*
Content-Length: 87
Connection: keep-alive
User-Agent: DamnVulnerableIOSApp/1.0 CFNetwork/672.0.8 Darwin/14.0.0

{
  "card_name" : "Test User",
  "card_cvv" : "123",
  "card_number" : "123456789012"
}
```

Analyzing traffic over HTTP

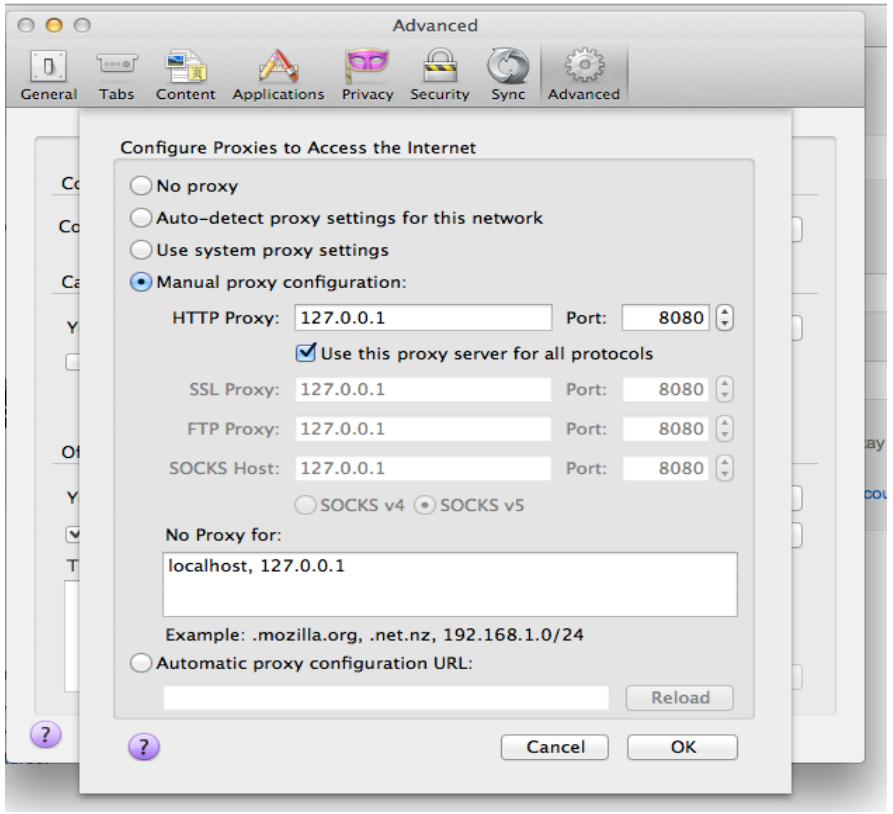
- You can now intercept the traffic as it goes to the server.




```
POST / HTTP/1.1
Host: google.com
Proxy-Connection: keep-alive
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=utf-8
Cookie: NID=67=U23QDGVu1C14fYI58rj8_FBdCuDJpauRawCRSoe5mlXclpxyi8Me4sCRnUPSKiOPhnT8RkwrHNic1AyUygXry0z
PREF=ID=8f8b1380871e2351:FF=0:TM=1395305946:LM=1395305946:S=qeRBcprMxfEqHxF3
Accept-Language: en-us
Accept: */*
Content-Length: 87
Connection: keep-alive
User-Agent: DamnVulnerableIOSApp/1.0 CFNetwork/672.0.8 Darwin/14.0.0

{
  "card_name" : "Test User",
  "card_cvv" : "123",
  "card_number" : "123456789012"
}
```

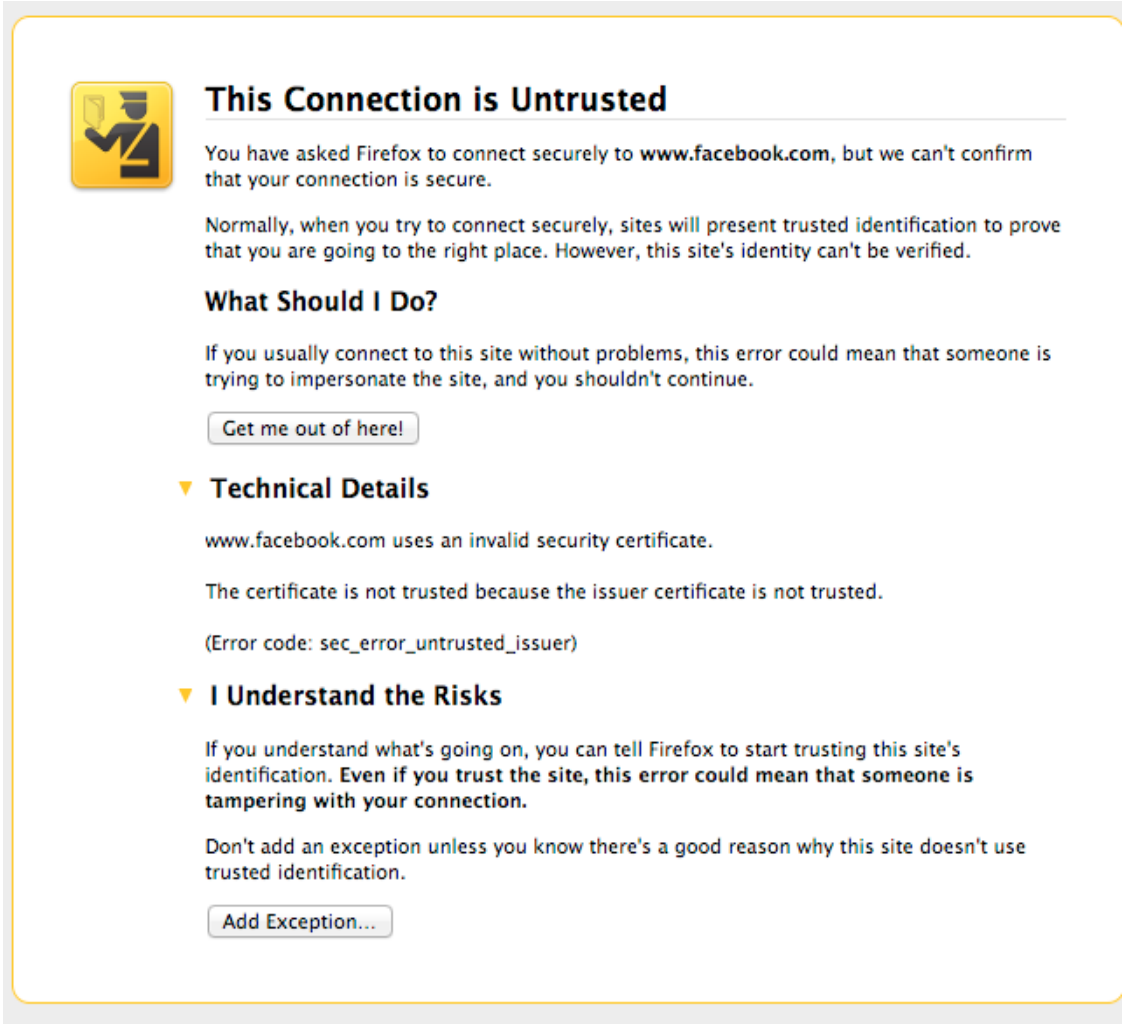
Analyzing traffic over HTTPS

- This will require you to install Burp's CA certificate as a trusted root on your device.
- Configure your browser to relay traffic over Burp proxy.



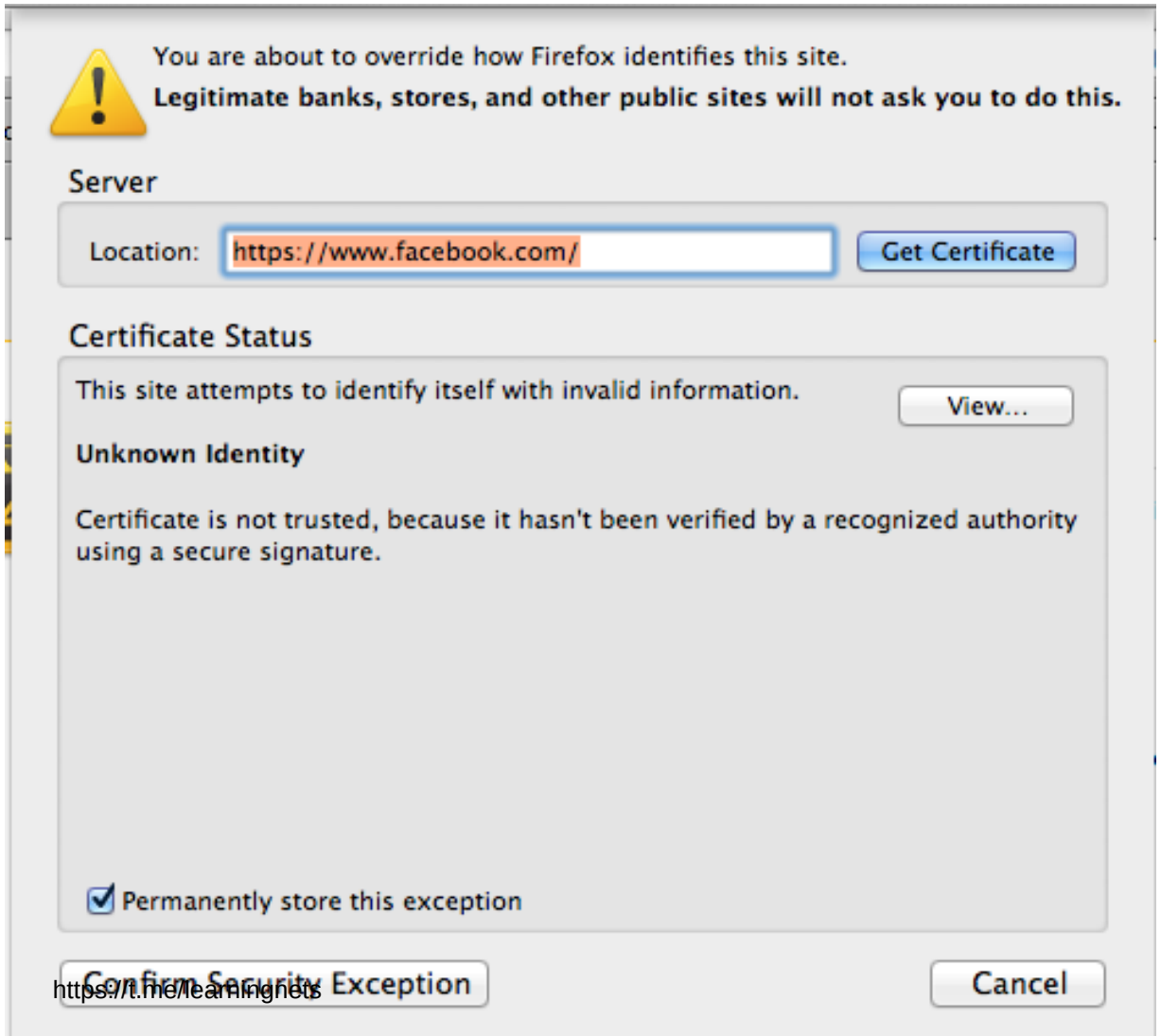
Analyzing traffic over HTTPS

- You will get a warning, click on Add Exception.



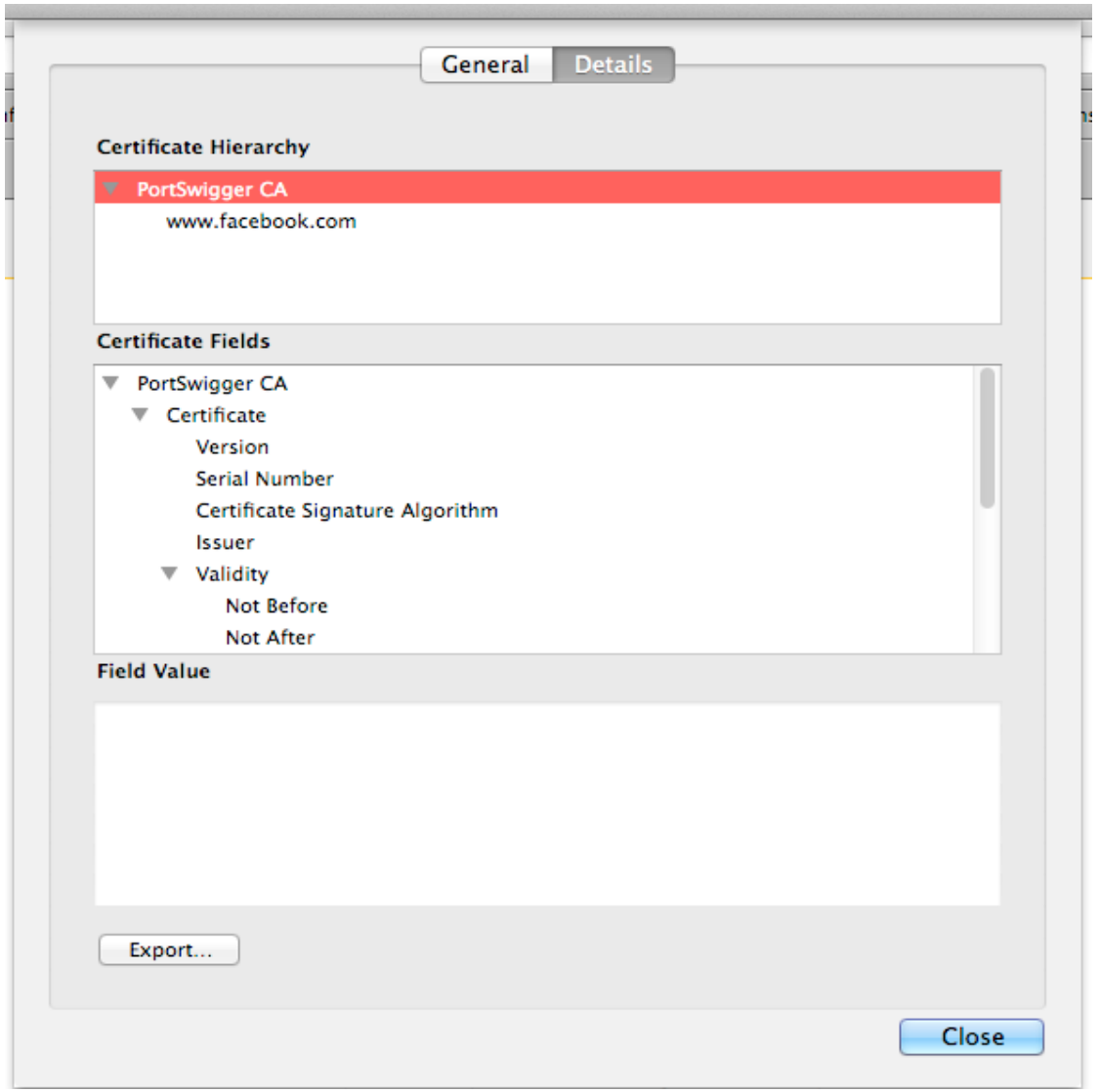
Analyzing traffic over HTTPS

- Click on View.



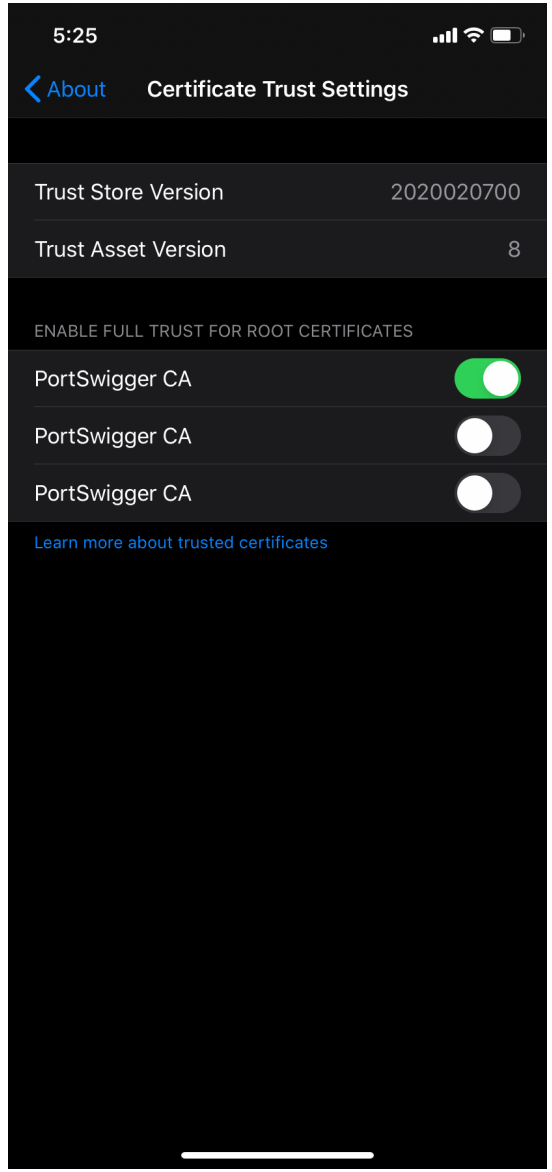
Analyzing traffic over HTTPS

- Go to Details, select the topmost certificate, click on Export and save the file with extension as .crt
- **Optionally, you can setup the proxy on your device and go to http://burp and click on CA Certificate to download the certificate**



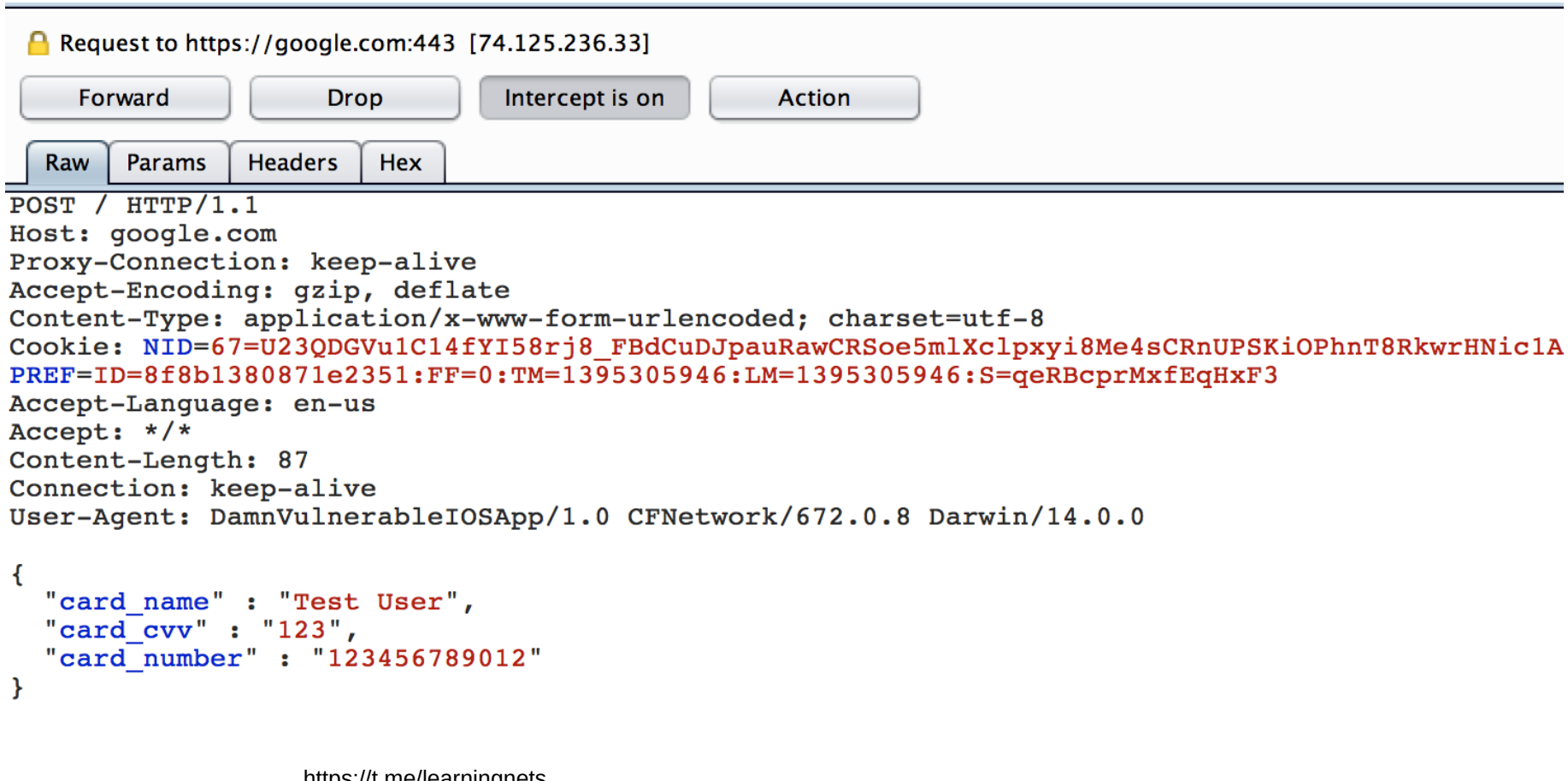
Analyzing traffic over HTTPS

- Send this file to your device via email, click on it and Install it. Accept all the instructions and click on Done.
- Explicitly Trust the Certificate by Going to **Settings -> General -> About -> Certificate Trust Settings**
- https://support.apple.com/en-ae/HT204477



Analyzing traffic over HTTPS

- Quit and restart the application you want to sniff traffic for. You will now be able to see the traffic even if it is over HTTPs

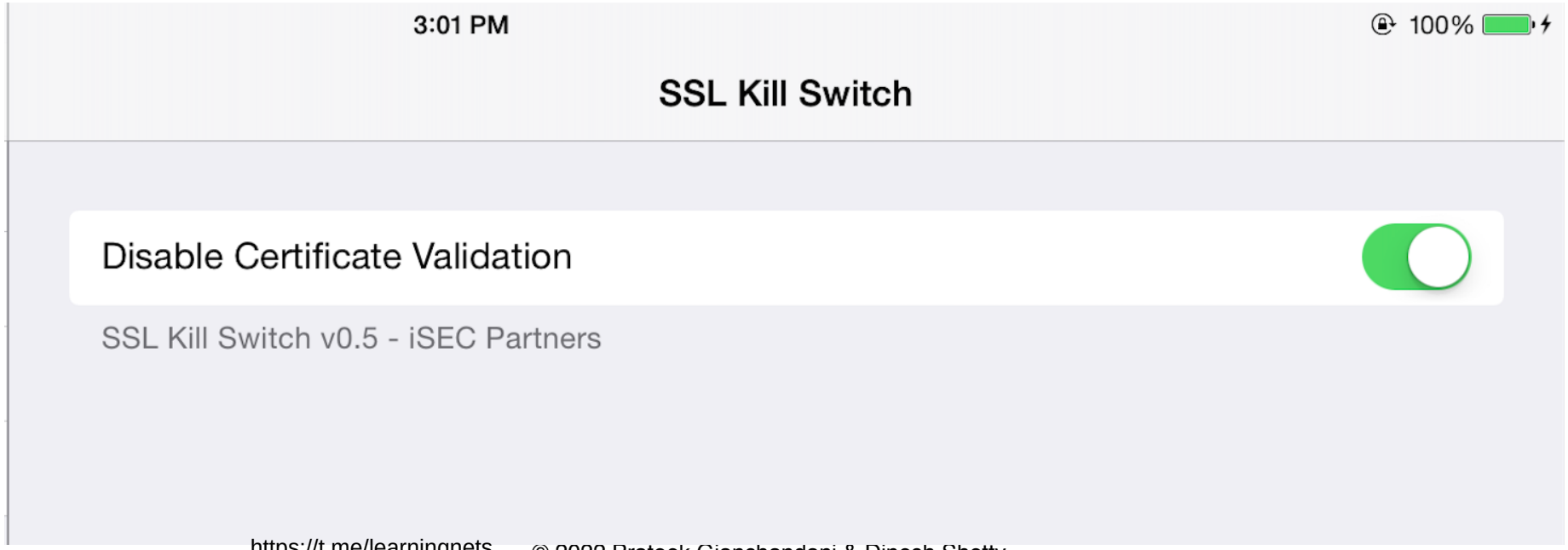


Certificate pinning

- The server's certificate is hardcoded in the application bundle and checked while exchanging data with the server.
- Provides protection against MITM attacks.
- Used by applications like Twitter, Square etc.

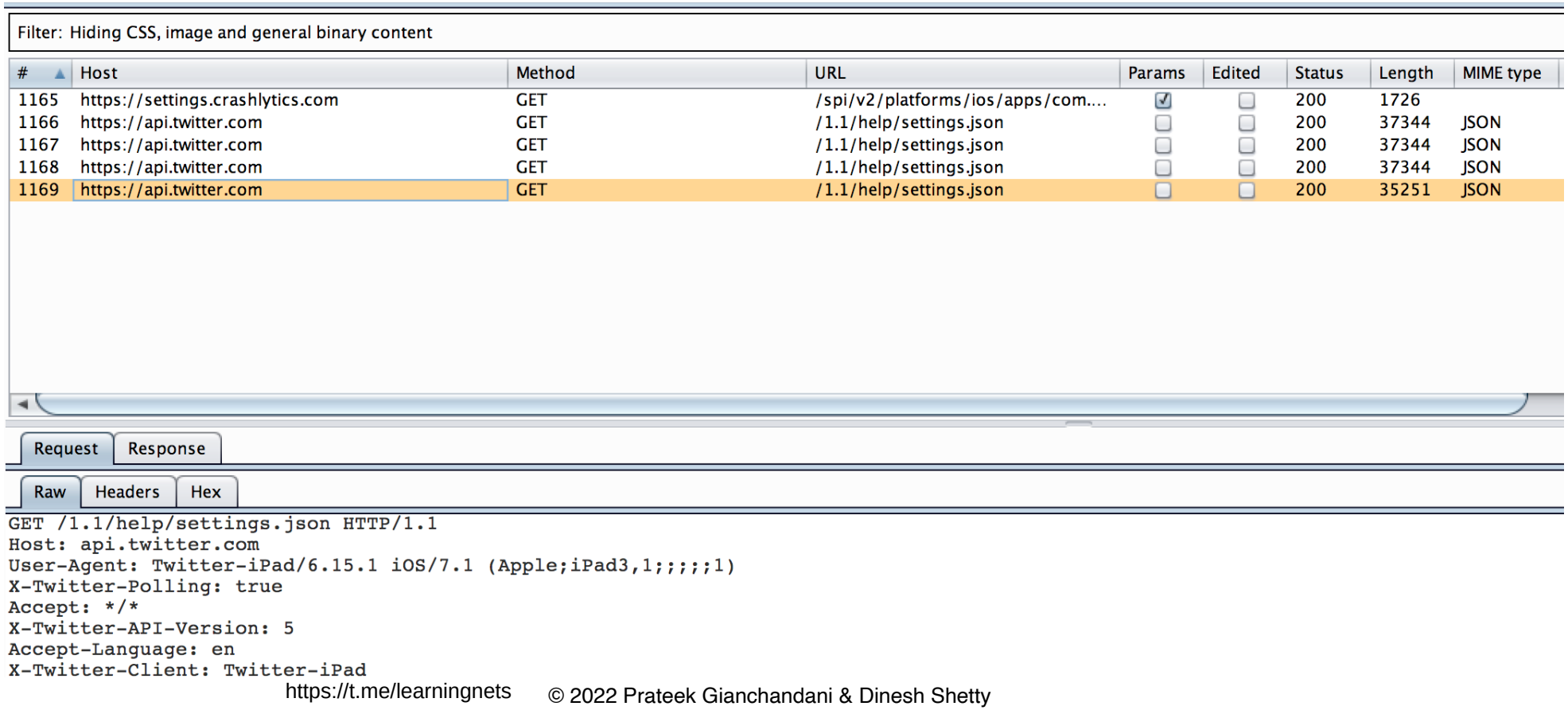
Certificate pinning

- Certificate pinning can be bypassed by hooking into some low level methods during runtime.
- iOS SSL kill switch was released in Blackhat to demonstrate this.
- https://github.com/nabla-c0d3/ssl-kill-switch2



Certificate pinning bypass

- Once it is enabled, user can see the traffic through applications like Twitter as well.



App Transport Security

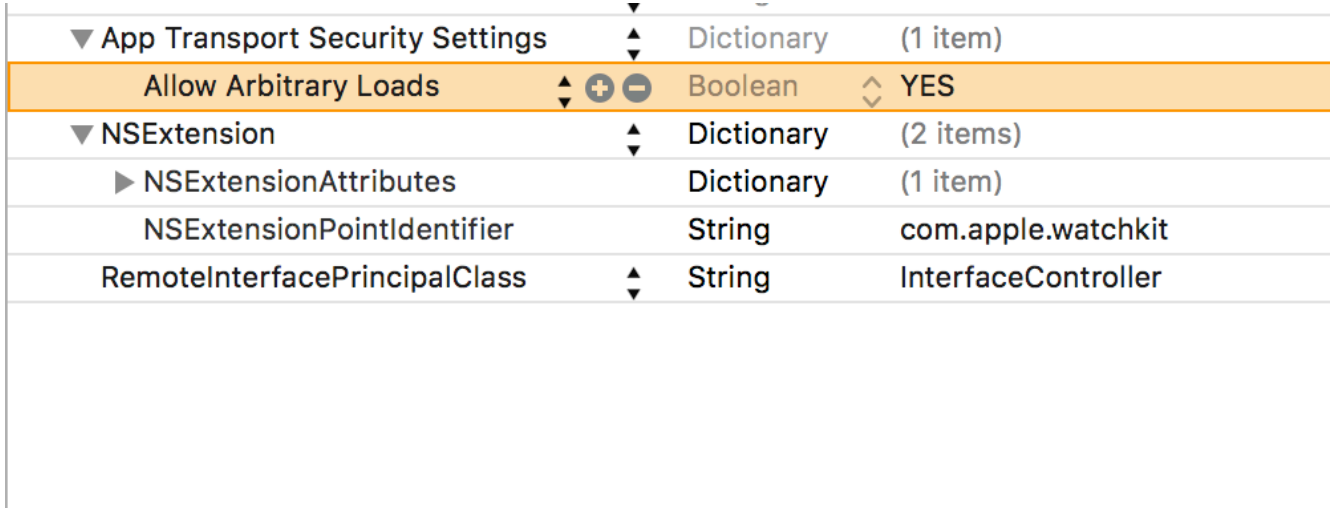
Requirements for Connecting Using ATS

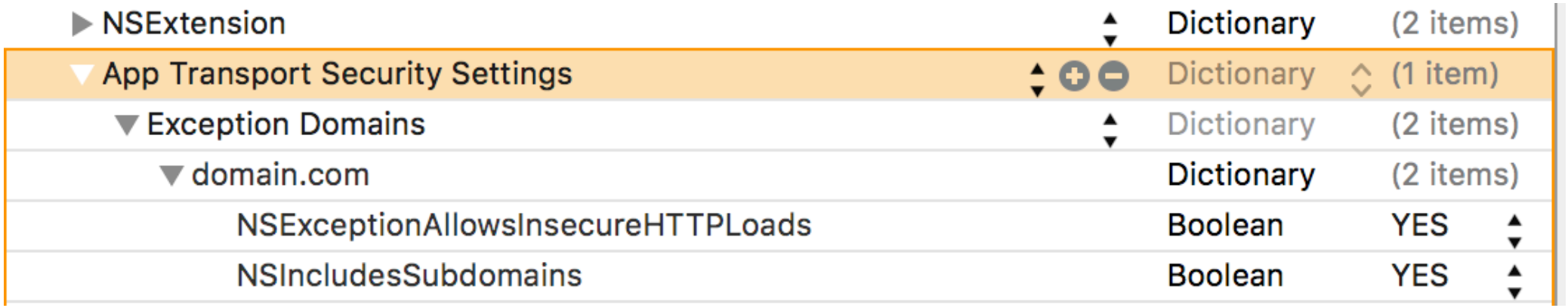
With ATS fully enabled, your app's HTTP connections must use HTTPS and must satisfy the following security requirements:

- The server certificate must meet at least one of the following trust requirements:
  - Issued by a certificate authority (CA) whose root certificate is incorporated into the operating system
  - Issued by a trusted root CA and installed by the user or a system administrator
- The negotiated Transport Layer Security version must be TLS 1.2
- The negotiated TLS connection cipher suite must support forward secrecy (FS) and be one of the following:
  - `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA`
  - `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA`
  - `TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`
  - `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA`
- The leaf server certificate must be signed with one of the following types of keys:
  - Rivest–Shamir–Adleman (RSA) key with a length of at least 2048 bits
  - Elliptic-Curve Cryptography (ECC) key with a size of at least 256 bits

  In addition, the leaf server certificate hashing algorithm must be Secure Hash Algorithm 2 (SHA-2) with a digest length of at least 256 (that is, SHA-256 or greater).

App Transport Security





Task - 15 mins

- Try to capture traffic over HTTP/HTTPs for the Facebook app or any app of your choosing
- Bypass SSL Pinning in the Twitter app
- Try and use some SSL pinning bypass frida scripts from codeshare https://codeshare.frida.re/browse

Flutter/Xamarin frameworks

- Cross platform frameworks for developing mobile apps
- Intercepting traffic in apps built with these frameworks is a bit challenging as they don’t respect system proxy
- Bypassing SSL pinning is more tricky
- To identify apps built these frameworks, Unzip the IPA and look inside the Frameworks folder for specific keywords like flutter and xamarin

```
prateek:: ls
Runner.app
prateek:: cd Runner.app/Frameworks
prateek:: ls
App.framework           Flutter.framework
prateek::
```

Method 1 - Hotspot method

- Setup hotspot from computer and connect mobile device to it
- In mobile device, set proxy as computer IP and set Invisible proxying in Burpsuite
- Create pf rules on computer to forward device data to burpsuite and enable Ip forwarding

rdr pass on bridge100 inet proto tcp from any to any -> 127.0.0.1 port 8080

sudo pfctl -f pf.rules

sudo sysctl -w net.inet.ip.forwarding=1

- You should be able to intercept the traffic now

Method 2 - reflutter framework

- https://github.com/ptswarm/reFlutter -> Flutter reverse engineering framework
- Test apps can be found at https://github.com/thedarksource/Android/tree/master/flutter-test-apps
- Patches the application

Key features:

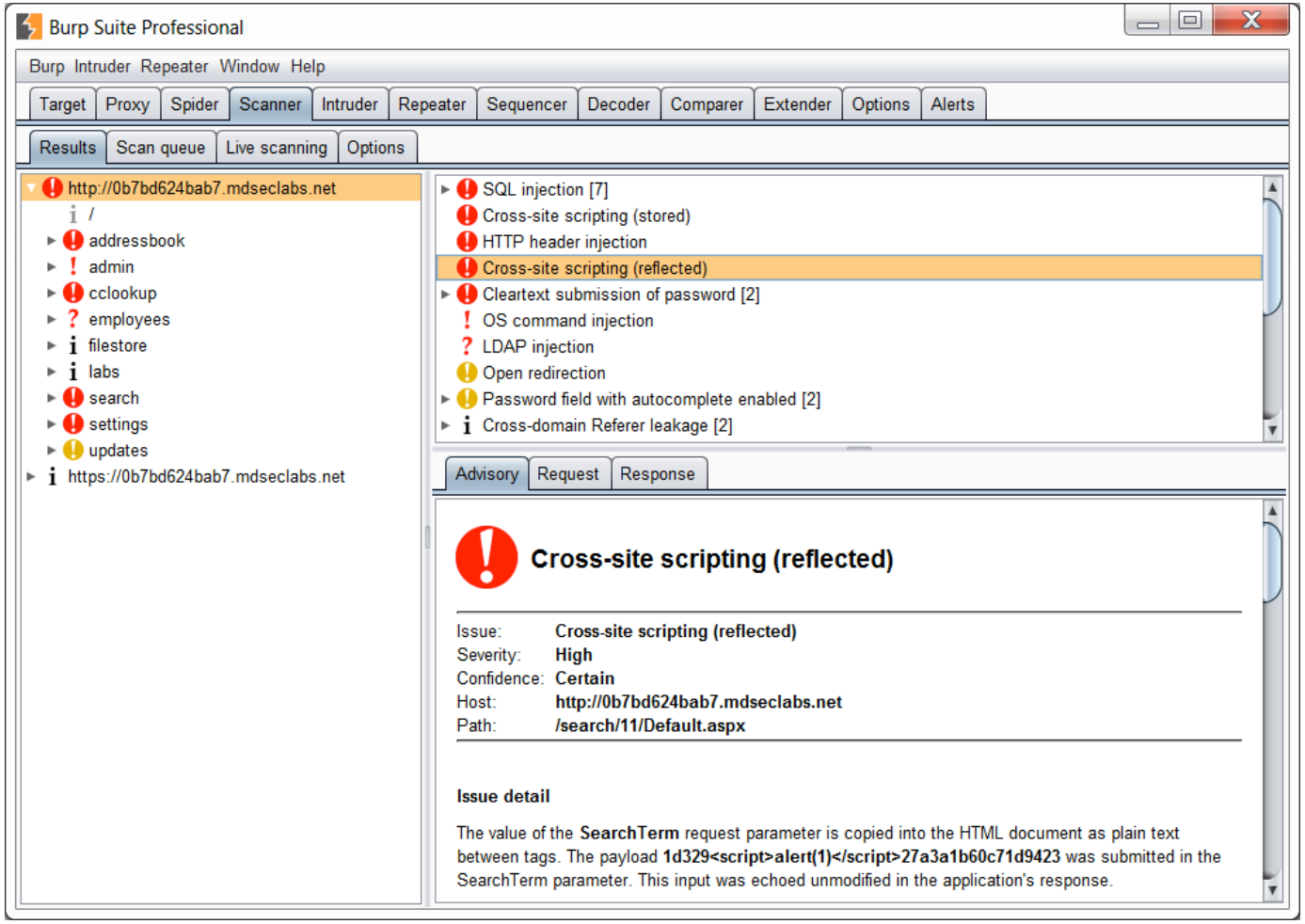
- `socket.cc` is patched for traffic monitoring and interception;
- `dart.cc` is modified to print classes, functions and some fields;
- contains minor changes for successfull compilation;
- if you would like to implement your own patches, there is manual Flutter code change is supported using specially crafted `Dockerfile`

Recommended Reading

- 8 ways of bypassing SSL pinning - https://www.appknox.com/blog/bypass-ssl-pinning-in-ios-app

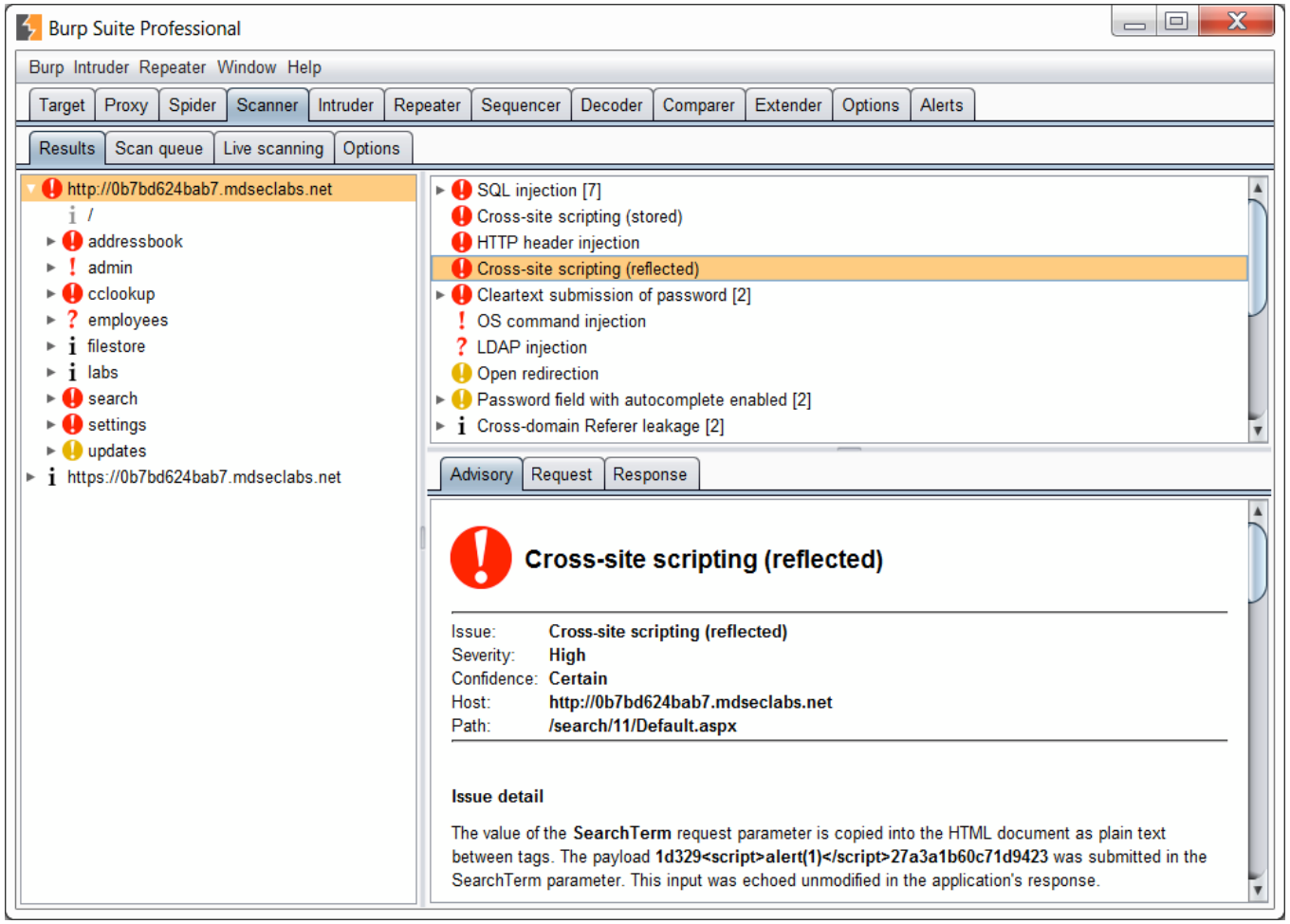
Scanning Server side vulnerabilities

- Same logic as testing for web application vulnerabilities
- Includes both manual and automated testing



Scanning Server side vulnerabilities

- Perform a scan on the vulnerable web application via OWASP Zap or Burpsuite.



WebViews

- WebViews are used to display web content
- By default, Javascript is enabled
- Attacks like Cross-Site scripting are possible
- Some popular classes are **UIWebView**, **WkWebView** and **SFSafariViewController**
- UIWebView has been deprecated and not recommended to be used
- Javascript can’t be disabled in **UIWebView**

WKWebView security advantages

- Javascript can be turned off by using the **javaScriptEnabled** property
- The **JavaScriptCanOpenWindowsAutomatically** property can be used to disable new windows such as popups
- The hasOnlySecureContent ensures resources loaded are through secure channels.
- WKWebView implements out-of-process rendering. Memory corruption bugs like **JIT optimisation bugs** such as **UaF** etc will not affect the main app process.

UIWebView vs WKWebView

UIWebView has this issue due to two properties — `WebKitAllowUniversalAccessFromFileURLs` and `WebKitAllowFileAccessFromFileURLs` are by default turned on. This allows UIWebView to access file with its url. We can turn it off manually to remove the danger.

On the other hand, we do not need to worry about WKWiebView, as those mentioned properties are by default turned off.

Finally, here are tips to avoid security flaws in your app:

- Keep an eye on any html files adopted by app to load something
- Pay attention to the html file if it touches files on device
- Do not use UIWebView
- Please do not turn on `WebKitAllowUniversalAccessFromFileURLs` or `WebKitAllowFileAccessFromFileURLs` while using WKWebView.

https://medium.com/ios-os-x-development/security-flaw-with-uiwebview-95bbd8508e3c

SfSafariViewController

- Isolated from the app context itself
- Shares cookies and other website data with Safari
- Recommended if you want to open random links inside the app, but without any interactions with the link and their content itself

Source: https://developer.apple.com/documentation/safariservices/sfsafariviewcontroller

Testing for WebViews

- **grep** for symbols starting with UIWebView, WKWebView etc
- SFSafariViewcontroller doesn’t have any security implications
- Make sure the javascript settings and configurations are properly enabled, in some cases having javascript enabled would be a requirement, in other cases not.

Deep Linking: URL schemes vs Universal links

- https://medium.com/wolox/ios-deep-linking-url-scheme-vs-universal-links-50abd3802f97
- http://highaltitudehacks.com/2014/03/07/ios-application-security-part-30-attacking-url-schemes/
- https://www.mobileiron.com/en/blog/ios-url-scheme-hijacking-xara-attack-analysis-and-countermeasures
- https://developer.apple.com/documentation/xcode/allowing_apps_and_websites_to_link_to_your_content/supporting_universal_links_in_your_app

> **Warning**
>
> Universal links offer a potential attack vector into your app, so make sure to validate all URL parameters and discard any malformed URLs. In addition, limit the available actions to those that don't risk the user's data. For example, don't allow universal links to directly delete content or access sensitive information about the user. When testing your URL-handling code, make sure your test cases include improperly formatted URLs.

Jailbreak Detection

- For critical applications like banking applications etc, it is important that you ensure that the application doesn't work on a jailbroken device.
- With a copy of your app's binary and tools like Cycript at his disposal, an attacker is in complete control.
- It is therefore important to check for a jailbroken device and disable certain features of the application or quit the application in order to protect it.

Jailbreak Detection

- There are many ways to check for a jailbroken device.
- Checking for specific files that exist on a jailbroken device is one of the most common techniques being used.
- Another way is to check if the application is able to modify a file outside it's own sandbox.
- Most than 80% of the jailbroken devices have Cydia installed, so check if you can open a url that starts with Cydia's URL scheme, i.e cydia://
- Using C code, we can check for open ports, fork command etc
- It is important to note that no that there is no foolproof technique to detect a jailbroken device, however a combination of checks can make the job difficult for even a skilled hacker.

Jailbreak Detection

- Combining all these techniques, we get this method.

```
+(BOOL)isJailbroken{

  #if !(TARGET_IPHONE_SIMULATOR)

    if ([[NSFileManager defaultManager] fileExistsAtPath:@"/Applications/Cydia.app"]){
      return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/Library/MobileSubstrate/MobileSubstrate.dylib"]){
      return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/bin/bash"]){
      return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/usr/sbin/sshd"]){
      return YES;
    }else if([[NSFileManager defaultManager] fileExistsAtPath:@"/etc/apt"]){
      return YES;
    }

   NSError *error;
    NSString *stringToBeWritten = @"This is a test.";
    [stringToBeWritten writeToFile:@"/private/jailbreak.txt" atomically:YES
        encoding:NSUTF8StringEncoding error:&error];
    if(error==nil){
    //Device is jailbroken
    return YES;
   } else {
    [[NSFileManager defaultManager] removeItemAtPath:@"/private/jailbreak.txt" error:nil];
   }

   if([[UIApplication sharedApplication] canOpenURL:[NSURL URLWithString:@"cydia://package/com.example.package"]]){
    //Device is jailbroken
    return YES;
   }
  #endif

   //All checks have failed. Most probably, the device is not jailbroken
   return NO;
}
```

Jailbreak Detection

- The problem is that the signature of this method gives everything away.
- Attacker can use Frida to use bypass the check for jailbreak detection.

Swift 4 JB Detection

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

Jailbreak Detection

- It is better to rename the method to something that doesn't look important.
- Something like **+(BOOL)isDefaultColour**
- Yeah I know, we do ignore the coding guidelines, but in this case, the guidelines are something that gives everything away.
- After analyzing the class-dump output of the application, the hacker is most likely to ignore this method.
- He can always reverse engineer this method to see what's going on inside, so this method is also not foolproof.

Broken Cryptography

- Occurs when data stored on the device is not encrypted properly.
- One of the most common vulnerabilities found in iOS applications.
- Can occur by use of deprecated or weak algorithms.
- Sometimes the key while encryption is hardcoded in the app thereby making it much easier for the attacker to break the application.
- Related article: http://www.andreas-kurtz.de/2013/07/how-to-easily-spot-broken-cryptography.html

Task Broken Cryptography - 40 mins

- Open the Broken Cryptography challenge in **DVIA** and set a password, use reversing and Frida in the application to find out the following

  a) Which encryption library is being used ?

  b) And Which function is responsible for encryption ?

  c) Which encryption algorithm is being used ?

  d) Where is the encryption key being stored ?

  e) Which file stores the encrypted data ?

  f) How will you decrypt the encrypted data ?

  Post Solution in the Slack Channel #Labs

Code obfuscation

- Difficult to do because of the way iOS application are compiled and executed.
- For hiding class information - iOS class guard https://github.com/Polidea/ios-class-guard
- For sqlite databases - SQLCipher - https://www.zetetic.net/sqlcipher/
- For many other things - iMAS - https://github.com/project-imas/memory-security/

Code obfuscation

- Add bogus code
- Changes names of important classes, methods
- Use string encryption. Encrypt the key as well
- Generate the key at runtime, for e.g by adding 2 different strings

Patching iOS applications

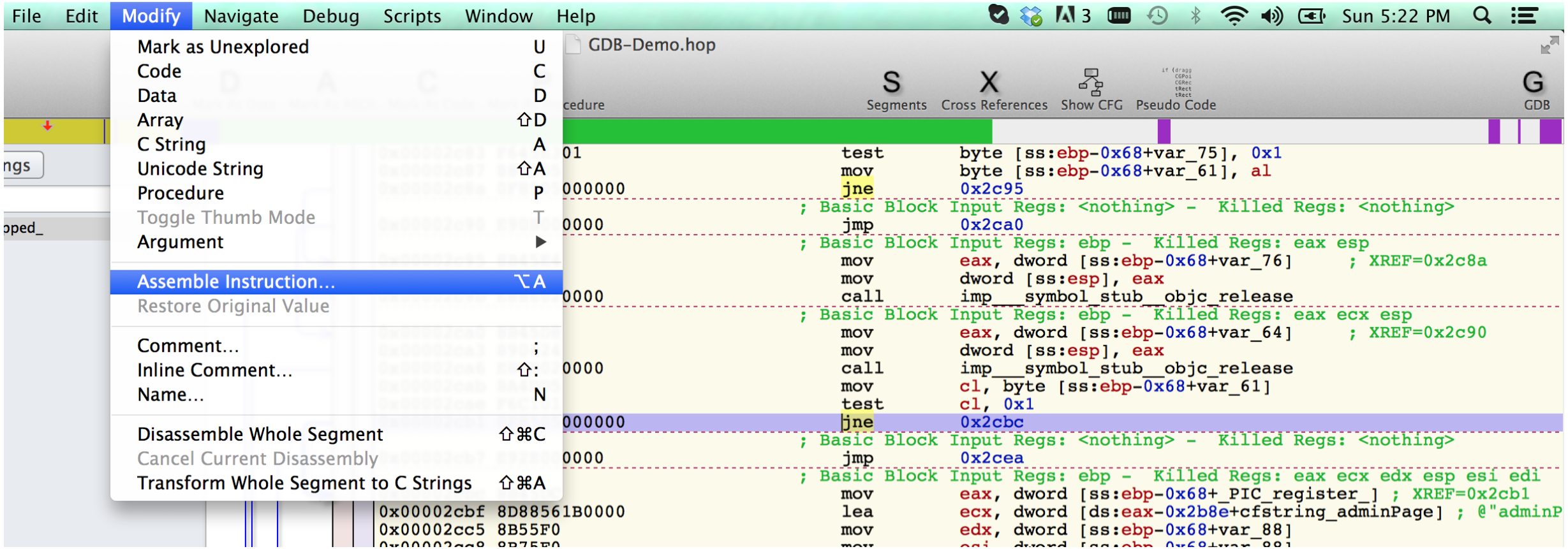
- Patching an application changes its login permanently.
- This is better that making a change in cycript where you have to repeat the same process over and over again.
- Often used to disable checks like Jailbreak detection, piracy check etc.
- Tools used for patching iOS application: radare, IDA Pro, Hexfiend and Hopper.

Patching iOS applications

- Hopper is one of the best tools available for patching iOS applications.
- Not free, but the value for money is very good.
- Patching iOS applications with Hopper: http://highaltitudehacks.com/2014/01/17/ios-application-security-part-28-patching-ios-application-with-hopper/
- Patching iOS applications with IDA Pro and Hex fiend:http://highaltitudehacks.com/2013/12/17/ios-application-security-part-26-patching-ios-applications-using-ida-pro-and-hex-fiend

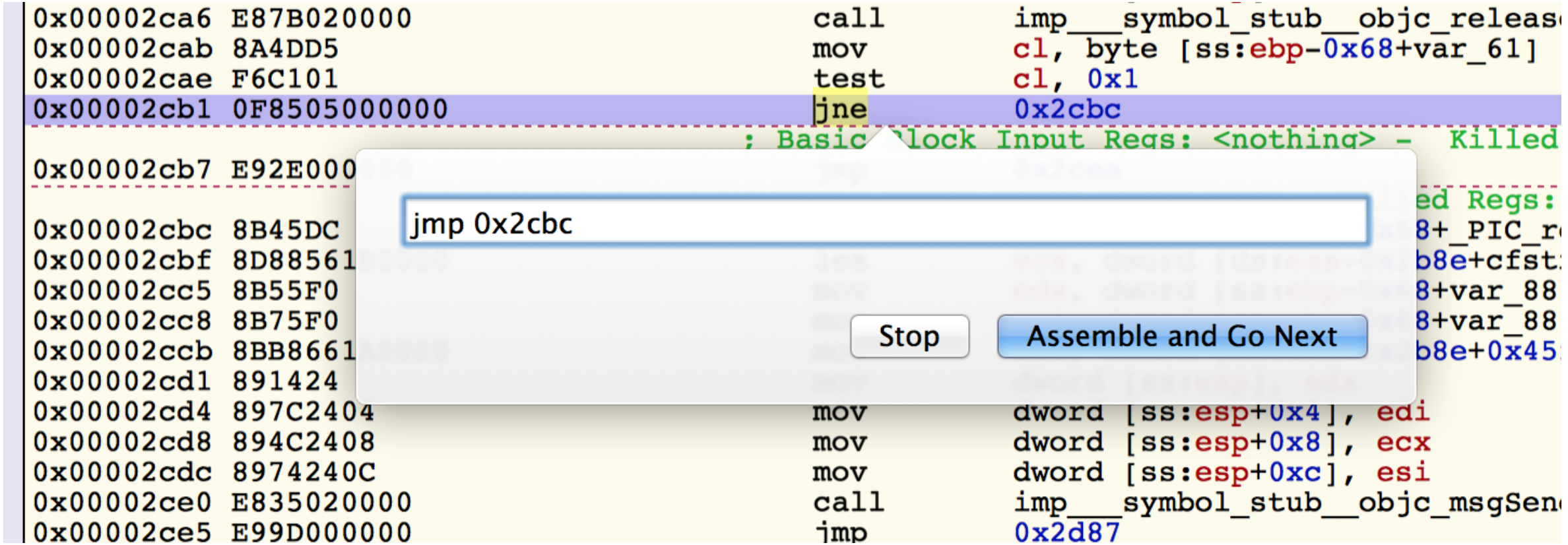
Patching iOS applications

- To modify any instruction in Hopper, click on it and click on Modify -> Assemble Instruction.



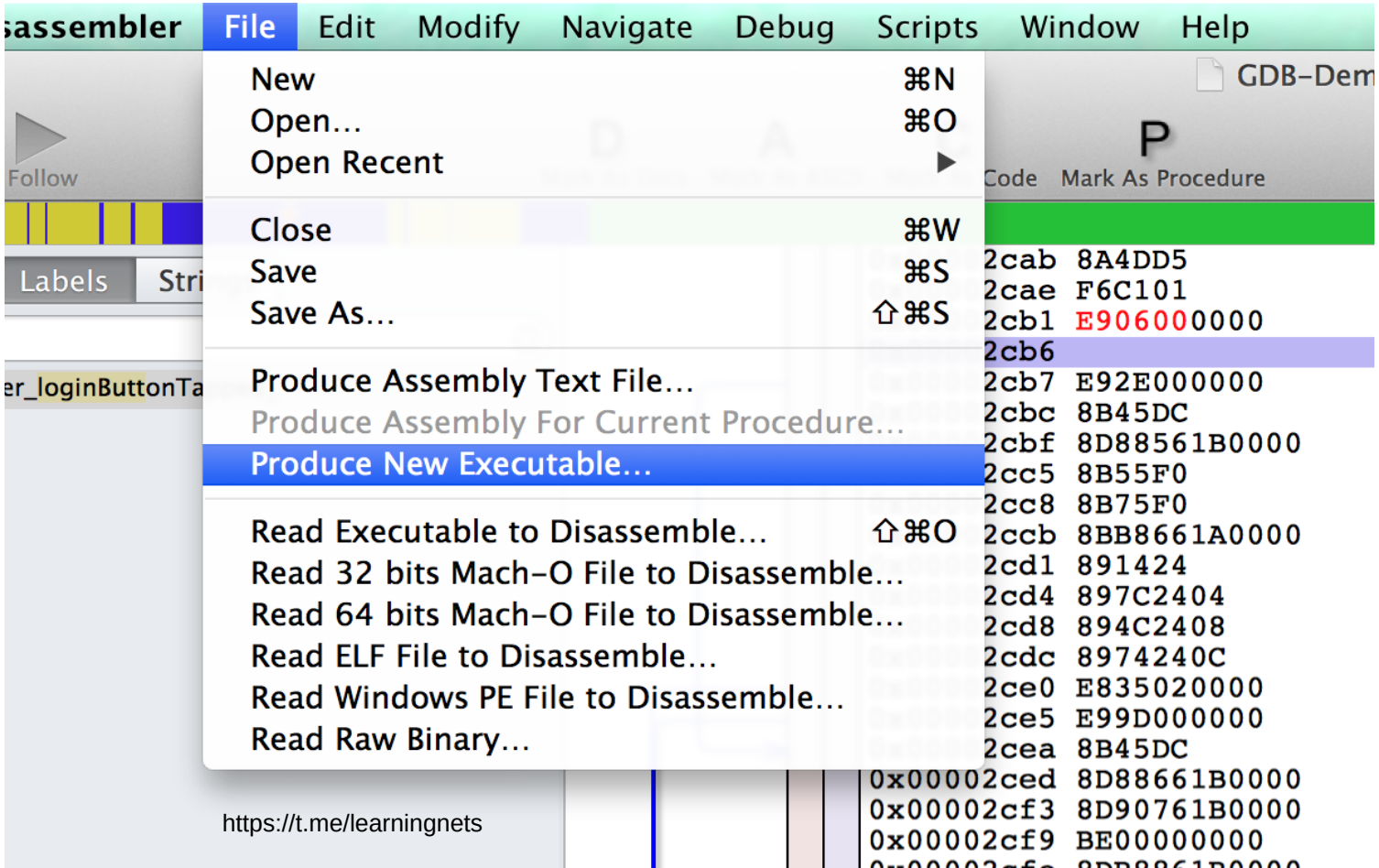
Patching iOS applications

- Make the change and click on *Assemble and Go Next*



Patching iOS applications

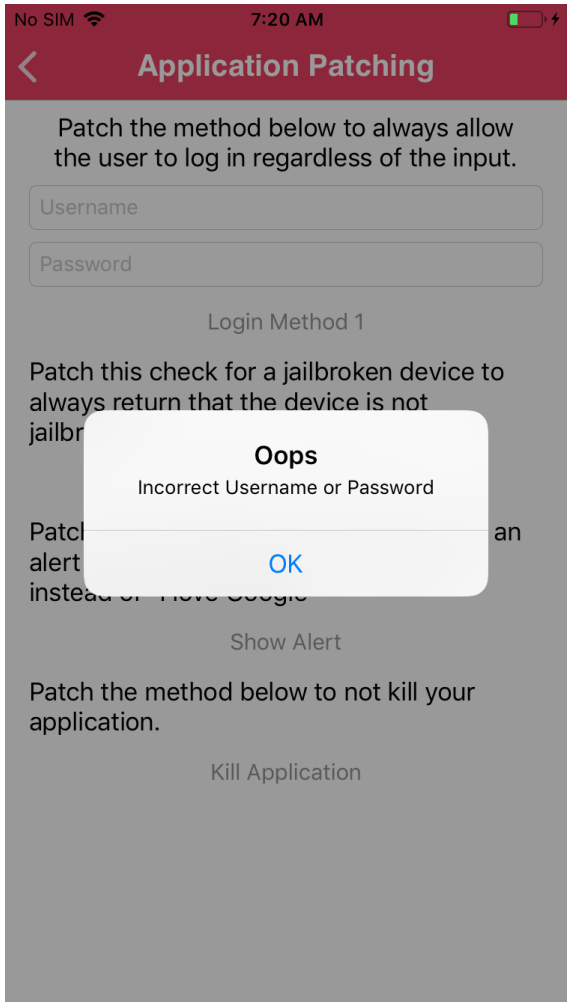
- Once the change has been made, click on File -> Produce new executable and overwrite the existing one.



Patching iOS applications

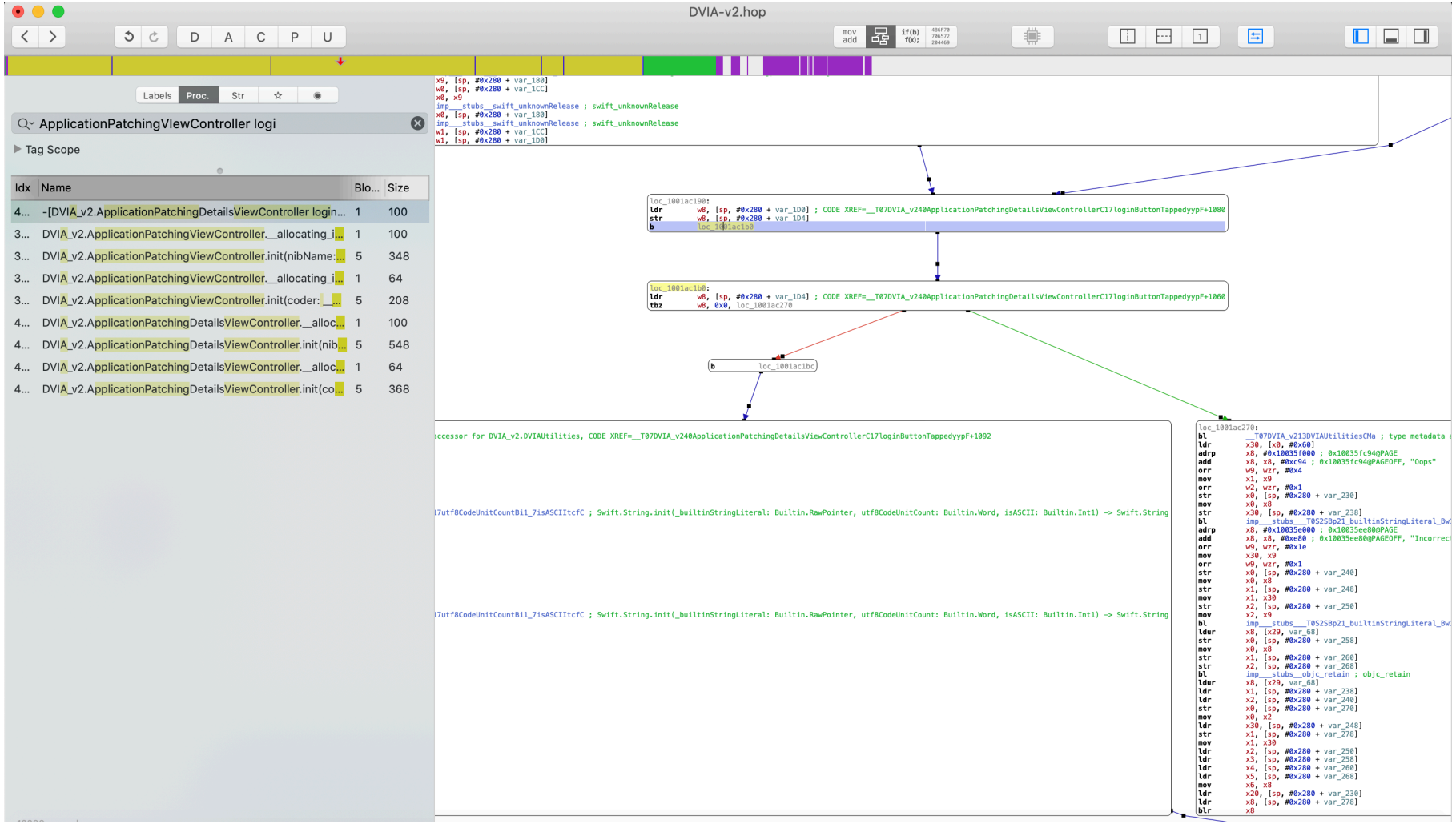
- To deploy the application back to your device, resign the application binary using Impactor and deploy it to the device
- Let's use **r2** to patch an application since its free
- Let's solve the first challenge in **"Application patching"** section in **DVIA-v2**

Challenge - DVIA-v2 Application Patching ->Login method 1



Hopper view

- Replace the instruction with ***b loc_1001ac1bc***

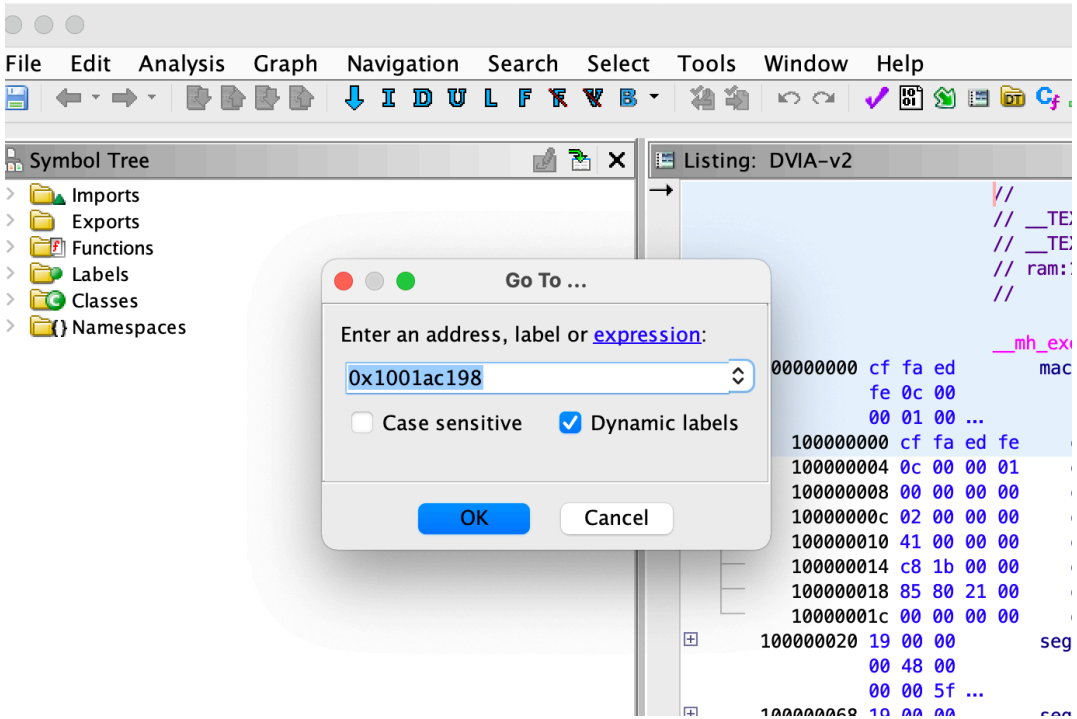


Deploying apps back onto the device

- To deploy the application back to your device, resign the application binary using Impactor
- Let’s patch the same instruction using **r2** and **Ghidra** since they are free
- **Since we are using Corellium , we don’t need to sign the application before installing, and we can use the Manual method discussed earlier today to install the application back on to the device**

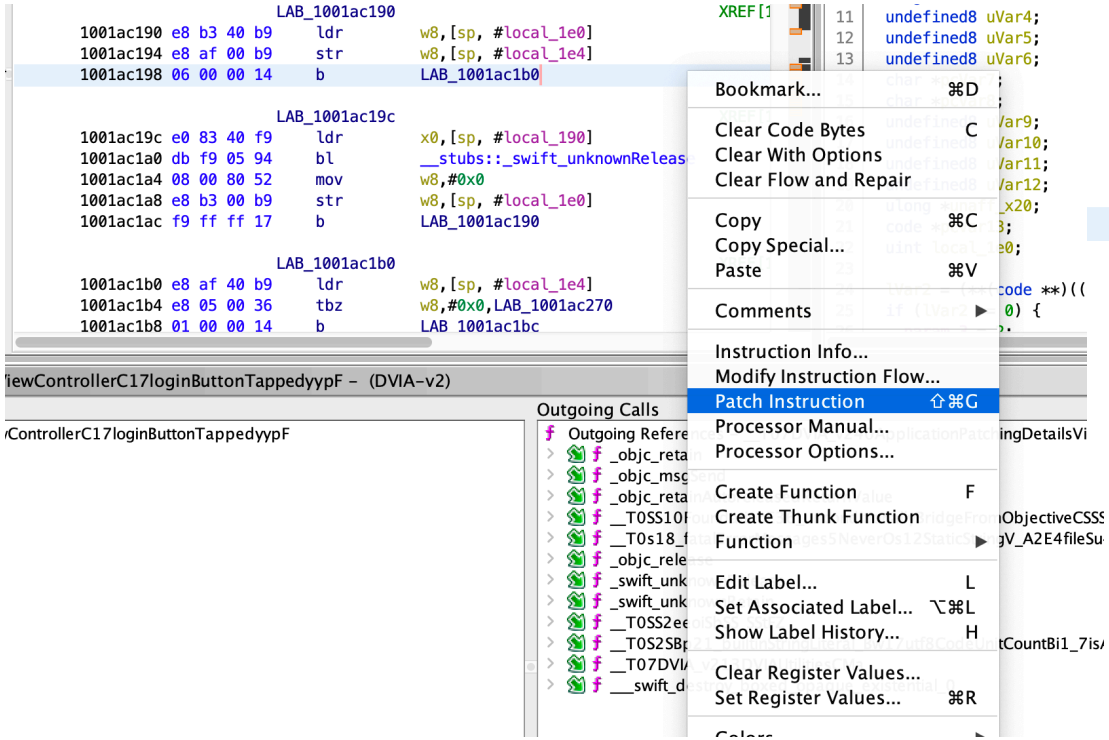
Steps using Ghidra

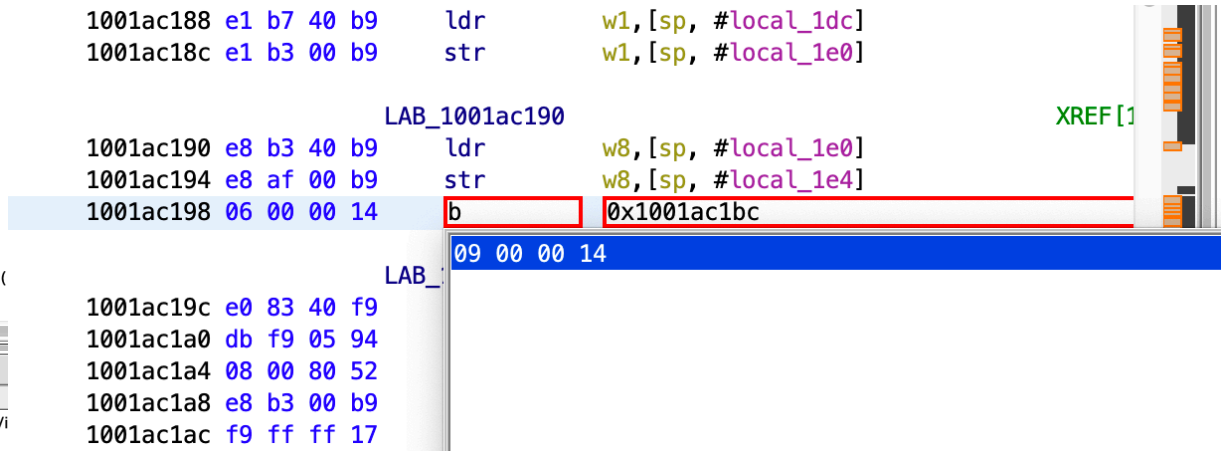
- Unzip **DVIA-v2** to **DVIA-v2.app** and open the **DVIA-v2** binary using Ghidra
- Click on **Navigate -> Go To** and Put the address **0x1001ac198** and Click **Ok**



Steps using Ghidra

- Right click on the instruction and Click on **Patch Instruction**
- Enter the new instruction **b 0x1001ac1bc**





Steps using Ghidra

- Click on **File -> Export File** and Choose the Format as **Binary**
- Rename the file from **DVIA-v2.bin** to **DVIA-v2**

  Now use the **manual method** to install the app by moving the **.app** folder to /**Applications** and running the **uicache** command (Refer to Sideloading apps section discussed earlier today)

  The app should run and the login check should show **Success** now

Steps using r2

- Unzip DVIA-v2 from the IPA folder **cd Desktop/IPA**
- Find the application binary and run the following commands

r2 -Aw DVIA-v2 - Open with write mode and analyze (takes some time)

i - Show info of current file

iz - Print strings in the data section

izz - Print strings in the whole binary

s main - Seek to the main function

pdf - Print disassembled version of the main function

Steps using r2

is - Show all the sections

s 0x1001ac198 - Jump to the instruction to be changed

pdf - Print disassembled function

wa b 0x1001ac1bc - Write the following opcode in that specific address

q - Quit

Now use the **manual method** to install the app by moving the **.app** folder to /**Applications** and running the **uicache** command (Refer to Sideloading apps section discussed earlier today)

The app should run and the login check should show **Success** now

Task - Patching apps

Patch the DVIA-v2 app using the steps mentioned in the previous slides

App Patching Fraud

- Attackers sometimes deploy multiple instances of the same app onto the device to conduct fraudulent attacks by modifying the app id of the app
- For e.g replacing **com.abcd.def** with **com.abcd.dee**
- Adding a simple check on your app for the app id can prevent script kiddies from modifying the app id and installing a new version on the device

```
func application(_ application: UIApplication,
    didFinishLaunchingWithOptions launchOptions:
    [UIApplication.LaunchOptionsKey: Any]?) -> Bool {
      // Override point for customization after application
          launch.
      let bundleIdentifier =  Bundle.main.bundleIdentifier
      // Generate string during runtime
      let a = "com.sbe"
      let b = "scaper.d"
      let c = "emo.Harde"
      let d = "ned-Hello"
      let e = "Worle"
      let x = a + b + c + d + e
      //Check if Application has been patched and installed as
          another package name (check multiple instances of the
          app
      if bundleIdentifier != x
      {
          exit(0)
      }
      return true
  }
```

Testing Apps built with other frameworks

Most of the logic for testing would be the same

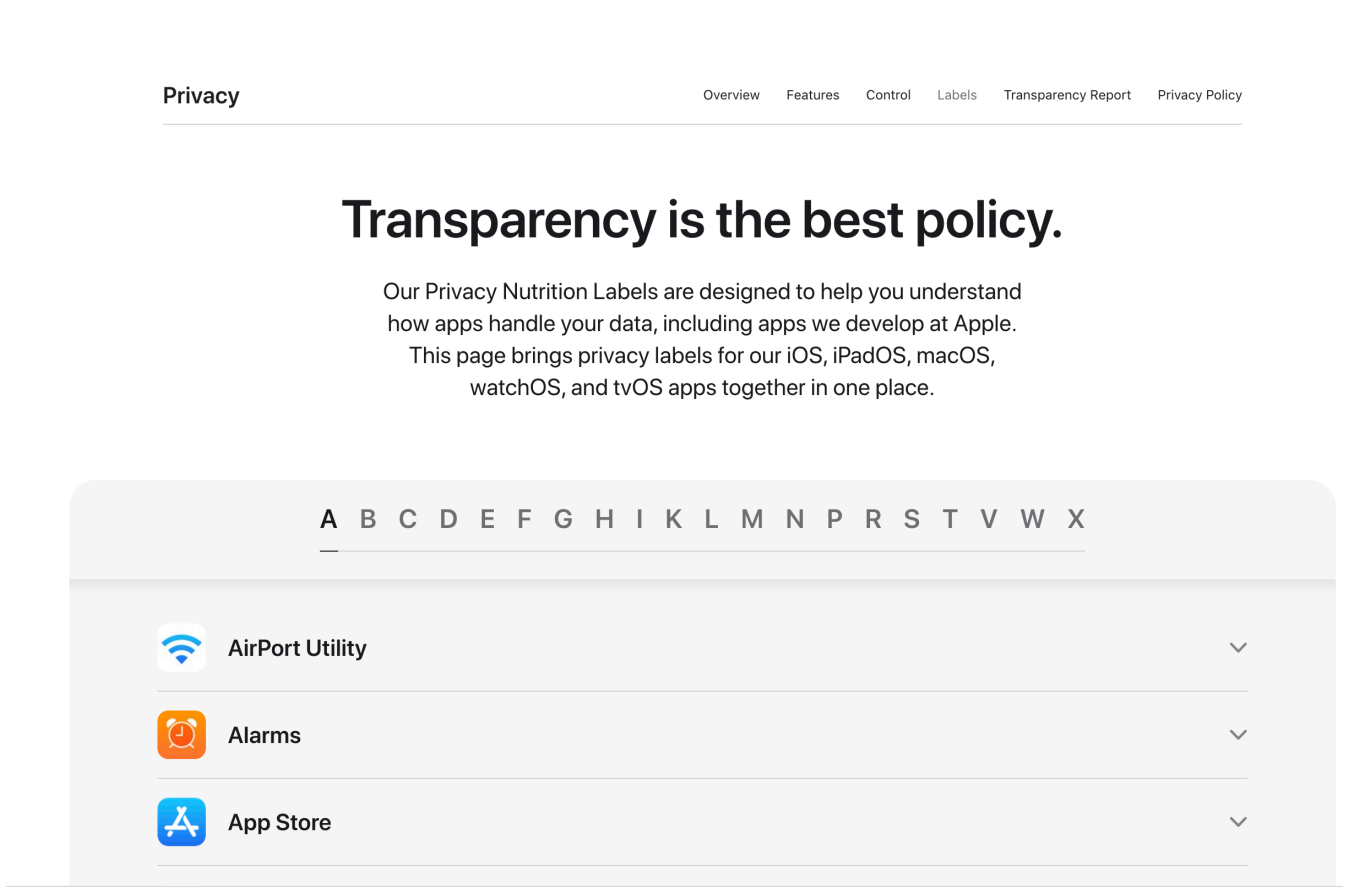
For JS Frameworks, most of the code would lie in some packaged JS File,

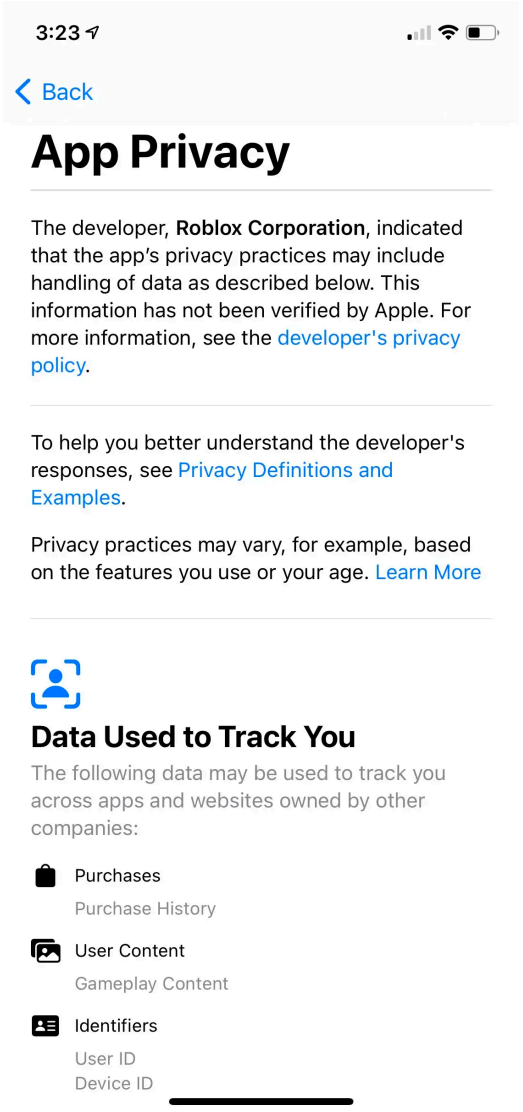
for e.g **main.jsbundle**

You can use tools like https://mindedsecurity.github.io/jstillery/ to analyze

these JS files and look for any sensitive info

Apple nutrition label



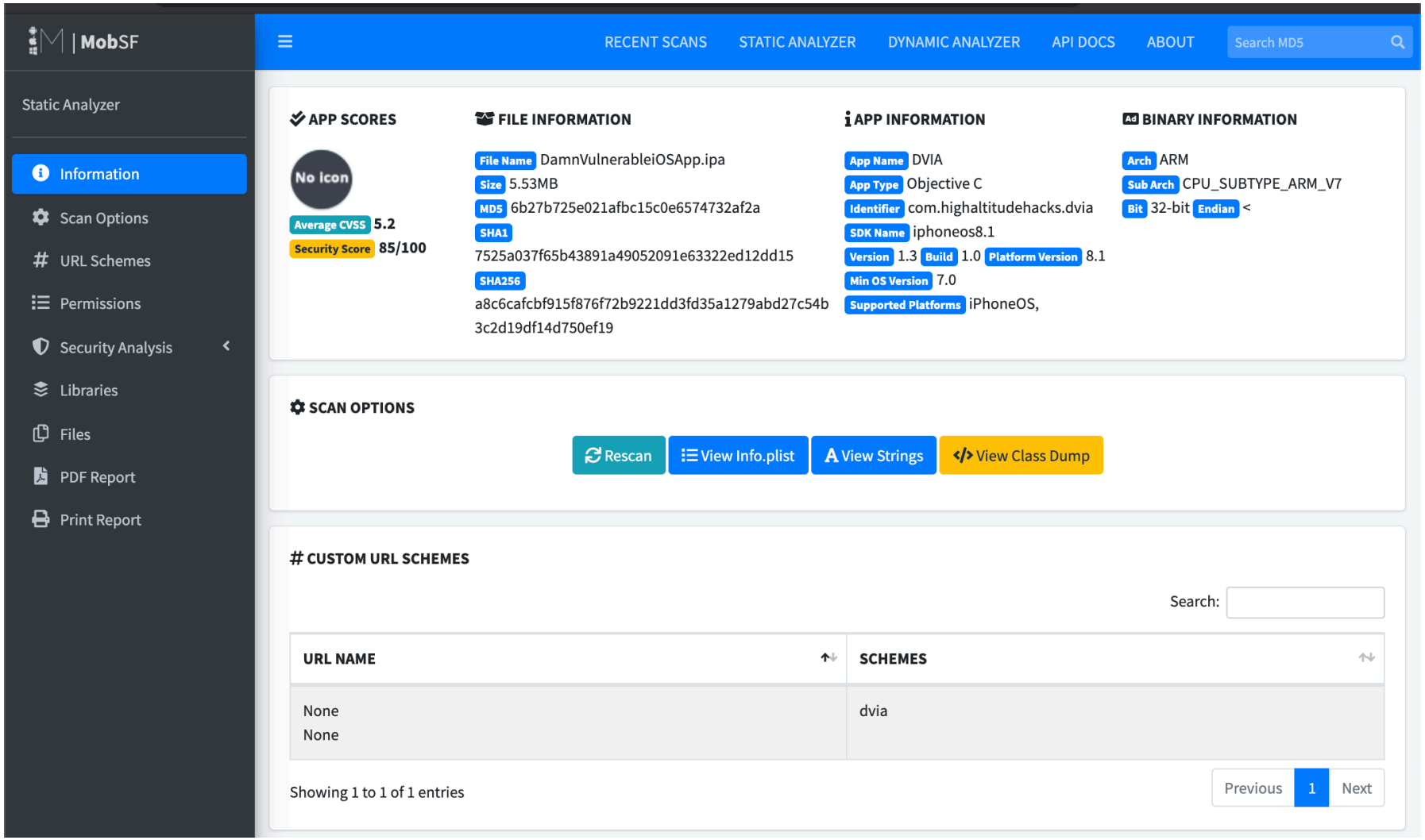


https://www.businessinsider.com/what-are-apple-privacy-nutrition-labels

Automated testing

- Automating tests while doing an iOS penetration test can help you save a lot of time.
- Though not all tests can be automated, there are some tools that do a very good job at this.
- **Objection** - https://github.com/sensepost/objection
- **Needle** - https://github.com/mwrlabs/needle
- **PassionFruit** - https://github.com/chaitin/passionfruit
- **MobSF** - https://github.com/MobSF/Mobile-Security-Framework-MobSF

MobSF



Source: https://github.com/MobSF/Mobile-Security-Framework-MobSF

Objection

```
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] # file download userInfo.plist
Downloading /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Documents/use
erInfo.plist
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] # !cat userInfo.plist
Running OS command: cat userInfo.plist

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>password</key>
        <string>hehe</string>
        <key>username</key>
        <string>hand</string>
</dict>
</plist>

...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] #
```

Objection

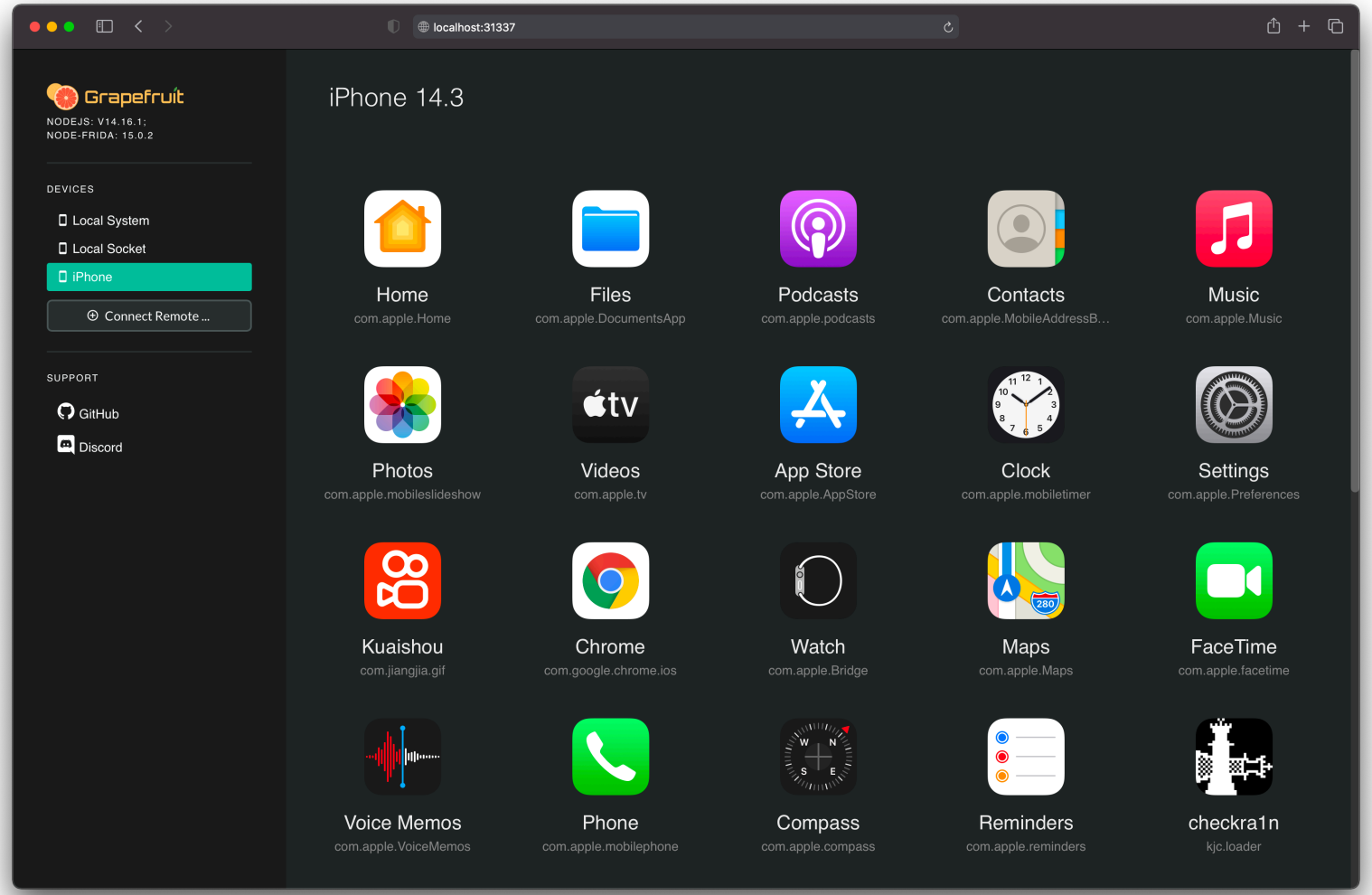
```
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] # ios keychain dump
Note: You may be asked to authenticate using the devices passcode or TouchID
Get all of the attributes by adding `--json keychain.json` to this command
Reading the iOS keychain...

Class                    Account        Service                              Generic    Data
-----------------------  -------------  -----------------------------------  ---------  ------
kSecClassGenericPassword  keychainValue  com.highaltitudehacks.DVIA-v2tester             ivjff
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] #
```

```
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] # env

Name                          Path
----------------------------  ------------------------------------------------------------------------------------------
-------------------
DocumentDirectory             /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Documents
LibraryDirectory              /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Library
CachesDirectory               /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Library/Cac
hes
BundlePath                    /var/containers/Bundle/Application/9F8A2779-8149-4834-AF32-72FCA8156400/DVIA-v2.app
ApplicationDirectory          /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Application
s
DemoApplicationDirectory      /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Application
s/Demos
DeveloperApplicationDirectory  /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Developer/A
pplications
UserDirectory
CoreServiceDirectory
AutosavedInformationDirectory  /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Library/Aut
osave Information
DesktopDirectory              /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Desktop
ApplicationSupportDirectory   /var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Library/App
lication Support
ReceiptPath                   /private/var/mobile/Containers/Data/Application/767F1D5C-F8D4-4A40-9421-5DDDC88C55E3/Sto
reKit/sandboxReceipt
ResourcePath                  /var/containers/Bundle/Application/9F8A2779-8149-4834-AF32-72FCA8156400/DVIA-v2.app
...ighaltitudehacks.DVIA-v2tester on (iPhone: 11.3.1) [usb] #
```

Needle

```
[needle] > use dynamic/detection/jailbreak_detection
[needle][jailbreak_detection] > info

      Name: Jailbreak Detection
      Path: modules/dynamic/detection/jailbreak_detection.py
    Author: @LanciniMarco (@MWRLabs)

Description:
  Verify that the app cannot be run on a jailbroken device. Currently detects if
 the app applies
  jailbreak detection at startup.

Comments:
  * Make sure that the device is unlocked before you run this module

Options:
  No options available for this module.

[needle][jailbreak_detection] >
```

GrapeFruit



https://github.com/ChiChou/grapefruit

Static Analysis of iOS applications

Sr. No.	Test cases	Analysis Logic
1	Does the application leak sensitive information via device memory?	Check for presence of NSFile, writeToFile in Source Code
2	Can the application leak sensitive information due to iOS default Screencapture feature?	Check for the presence of window.hidden in applicationWillEnterBackground and applicationWillTerminate functions in Source code.
3	Does the application leak sensitive information via hardcoded secrets?	Check for presence of // and /* */ in Source code

Sr. No.	Test cases	Analysis Logic
4	Is the application vulnerable to buffer overflow attack?	Check for the presence of strcat, strcpy, strncat, strncpy, sprintf, vsprintf, gets in the Source code
5	Can malicious activties be performed due to insecure implementation of URL Schemes?	Check for the presence of presence of Authorisation in functions having openUrl, handleOpenURL.
6	Does the application leak sensitive information viaSQLite db?	Check for presence of db, sqlite, database, insert, delete, select, table, cursor, sqlite3_prepare, sqlite3_compile in Source code.
7	Does the application leak sensitive information due to insecure Logging mechanism?	Check for presence of NSLog, Xlog, ZNLog in Source code.

Sr. No.	Test cases	Analysis Logic
8	Is critical data of the application encrypted using proper control?	Check for presence of MD5, base64, des in Source code.
9	Safe keychain mechanism?	Check for presence of kSecASttr, SFHFKkey in Source code.
10	Debugging status	Check for the presence of #ifdef DEBUG

Sr. No.	Test cases	Analysis Logic
11	Does the application implement a insecure transport mechanism?	Check for presence of http://, URL, setAllowsAnyHTTPSCertificate, NSURL,writeToUrl, NSURLConnection, CFStream, NSStreamin Source code. Also check for presence of redirection to http in via didFailWithError in the Source code.
12	Does the application misuse or leaksensitive information like device identifiers or via a side channel?	Check for the presence of uid, user-id, imei, deviceId, deviceSerialNumber, devicePrint, X-DSN, phone, mdn, did, IMSI, uuid in Source code
13	Does the application misuse or leaksensitive information like Location Info or via a side channel?	Check for the presence of CLLocationManager, startUpdatingLocation, locationManager, didUpdateToLocation, CLLocationDegrees, CLLocation, CLLocationDistance, startMonitoringSignificantLocationChanges, LOCATION in Source code

Further practice

Try out all the challenges in Damn Vulnerable iOS App v2 (http://damnvulnerableiosapp.com)