Presentation On iOS Pentesting By Sunil

Suma Soft Pvt Ltd

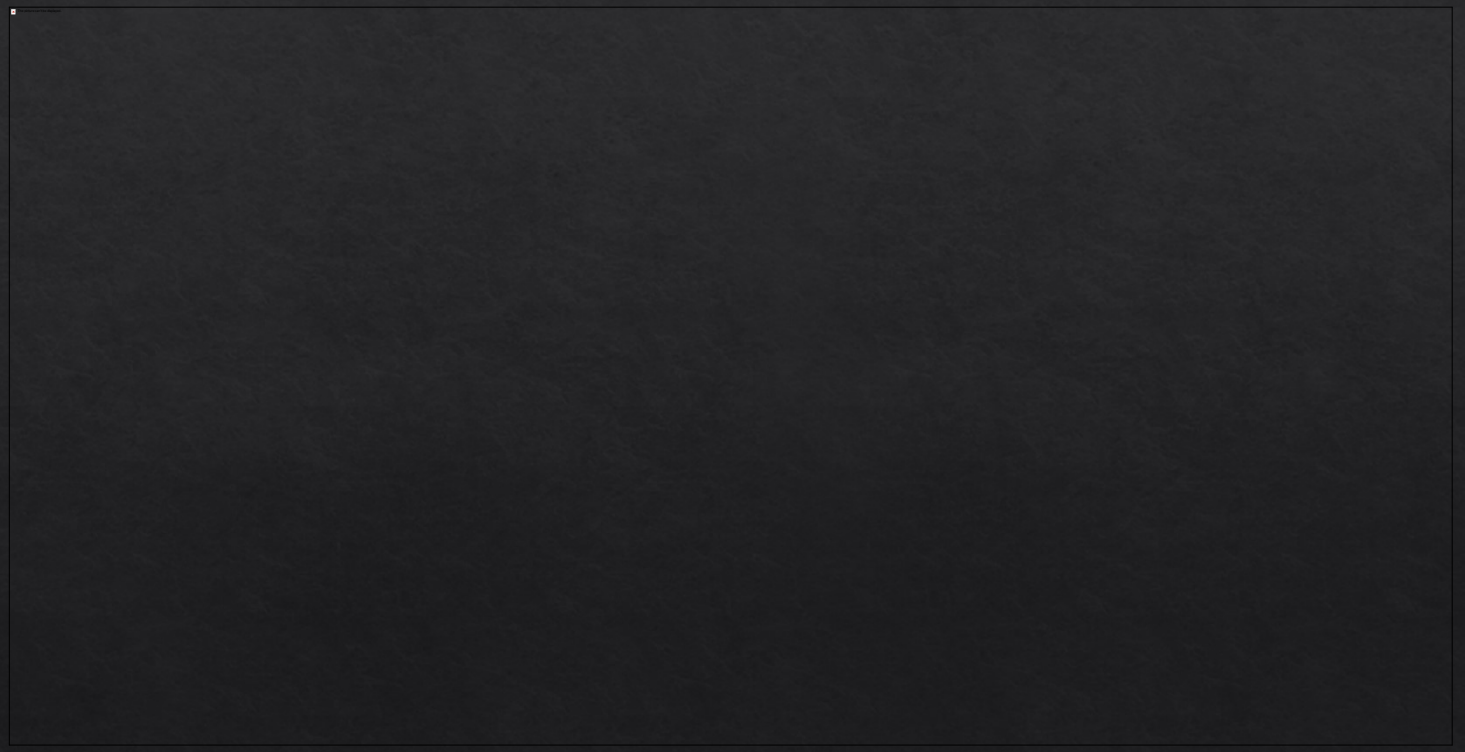
Topics We Cover

- iOS Platform and Security Mechanisms
- iOS Device Setup and Security Testing Basics
- iOS Application Structure
- About Jailbreak
- Frida Setup
- Objection
- Dump IPA File Using Frida
- Analyze Local Data Storage
- URL Schema attack
- Data Leakage
- Insecure Cryptography
- Intercepting Network Communication / SSL Pinning
- Stateful and Stateless Authentication
- Static Analysis of an IPA
- Dynamic analysis
- Jailbreak Detection via tools and manual.

Additional Exercises

What is iOS ?

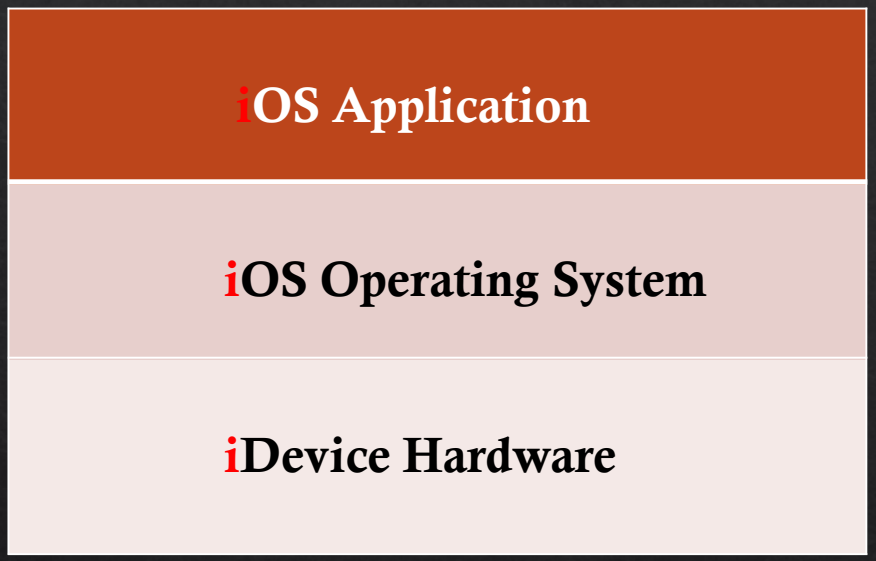
1. iOS is a mobile operating system created and developed by Apple Inc. exclusively for iPhone, iPod, iPad & Apple TV
2. Provides multi-tasking (allowing a user to perform more than one computer task)
3. It only allows to run Apple signed applications.
4. Looks similar to Traditional Linux
5. ARM Processors



iOS App Basics

- ✓ iOS native apps are developed using Objective-C & Cocoa Touch API
- ✓ XCode is the IDE only for macOS
- ✓ iOS apps are .IPA files.
- ✓ Can be extracted using unzip command.
- ✓ We can download XCode from app store
- ✓ Develop your app is free
- ✓ Run App on simulator is also Free – only macOS only
- ✓ To run your App on iDevice like iPhone/iPad – Then you need to buy pro profile from apple has cost 99$ per year

iOS Architecture



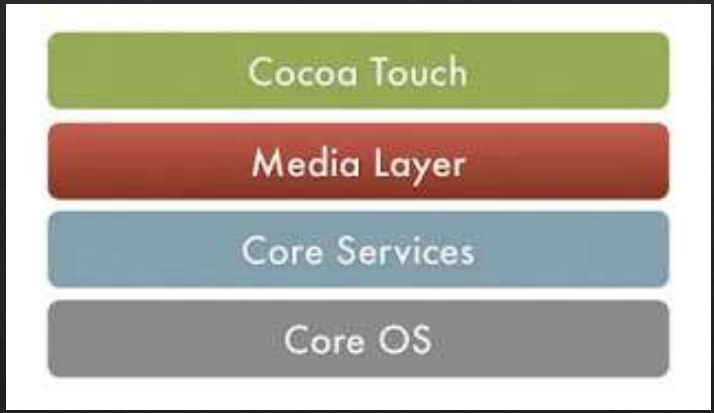
- Some key features of the iOS security model are as follows:
- ✓ Security architecture is layered as hardware level, OS level, and application level
- ✓ Encryption right from hardware/firmware level
- ✓ Application sandboxing
- ✓ Data protection using encryption
- ✓ Code signing

iOS Structure

* It contains an intermediate layer between the applications and the hardware so they do not communicate directly.it contains four layers

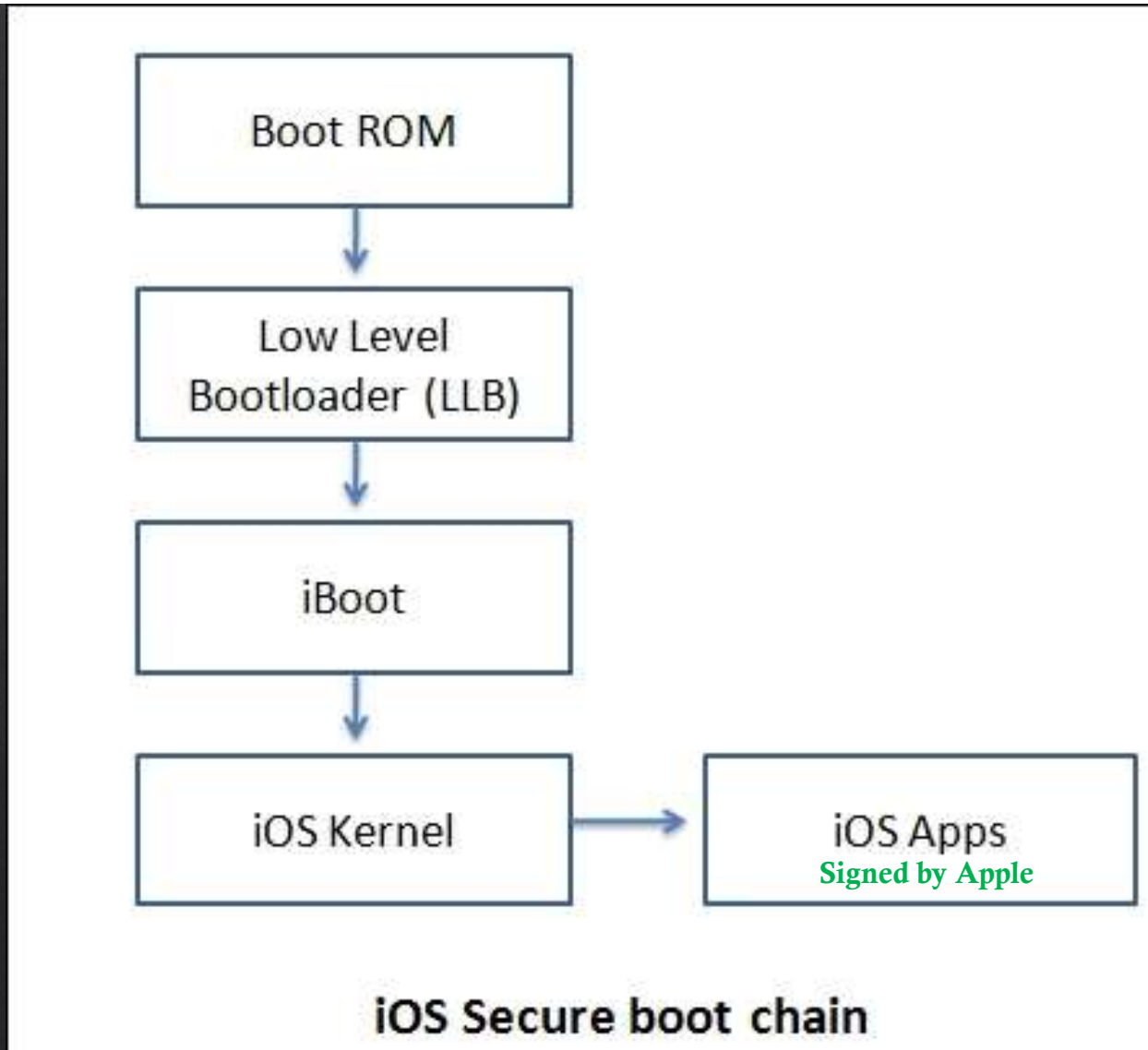
1. Core OS
2. Core Services
3. Media Services
4. Cocoa Touch

Fore More Info - https://techfrendz007.blogspot.com/2020/02/1-what-is-ios-and-its-structure.html



iOS Secure Boot Chain

- ✓ Recovery Mode
- ✓ DFU Mode
  - ✓ Default Firmware Upgrade



iOS Secure boot chain

Code Signing

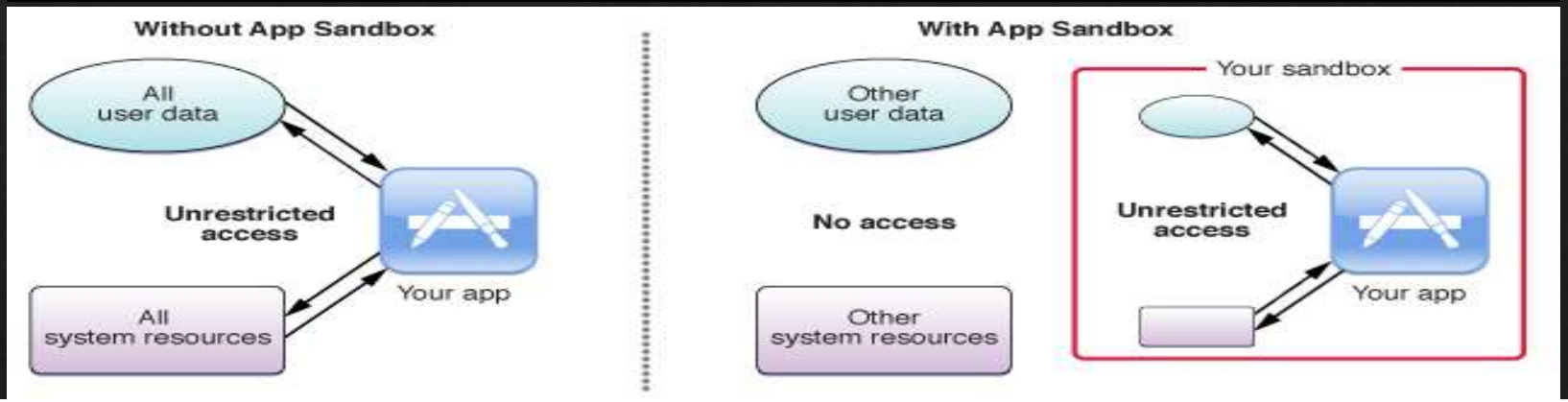
1. Only Apple approved code is allowed to be executed on an iOS device.
2. Enforces to stay “up-to-date” (you cannot downgrade to earlier iOS versions)
3. Before an app can be installed on a device, or be submitted to the App Store, it must be signed with a certificate issued by Apple (XCode is taking care of that).
4. The developer signs the apps and submits application to Apple
5. Apple verifies it (performs some rudimentary checks, not vulnerability assessment of app)
6. Need to be enrolled in the Apple Developer Program or Enterprise Program (but you can also use a free Apple account to sign apps for ~7 days)

Refer this - https://medium.com/@abhimuralidharan/what-is-a-provisioning-profile-in-ios-77987a7c54c2

Sandboxing

- Sandbox is Apple's way of allowing applications to set up a database locally on the device
- Because multiple applications will be storing data on the device's hard drive, iOS ensures that there will be no conflicts by assigning each app its own sandbox.
- So, as shown in the preceding figure, **Application 1** can't access the data of **Application 2** by default. There are certain features such as URL schemes through which they can transfer the data but with limited conditions.
- If Device is not-jailbroken then you cannot access this directory

```
Sunils-iPad:/var/mobile/Containers/Data/Application/84733569-0986-4DF9-90CD-7925108435C3/Documents root# ls -al
total 24
drwxr-xr-x 7 mobile mobile   224 Apr 27 15:13 ./
drwxr-xr-x 7 mobile mobile   224 Apr 27 14:57 ../
-rw-r--r-- 1 mobile mobile 16384 Apr 27 15:13 default.realm
-rw-r--r-- 1 mobile mobile  1184 Apr 27 15:13 default.realm.lock
drwxr-xr-x 6 mobile mobile   192 Apr 27 15:13 default.realm.management/
prw------- 1 mobile mobile     0 Apr 27 15:13 default.realm.note|
-rw-r--r-- 1 mobile mobile   278 Apr 27 15:12 userInfo.plist
Sunils-iPad:/var/mobile/Containers/Data/Application/84733569-0986-4DF9-90CD-7925108435C3/Documents root# |
```



Hardware UID & GID

- ✓ Every Apple device will have UID and GID
- ✓ Which is Unique to each device for chip that used on hardware level
- ✓ UID and GID are encrypted using AES-256 encryption keys compiled during manufacture Processor.
- ✓ No Software and firmware can read them directly
- ✓ Only able to see task performed by them
- ✓ Even Apple claims that apple doesn't know this UID and GID

Jailbreaking

The purpose of jailbreaking is to disable iOS protections (Apple's code signing mechanisms in particular) so that arbitrary unsigned code can run on the device.

1. Jailbreak gives full access to the device
2. We Can say it is the privilege escalation from a normal user to root.
3. Allows to install Apps which are not authorized
4. Install Application from Cydia which is AppStore for jailbreak Devices.

Types of jailbreaking

https://canijailbreak.com – To check available jailbreak

Tethered

Users need to reconnect to their computer every time the device is restarted

Untethered

it's one-time activity. no need to reconnect.

For more - https://techfrendz007.blogspot.com/2020/02/3-lets-jailbreak-iphoneipad.html

iOS – Application Structure

What is an IPA?

- iOS apps are distributed as IPA (iOS App Store Package) archives.
- The IPA file is a ZIP-compressed archive that contains all the code and resources required to execute the app.

Rename the .ipa file to .zip and unzip it.

```
Sunils-MacBook-Pro:Payload sunil45$ cd DVIA-v2.app/
Sunils-MacBook-Pro:DVIA-v2.app sunil45$ ls
AntiAntiHookingDebugging.storyboardc
AppIcon20x20@2x.png
AppIcon20x20@2x~ipad.png
AppIcon20x20@3x.png
AppIcon20x20~ipad.png
AppIcon29x29@2x.png
AppIcon29x29@2x~ipad.png
AppIcon29x29@3x.png
AppIcon29x29~ipad.png
AppIcon40x40@2x.png
AppIcon40x40@2x~ipad.png
AppIcon40x40@3x.png
AppIcon40x40~ipad.png
AppIcon60x60@2x.png
AppIcon60x60@3x.png
AppIcon76x76@2x~ipad.png
AppIcon76x76~ipad.png
AppIcon83.5x83.5@2x~ipad.png
AppIcons
ApplicationPatching.storyboardc
Assets.car
AttackingThirdPartyLibraries.storyboardc
Base.lproj
BinaryProtection.storyboardc
BrokenCryptography.storyboardc
ClientSideInjection.storyboardc
DVIA-v2
DVIASplash
DVIA_v2.momd
Donate.storyboardc
ExcessivePermissions.storyboardc
Frameworks
Home.storyboardc
Info.plist
InsecureDataStorage.storyboardc
JailbreakDetection.storyboardc
LICENSE.txt
LaunchImage-1100-Portrait-2436h@3x.png
LaunchImage-700-568h@2x.png
LaunchImage-700-Portrait@2x~ipad.png
LaunchImage-800-667h@2x.png
LaunchImage-800-Portrait-736h@3x.png
LearniOSApplicationSecurity.storyboardc
META-INF
Model.momd
Phishing.storyboardc
PkgInfo
README.txt
RuntimeManipulation.storyboardc
SecurityDecisonsViaUntrustedInput.storyboardc
SensitiveInformation.storyboardc
SideChannelDataLeakage.storyboardc
TouchIDBypass.storyboardc
TransportLayerProtection.storyboardc
_CodeSignature
back.png
embedded.mobileprovision
example.der
fingerprint.png
libswiftRemoteMirror.dylib
main-bg.png
main-bg@2x.png
menu.png
slider-active.png
slider-active@2x.png
slider-bg.png
slider-bg@2x.png
```

iOS – Application Structure

You can unzip the IPA using the standard tool unzip or any other ZIP utility. Inside you'll find a Payload folder containing the Application Bundle. The following is an example :

File Structure of an IPA:

- App binary (Test): The executable file containing the compiled application
- Frameworks: Dynamic libraries
- Info.plist: Application metadata
- Embedded.mobileprovsion: Certificate

Bundle Identifier can be found in Info.plist

- Go to location and use command **find | grep "DVIA"**

```
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root# plutil Info.plist
{
    BuildMachineOSBuild = 17D47;
    CFBundleDevelopmentRegion = en;
    CFBundleExecutable = "DVIA-v2";
    CFBundleIcons =     {
        CFBundlePrimaryIcon =         {
            CFBundleIconFiles =             (
                AppIcon20x20,
                AppIcon29x29,
                AppIcon40x40,
                AppIcon60x60
            );
            CFBundleIconName = AppIcon;
        };
    };
    "CFBundleIcons~ipad" =     {
        CFBundlePrimaryIcon =         {
            CFBundleIconFiles =             (
                AppIcon20x20,
                AppIcon29x29,
                AppIcon40x40,
                AppIcon60x60,
                AppIcon76x76,
                "AppIcon83.5x83.5"
            );
            CFBundleIconName = AppIcon;
        };
    };
    CFBundleIdentifier = "com.highaltitudehacks.DVIAswiftv2";
    CFBundleInfoDictionaryVersion = "6.0";
    CFBundleName = "DVIA-v2";
    CFBundlePackageType = APPL;
    CFBundleShortVersionString = "2.0";
    CFBundleSupportedPlatforms =     (
        iPhoneOS
    );
```

App Directory Structure

- ✓ Preinstalled App Directory - /var/Application

```
Sunils-iPad:/Applications root# cd /Applications/
Sunils-iPad:/Applications root# ls
AXUIViewService.app/                DNDBuddy.app/                InCallService.app/           Preferences.app/                TVAccessViewService.app/
AccountAuthenticationDialog.app/    DataActivation.app/          LoginUI.app/                 Print\ Center.app/              TVRemoteUIService.app/
ActivityMessagesApp.app/            DemoApp.app/                 MTerminal.app/               SIMSetupUIService.app/          TrustMe.app/
AdPlatformsDiagnostics.app/         Diagnostics.app/             Magnifier.app/               SLGoogleAuth.app/               Utilities/
AppStore.app/                       DiagnosticsService.app/      MailCompositionService.app/  SLYahooAuth.app/               VideoSubscriberAccountViewService.app/
AskPermissionUI.app/                FTMInternal-4.app/           MessagesViewService.app/     SafariViewService.app/          Web.app/
BusinessExtensionsWrapper.app/      Family.app/                  MobileSMS.app/               ScreenSharingViewService.app/   WebApp1.app/
CTCarrierSpaceAuth.app/             Feedback\ Assistant\ iOS.app/ MobileSafari.app/           ScreenshotServicesService.app/  WebContentAnalysisUI.app/
Camera.app/                         FieldTest.app/               MobileSlideShow.app/         Setup.app/                      WebSheet.app/
CheckerBoard.app/                   FindMyiPhone.app/            MobileTimer.app/             SharedWebCredentialViewService.app/ iAdOptOut.app/
CompassCalibrationViewService.app/  FunCameraShapes.app/         MusicUIService.app/          SharingViewService.app/         iCloud.app/
ContinuityCamera.app/               FunCameraText.app/           PassbookStub.app/            SiriViewService.app/
CoreAuthUI.app/                     GameCenterUIService.app/     PassbookUIService.app/       SoftwareUpdateUIService.app/
Cydia.app/                          HashtagImages.app/           PhotosViewService.app/       StoreDemoViewService.app/
DDActionsService.app/               HomeUIService.app/           PreBoard.app/                StoreKitUIService.app/
Sunils-iPad:/Applications root#
```

- ✓ Bundle Directory - /var/containers/Bundle/Application

```
Sunils-iPad:/var/containers/Bundle/Application root# ls
019CCB99-1528-4950-AF5C-E18F342C7017/  55667955-73E2-4ED5-8658-D5EA69FA48C4/  7E3AB03D-FAED-4642-9E42-05DC48065BB2/  AD5B4CB6-8377-48DD-8025-A3FA1319C5C6/
04228C38-CC2E-4446-A5D1-6BF310FC1D4B/  55C56F71-F2A1-42EA-826A-DC9877E4FB5D/  854D3B06-6494-43EC-AFA9-18D60E41A84B/  B935C593-3548-4199-A7AE-39C64BCFAC00/
0D74D000-6274-4AE2-9B54-892B26D5BAFF/  55CFF0B8-9555-49A4-B344-62E3097C7965/  8B114520-E572-406F-8291-1948814A8360/  B9BE51D8-EB20-4B24-A995-48025BE15F39/
0EFADE68-8D9F-450B-9832-A7C2605B6E62/  5F1FC958-30AF-4C62-B502-884E6CD818C7/  8B3677A8-AEB5-4D57-B516-1D60C1184CB3/  B9CE9787-401D-4D68-BCEB-185E550B2AEC/
1076656C-1E4E-4A2D-A28F-64D10C9C0E62/  6805F38A-1FE2-44CC-9744-44A966F395A8/  93F8A990-C068-4C7B-90DF-A9F40F26367E/  BDB52F7E-531E-4CA5-A59F-999CB9F449F1/
1B18A621-20D2-481C-8CF9-30AE7E9FE927/  6814AE57-8722-4D95-B51D-53DD87077B8A/  9429CC3A-E4BD-427E-BADB-05FFF38233B2/  C008FF03-AE5E-4547-B3EF-15B90102904B/
22FC57FE-C772-470A-9BEF-D061356D1E0A/  6CA99C0F-11AF-4D1D-A6E5-D948A313E925/  98EF1512-ADD9-4764-8B42-F5709B6B5A11/  D5A14430-D110-4596-BF7F-91C661D8EAED/
2936A02D-A343-46EB-9FA4-D50E3ADB7B08/  6EA6E061-8968-43F8-B91A-DBAA2031BE8B/  9BDA30EC-2F2C-4D3B-AF81-EB92D69AB683/  D6B6E18E-F86B-48F4-9BAB-108C17A0473F/
3019B3E9-0215-420D-BDAB-690706E20A86/  733E1C78-0BFB-4340-BBE7-B09F4968F97F/  9D0E2920-D168-4C65-8425-3F8D8496DE38/  D753D077-3135-4DB9-B191-5D199BE2B2FD/
30F16AF4-1130-46E1-A778-214BF4C2159C/  739FDB86-799E-4F3B-86C0-DD0683F5306D/  9D46EB3D-0B1B-4999-889A-BDE2166F01C8/  D9F01518-0283-4DB9-8767-1F1017889439/
31E48C0A-04D2-4C9A-B667-EA18564930BE/  73BF5DF3-5D6B-4AE7-AE2A-2F5C675AC98C/  9F44AD34-1CD7-44DD-BE29-E14F067BAACC/  DD509F0E-0058-4B9B-9716-77CAC324DE25/
37A073F5-CE44-48CB-9DB5-7C98D86A5E22/  75C24717-E529-4BB3-B052-5C2161F8CEA1/  A167764D-1CE2-4547-AB72-AD3FBFF7A8C2/  DE9E6934-D626-45CD-A455-7E9DF66607B1/
400EF6B8-481C-4D54-BAC9-65E15037C9FE/  76F2E906-1F7B-4B76-93A5-E30E7DA6A68B/  A45E7851-EC7D-4976-B276-7794BE26128D/  E38E7CE7-77E3-4D7D-95D6-A0FDFAF6DDD1/
424FB278-055C-4307-BAA8-08E9DD419EE8/  7754F747-F999-4B2C-A606-9B88372E271D/  A4768591-32A6-4602-A9F3-B1F2A0643B28/  EBE27475-55FF-4FF4-8606-2B62DD2757D5/
509EE7F4-7FD8-427C-819F-474353FBC414/  7A07DB0C-4C4F-47D0-B7D5-05EB44614FFD/  A68D0B05-C0DE-4BCF-ADAD-53FBA698BDF0/  ED3A736C-0AE4-448A-96FC-9DE595B5F335/
51B6050B-A588-41FB-80CB-F1314B3C2110/  7C2FB8EB-A9D1-4C5A-92EB-7AF10889DAD7/  A6999899-3761-4F72-89F1-17A63DF8F971/  EFB9BBC0-B019-45D3-ABBB-7CBF7EEC0C2C/
53DD1B52-08D9-47C2-A326-D31A2642C90E/  7C5839C9-5E88-49A0-B9E4-3430F46CF5F0/  A848F45E-DF1C-421A-AF70-CDB3FF5C7196/  F17A58E8-BC40-441A-A1E2-365FF511AA71/
Sunils-iPad:/var/containers/Bundle/Application root#
```

- ✓ Data Directory - /var/mobile/Containers/data/Application

```
ıils-iPad:/var/mobile/Containers/Data/Application root# ls
'98BC0-2C03-4A46-BA07-83DB063153AA/  3B085227-DA45-4CD3-BF4F-F60493133197D/  7B4DBEF6-4424-4CF4-8A5E-D339D7F378B1/  C88FA0EA-84ED-450B-8322-3A7969B57C21/
:2F3D7-0607-4FE3-A0B7-F4C38FA65A52/  3D98D770-B942-46D5-B7CC-E4EB4F731E3F/  7BCA9AC4-B966-4482-B8C4-B97A4B18DA80/  CBC9FC46-C490-4F88-A0E7-F29AFFFCB37B/
384151-71D4-48A4-8811-1F1B7E98FD4E/  3E539942-2209-4F1B-A9BB-FE031F992B27/  7CF2040C-CFD4-436E-94F5-3D79E312992C/  CDD28BB6-B5AF-465F-A654-99A2029EB252/
:E4736-BDD1-41D9-86FD-882F9FCF7356/  3FA320CD-B379-4023-ABE5-952D17100FF3/  7EEC639C-67FB-4C01-9DC7-802C304F2D78/  CE84CD2D-06A7-4A12-A128-227DDEC105DE/
```

Address Space Layout Randomization (ASLR)

- ✓ To protect against exploits.
- ✓ ASLR randomizes the location of an application heap.
- ✓ Other element like binary, data, stack, and dynamic linkers are mapped to fix locations.
- ✓ We can use Position Independent Executable (PIE) flag to avoid this.

To check whether PIE is set or not use Otool

To install Otool Go to Cydia and install Darwin CC tool

```
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root# otool --help
error: otool: unknown char `-' in flag --help

Usage: otool [-arch arch_type] [-fahlLDtdorSTMRIHGvVcXmqQjCP] [-mcpu=arg] [--version] <object file> ...
        -f print the fat headers
        -a print the archive header
        -h print the mach header
        -l print the load commands
        -L print shared libraries used
        -D print shared library id name
        -t print the text section (disassemble with -v)
        -p <routine name>  start dissassemble from routine name
        -s <segname> <sectname> print contents of section
        -d print the data section
        -o print the Objective-C segment
        -r print the relocation entries
        -S print the table of contents of a library
        -T print the table of contents of a dynamic shared library
        -M print the module table of a dynamic shared library
        -R print the reference table of a dynamic shared library
        -I print the indirect symbol table
        -H print the two-level hints table
        -G print the data in code table
        -v print verbosely (symbolically) when possible
        -V print disassembled operands symbolically
        -c print argument strings of a core file
        -X print no leading addresses or headers
        -m don't use archive(member) syntax
        -B force Thumb disassembly (ARM objects only)
        -q use llvm's disassembler (the default)
        -Q use otool(1)'s disassembler
        -mcpu=arg use `arg' as the cpu for disassembly
        -j print opcode bytes
        -P print the info plist section as strings
        -C print linker optimization hints
        --version print the version of otool
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root# otool -hv DVIA-v2
DVIA-v2:
Mach header
      magic cputype cpusubtype  caps    filetype ncmds sizeofcmds      flags
MH_MAGIC_64   ARM64        ALL  0x00     EXECUTE    65       7112   NOUNDEFS DYLDLINK TWOLEVEL WEAK_DEFINES BINDS_TO_WEAK PIE
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root#
```

Stack Smashing Protection:

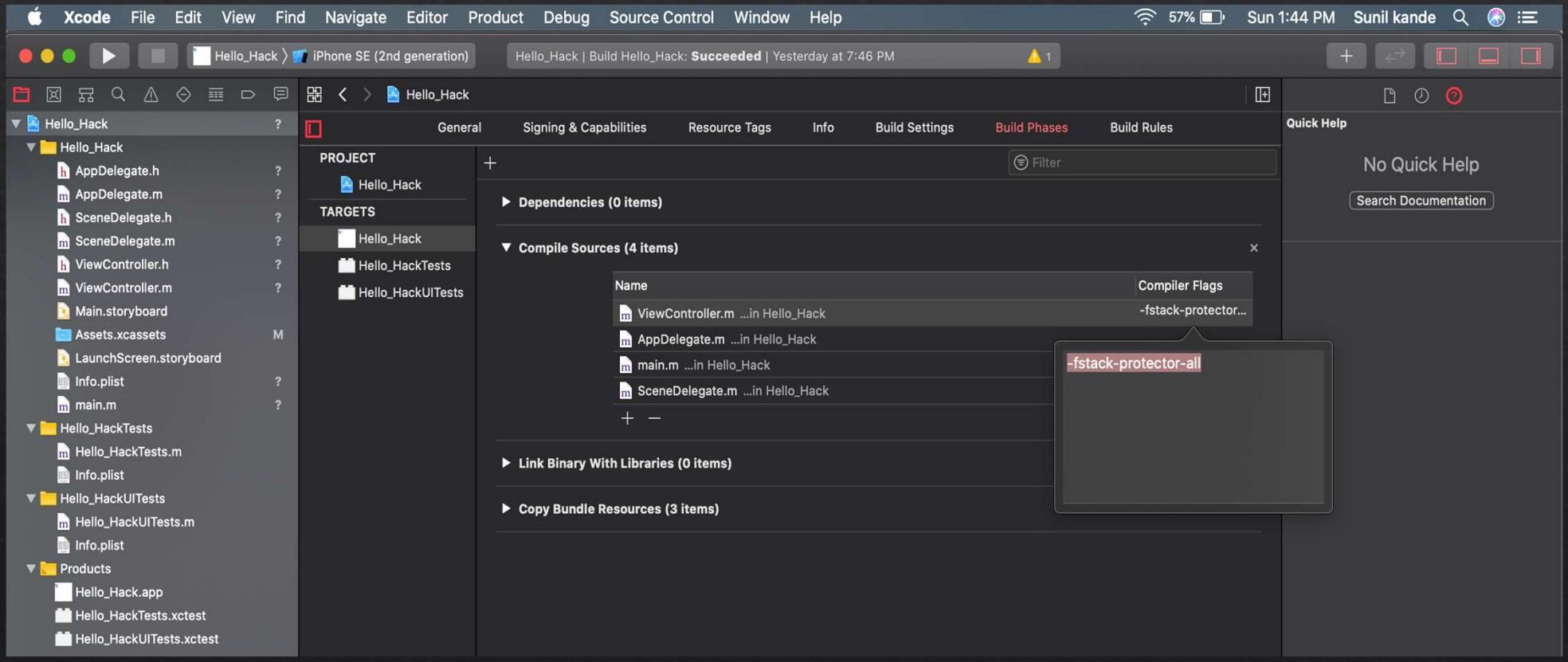
- ✓ Stack smashing protection is an exploit mitigation technique that protects against stack overflow attacks by placing a random value known as stack canary before local variables on stack.
- ✓ iOS applications which use the stack canaries will contain **_stack_chk_fail and _stack_chk_guard** symbols in the binary.
- ✓ To find out whether the application is protected with stack smashing protection or not, connect the iPhone over SSH and execute the command below.
- ✓ **Otool –I –v ApplicationBinary | grep stack**

```
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root# otool -I -v DVIA-v2 | grep stack
0x0000000100329b80 70498 ___stack_chk_fail
0x00000001003a4458 70499 ___stack_chk_guard
0x00000001003a55b8 70498 ___stack_chk_fail

[1]+  Stopped                 otool -I -v DVIA-v2 | grep stack
[1]+  Done                    otool -I -v DVIA-v2 | grep stack
Sunils-iPad:/var/containers/Bundle/Application/53DD1B52-08D9-47C2-A326-D31A2642C90E/DVIA-v2.app root#
```

- ✓ https://resources.infosecinstitute.com/penetration-testing-for-iphone-applications-part-5/#gref

We can enable it in XCode Beforcompiling an app



ATS Security in iOS 12

- App Transport Security (ATS) is an iOS feature that forces mobile apps to connect to back-end servers using HTTPS, instead of HTTP.
- For iOS 9.0 or later enable ATS by default, but developers can opt-out by setting an app's NSAllowsArbitraryLoads key to True.
- Apple announced a requirement to be enforced by the end of 2016 — apps submitted to the App Store must support ATS
- it's not clear how long Apple plans to suspend the requirement. But developer should take care of it.

  - **NSAllowsArbitraryLoads key to True – Misconfigured - Look for Plist file**
    - meaning in some cases it's possible that those apps send sensitive data insecurely over HTTP.

  NSAllowsArbitraryLoads key to – false

https://developer.apple.com/library/archive/documentation/General/Reference/InfoPlistKeyReference/Articles/CocoaKeys.html#//apple_ref/doc/uid/TP40009251-SW33

https://www.nowsecure.com/blog/2017/08/31/security-analysts-guide-nsapptransportsecurity-nsallowsarbitraryloads-app-transport-security-ats-exceptions/

Auditing ATS implementations

- ✓ NSAllowsArbitraryLoads
- ✓ NSAllowsArbitraryLoadsForMedia
- ✓ NSAllowsArbitraryLoadsInWebContent
- ✓ NSExceptionAllowsInsecureHTTPLoads
- ✓ NSExceptionMinimumTLSVersion

- ✓ Unzip ipa file or go to bundle directory
- ✓ Take a look at Plist file

ATS configuration which will look something like this:

```
MinimumOSVersion = "12.0";
NSAppTransportSecurity =     {
    NSAllowsArbitraryLoadsInWebContent = 1;
    NSExceptionDomains =         {
        "l.yimg.com" =             {
            NSExceptionAllowsInsecureHTTPLoads = 1;
        };
        localhost =             {
            NSExceptionAllowsInsecureHTTPLoads = 1;
        };
    };
};
```

KeyNotes

1. NSAllowsArbitraryLoads - key is set to NO by default. Setting the key to YES will opt-out of ATS and its associated security benefits. If in testing an app you find this key set to YES, verify why the developers decided to opt out
2. NSExceptionAllowsInsecureHTTPLoads - By default this key is set to NO. When this key is set to YES, the app will be allowed to send HTTP traffic to that domain.

3. NSAllowsArbitraryLoadsInWebContent - By default, NSAllowsArbitraryLoadsInWebContent is set to NO. When the key is set to YES, ATS is disabled for WebView requests.
    1. so it's crucial to verify their security. For example, web views can be vulnerable to a number of common web-based vulnerabilities such as SQL injection, cross-site request forgery, and cross-site scripting attacks
4. NSExceptionMinimumTLSVersion - This key allows developers to lower the minimum accepted version of TLS. By default, TLS 1.2 and higher are the accepted versions

Imp Instruction

If the device is getting locked while you are analyzing an app, tools might break, or you get timeouts!

Increase the time for the Auto-Lock!

Go to Setting / Display & Brightness / Auto-lock and set it to either 5 minutes or Never

Insecure Data Storage

- Is a vulnerability which occurs when sensitive data is stored locally on the device without any proper protection.
- Common place like following
  - Plist file
  - NSUser default
  - Keychain
  - Core Data
  - SQLite etc.

Analysis of Plist File

From the apple documentation:

- *An information property list file is a structured text file that contains essential configuration information for a bundled executable.*
- The system uses these keys and values to obtain information about your app and how it is configured.
- So what does this mean

  Every application is expected to have an Info.plist file so the system can read it and work according to it's contained values. When you create a new project you are given an Info.plist file with default settings all of which can be customized.

Look for following info in Plist file

- ✓ URL Schema
- ✓ Look for ATS Security
- ✓ Look for Bundle Identifier
- ✓ Look for hard coded data
- ✓ Min iOS version supportable etc.

API Keys In Plist File

- ✓ Always Look at Plist file under Bundle directory.
- ✓ Many times keys are hardcoded like firebase, google map etc
- ✓ Then search use keyhack for the exploitation.
  - ✓ https://github.com/streaak/keyhacks

```
    }
);
CFBundleVersion = 1065;
DTAppStoreToolsBuild = 10G3;
DTCompiler = "com.apple.compilers.llvm.clang.1_0";
DTPlatformBuild = 16E226;
DTPlatformName = iphoneos;
DTPlatformVersion = "12.2";
DTSDKBuild = 16E226;
DTSDKName = "iphoneos12.2";
DTXcode = 1020;
DTXcodeBuild = 10E125;
Fabric =     {
    APIKey = bde24571d7754c4bf69a5da86149c1c5f0b115c5;
    Kits =         (
                    {
            KitInfo =                 {
                consumerKey = 1cajoqaYUEWbrJyheJ7ocq36W;
                consumerSecret = wF677d702uj9kWQXwNpYT4A2vAP7u6b64aUqfAMnppYzL1tSw3;
            };
            KitName = Twitter;
        },
                    {
```

Check for Plist File.

- ✓ Plist files should always be used for saving information that is not confidential as they are unencrypted and can be easily be fetched even from a non-jailbroken device

```
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Documents root# plutil userInfo.plist
{
    password = mysecretpassword007;
    username = "Sunil ";
}
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Documents root#
```

SQLite Data

- ✓ Many times Developer Stores Username and Password Stored in cleartext without any encryption

```
Sunils-iPad:/var/mobile/Containers/Data/Application/31ECA56D-8EF4-43B6-A768-9620AB12C1A7/Library/Application Support/CouchbaseLite/dvcouchbasedb.cblite2 root# ls
attachments/  db.sqlite3  db.sqlite3-shm  db.sqlite3-wal
Sunils-iPad:/var/mobile/Containers/Data/Application/31ECA56D-8EF4-43B6-A768-9620AB12C1A7/Library/Application Support/CouchbaseLite/dvcouchbasedb.cblite2 root# sqlite3 db.sqlite3
SQLite version 3.24.0 2018-06-04 19:24:41
Enter ".help" for usage hints.
sqlite> .tables
docs       info       localdocs  revs       views
sqlite> select * from revs;
1|1|1-0d32cb4d5aea24df82aa416e40f09e58||1|0|{"password":"pass11337777","username":"sunil123"}|1|
sqlite>
```

NsUserDefault

- ✓ One of the most common ways of saving user preferences and properties in an application is by using NsUserDefault. The information stored in NsUserDefault persists even if you close the application and start it again.
- ✓ NsUserDefault store data in Plist file format file under preference folder will be NsUserDefault

```
[Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Preferences root# plutil com.highaltitudehacks.DVIAswiftv2.pl]
ist
{
    DemoValue = "I\U2019m hacker. Lol";
}
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Preferences root#
```

Core Data

- ✓ Since Core Data basically uses SQLite internally to save information.
- ✓ So basically, Core Data can be used to create a model, manage relationships between different types of objects, save the data locally, and fetch them from the local cache whenever you want with queries

```
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library root# cd Application\ Support/
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Application Support root# ls
FlurryFiles/  Model.sqlite  Model.sqlite-shm  Model.sqlite-wal
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Application Support root# sqlite3 Model.sqlite
SQLite version 3.24.0 2018-06-04 19:24:41
Enter ".help" for usage hints.
sqlite> .tables
ZUSER         Z_METADATA    Z_MODELCACHE  Z_PRIMARYKEY
sqlite> select * from ZUSER;
1|1|1|jamesbond@007.com|Sunil|hack007|9096777777
sqlite>
```

Keychain

- ✓ The keychain is just a data storage for storing all sensitive data, like passwords, certificates etc.
- ✓ Items added to the Keychain are encoded as a binary Plist and encrypted with a 128-bit AES
- ✓ You can configure data protection for Keychain items by setting the kSecAttrAccessible
  - ✓ kSecAttrAccessibleAlways
  - ✓ kSecAttrAccessibleAlwaysThisDeviceOnly
  - ✓ kSecAttrAccessibleAfterFirstUnlock
  - ✓ kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly
  - ✓ kSecAttrAccessibleWhenUnlocked
  - ✓ kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly
  - ✓ kSecAttrAccessibleWhenUnlockedThisDeviceOnly

  - ✓ https://mobile-security.gitbook.io/mobile-security-testing-guide/ios-testing-guide/0x06d-testing-data-storage

~ — objection -g com.highaltitudehacks.DVIAswiftv2 explore | ~/Desktop — ssh root@192.168.1.103

```
Sunils-MacBook-Pro:~ sunil45$ frida-ps -Ua
  PID  Name      Identifier
-----  --------  ------------------------------
10664  DVIA-v2   com.highaltitudehacks.DVIAswiftv2
 7899  Mail      com.apple.mobilemail
10684  Rivals    com.yahoo.rivals
10548  Settings  com.apple.Preferences
Sunils-MacBook-Pro:~ sunil45$ objection -g com.highaltitudehacks.DVIAswiftv2 explore
Using USB device `iPad 4`


A newer version of objection is available!
You have v1.8.2 and v1.9.1 is ready for download.

Upgrade with: pip3 install objection --upgrade
For more information, please see: https://github.com/sensepost/objection/wiki/Updating

Agent injected and responds ok!


     _   _         _   _
 ___| |_|_|___ ___| |_|_|___ ___
| . | . | |_ -|  _|  _| | . |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.8.2

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
....highaltitudehacks.DVIAswiftv2 on (iPad: 12.4) [usb] # ios keychain dump
Note: You may be asked to authenticate using the devices passcode or TouchID
Save the output by adding `--json keychain.json` to this command
Dumping the iOS keychain...
Created                  Accessible           ACL    Type      Account                  Service                                                  Data
-----------------------  -------------------  -----  --------  -----------------------  -------------------------------------------------------  ----------
-----------------------
2019-11-23 08:17:08 +0000  WhenUnlocked         None   Password  FlurryAPIKey             com.highaltitudehacks.DVIAswiftv2com.flurry.analytics
2019-11-23 08:17:08 +0000  WhenUnlocked         None   Password  FlurrySessionTimestampKey  com.highaltitudehacks.DVIAswiftv2com.flurry.analytics
2019-11-23 08:17:08 +0000  AlwaysThisDeviceOnly  None   Password  FlurrySessionInstallIDKey  com.highaltitudehacks.DVIAswiftv2com.flurry.analytics  1464C452-2
AAE-4017-AE37-D429FE5F96E8
2019-11-23 08:58:00 +0000  WhenUnlocked         None   Password  keychainValue            com.highaltitudehacks.DVIAswiftv2                        mysecret
....highaltitudehacks.DVIAswiftv2 on (iPad: 12.4) [usb] #
```

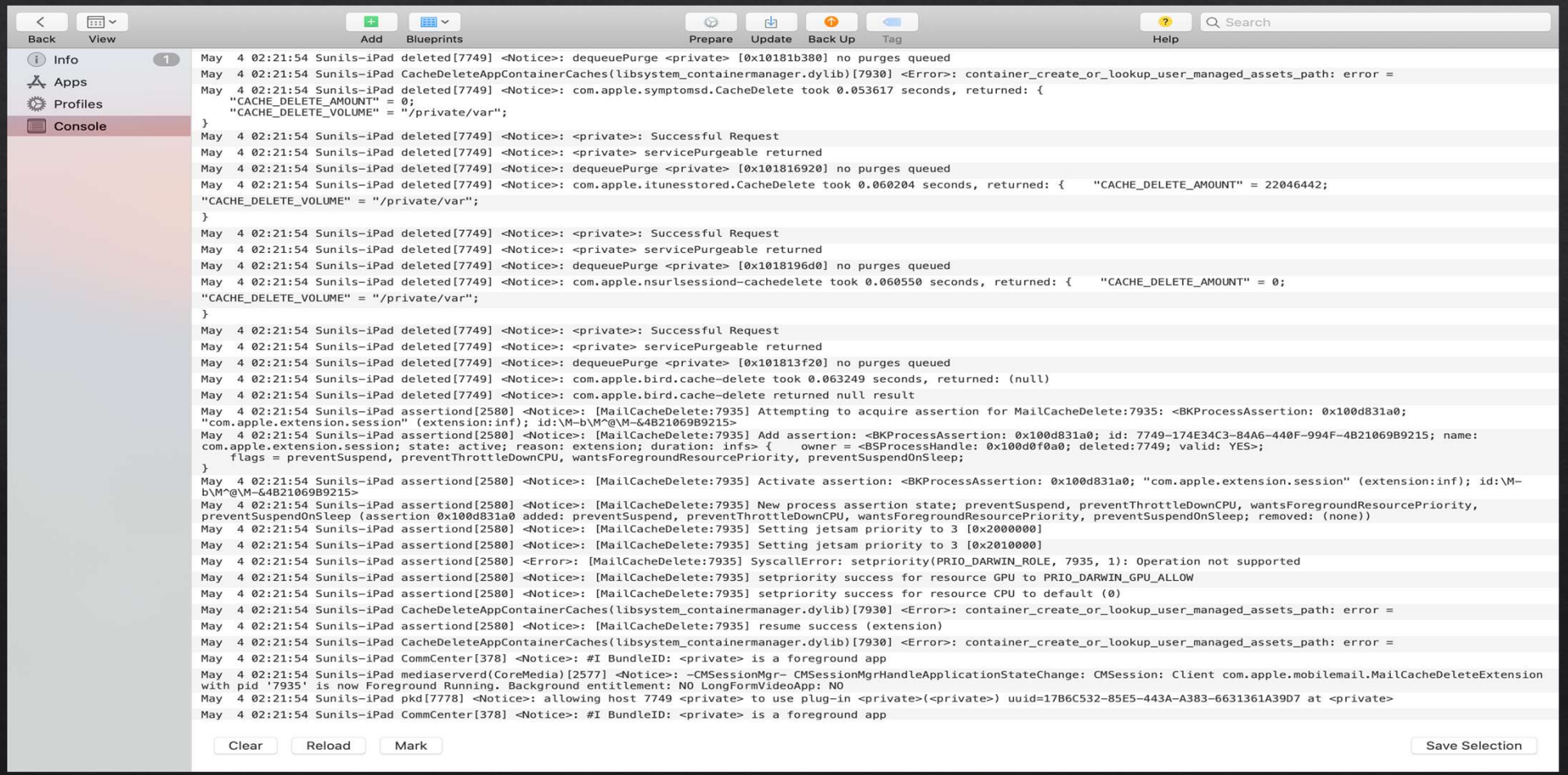
✓ Unintended Data Leakage

This happens When sensitive data is leaked accidentally by the application.

Attack Surface.

- ✓ Device Log
- ✓ App Screenshot
- ✓ Pasteboard
- ✓ Keystroke Logging
- ✓ Cookies

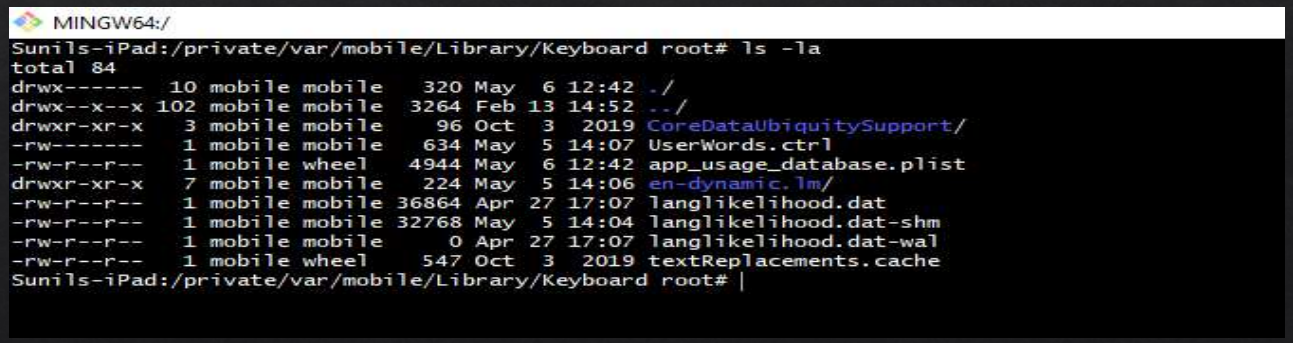
Device Log – Data Leak



App Screenshot

Pasteboard & Keyboard Cache

- ✓ iOS by default logs every input that you enter in a text fields unless text fields are not set as secure.
- ✓ All the Copy and autocorrected strings are stored in iOS cache store location.
  - ✓ **/private/var/mobile/Library/Keyboard**
  - ✓ Therefore it is essentials for developer to disable autocorrect values on sensitive filds like credit card number , passwords etc.



Binary Cookies

- Some application creates persistent Cookies and store them in locally

```
     _     _           _  _
 ___| |_|_|_|___ ___ _| |_|_|___ ___
| . | . | | |  _|  _| . | |   |
|___|___| |___|___|_| |_|___|_|_|
      |___|(object)inject(ion) v1.8.2

     Runtime Mobile Exploration
        by: @leonjza from @sensepost

[tab] for command suggestions
[....highaltitudehacks.DVIAswiftv2 on (iPad: 12.4) [usb] # ios cookies get --json                                                                ]
[
    {
        "domain": "highaltitudehacks.com",
        "expiresDate": "2052-03-03 20:11:55 +0000",
        "isHTTPOnly": "false",
        "isSecure": "false",
        "name": "username",
        "path": "/",
        "value": "admin123",
        "version": "0"
    },
    {
        "domain": "highaltitudehacks.com",
        "expiresDate": "2052-03-03 20:11:55 +0000",
        "isHTTPOnly": "false",
        "isSecure": "false",
        "name": "password",
        "path": "/",
        "value": "dvpassword",
        "version": "0"
    }
```

Jailbreak Detection Bypass

Two ways to bypass jailbreak detection, via tool or using frida/objection

1. Using Tools (Automatic)

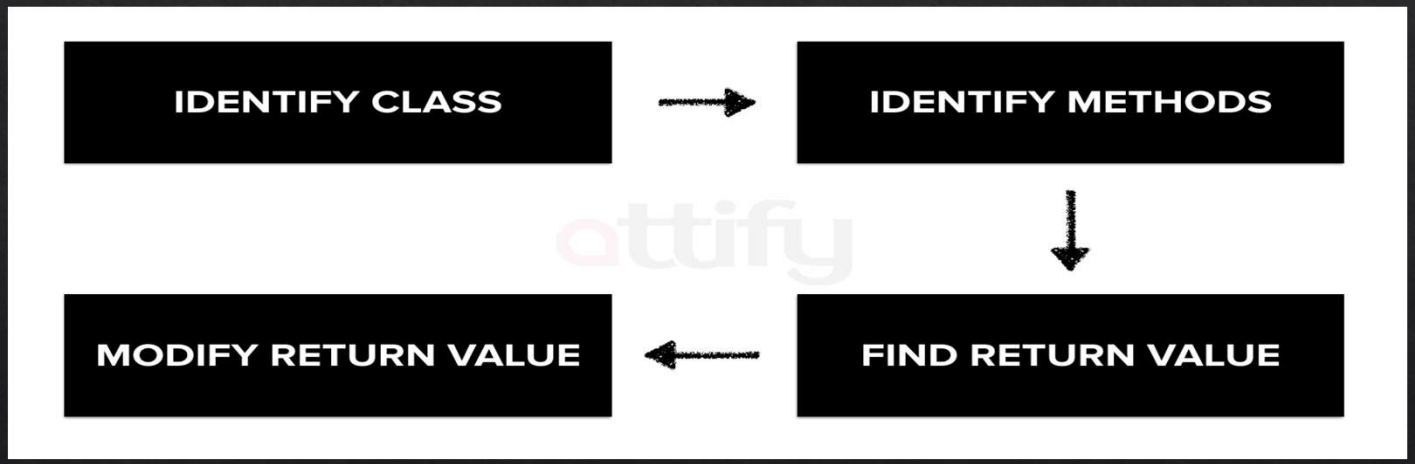
- ✓ JailProtect
- ✓ UnSub
- ✓ Liberty Lite

1. Open Cydia
2. Tap on Sources > Edit
3. Tap Add and type in the URL – **http://julioverne.github.io**
4. Tap on Add and leave it
5. When it's been added, tap on Search and type in JailProtect
6. Tap on Install
7. When it's done, you can open Settings > JailProtect to configure it

https://cydia-app.com/jailbreak-detection/

2. Using Frida

- We use **DVIA** iOS for jailbreak detection bypass.
- Here, we will be looking for anything related to Jailbreak, so that we are able to perform Jailbreak Bypass with the help of Frida.
- Here's how our process would look like



Frida Setup

- Frida is an open source runtime instrumentation framework that lets you inject JavaScript code or portions of your own library into processes on Windows, Linux, macOS and also native Android and iOS apps.
- Some of the use cases of Frida (depending on for which purpose you are using it) are –
- Hooking into a specific function and changing the return value
- Analyzing custom protocols in place and sniffing/decrypting the traffic on the fly
- Performing debugging of your own applications
- Dumping class and method information from an iOS app
- And so on.

Installation

Client =>

Requirements

- Python – latest 3.x is highly recommended
- Multiplatform (GNU/Linux, MacOS or Windows)

Open Terminal

```
$ pip3 install frida-tools
$ pip3 install frida
```

Remote Server /on iDevice

Cydia: Add Frida's repository: https://build.frida.re

Install Frida package

Verify

A quick test, to make the basics are working:

Connect Frida to your device over USB and list all running processes

$ frida-ps –U

PID Name

---

124 add

1171 com.sunil

1437 com.suma

Objection Setup

Open Terminal

$pip3 install objection

Connect iDevice via USB

$objection –g

Frida-Dump Setup

1. Install Frida Client & Server
2. Download frida-dump script from https://github.com/AloneMonkey/frida-ios-dump.git
3. Connect iDevice to PC via USB
4. Do the changes in dump.py on user, pass, port with respective to your configuration
5. Use command python3 dump.py –l for listing all installed app package name
6. Use command python3 dump.py com.sunil => name of app which want to dump

For more - https://techfrendz007.blogspot.com/2020/04/4-setting-up-fridaobjectionfirebase.html

Finding Classes for Jailbreak Detection

- Let's get started by identifying all the classes in the application. By using simple js code.

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

- it will show you a ton of classes present in our target process.

- Use command line
  - frida -U -l find-classes.js -f package/app_name
  - It will list all the classes name present.
  - To find particular class name use grep command like
    - frida -U -l find-classes.js -f com.sunil.app | grep -i jailbreak

To analyze particular Class methods

```
console.log("[*] Started: Find All Methods of a Specific Class");
if (ObjC.available) {
try {
var className = “classname";
var methods = eval('ObjC.classes.' + className + '.$methods');
for (var i = 0; i < methods.length; i++) {
try { console.log("[-] "+methods[i]); }
catch(err) { console.log("[!] Exception1: " + err.message); }
} }
catch(err) { console.log("[!] Exception2: " + err.message); } }
else { console.log("Objective-C Runtime is not available!"); }
console.log("[*] Completed: Find All Methods of a Specific Class");
```

Check what values Function Returns

```
if (ObjC.available) {

try { var className = "JailbreakDetectionVC";
var funcName = "- isJailbroken";
var hook = eval('ObjC.classes.' + className + '["' + funcName + '"]');

Interceptor.attach(hook.implementation, {
onLeave: function(retval) { console.log("[*] Class Name: " + className);
console.log("[*] Method Name: " + funcName);
console.log("\t[-] Type of return value: " + typeof retval);
console.log("\t[-] Return Value: " + retval); } }); }
catch(err) { console.log("[!] Exception2: " + err.message); } }

else { console.log("Objective-C Runtime is not available!"); }
```

frida -U -l return.js -f appname

Dump iOS App

- ✓ In order to scan for source code of an ios app we need IPA file
- ✓ We use frida-dump script will do this
- ✓ Download from https://github.com/AloneMonkey/frida-ios-dump.git
- ✓ Change configuration in dump.py file with respective to your iDevice.

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

User = 'root'
Password = 'sunil'
Host = '192.168.1.104'
Port = 22
```


Sunils-MacBook-Pro:frida-ios-dump sunil45$ python3 dump.py --help
usage: dump.py [-h] [-l] [-o OUTPUT_IPA] [-H SSH_HOST] [-p SSH_PORT]
[-u SSH_USER] [-P SSH_PASSWORD]
[target]
frida-ios-dump (by AloneMonkey v2.0)
positional arguments:
target Bundle identifier or display name of the target app
optional arguments:
-h, --help show this help message and exit
-l, --list List the installed apps
-o OUTPUT_IPA, --output OUTPUT_IPA
Specify name of the decrypted IPA
-H SSH_HOST, --host SSH_HOST
Specify SSH hostname
-p SSH_PORT, --port SSH_PORT
Specify SSH port
-u SSH_USER, --user SSH_USER
Specify SSH username
-P SSH_PASSWORD, --password SSH_PASSWORD
Specify SSH password
frida-ios-dump script
Start the target app com.netflix.Netflix
Dumping Netflix to /var/folders/s_/48k0xm2j4xj4vkl7g2w7q9lr0000gn/T
[frida-ios-dump]: ArgoCore.framework has been loaded.
[frida-ios-dump]: CoreAardvark.framework has been loaded.
[frida-ios-dump]: NFPlatformCommon.framework has been loaded.
[frida-ios-dump]: NFManifestAccess.framework has been loaded.
[frida-ios-dump]: Nbp.framework has been loaded.
[frida-ios-dump]: ReSwiftThunk.framework has been loaded.
[frida-ios-dump]: NFUIPlayerFoundation.framework has been loaded.
[frida-ios-dump]: MslClient.framework has been loaded.
[frida-ios-dump]: FTLProbeClient.framework has been loaded.
[frida-ios-dump]: widevine_cdm_sdk_oemcrypto_release.framework has been loaded.
[frida-ios-dump]: ReSwift.framework has been loaded.
[frida-ios-dump]: NFURLSession.framework has been loaded.
[frida-ios-dump]: NFCW444.framework has been loaded.
[frida-ios-dump]: Bugsnag.framework has been loaded.
[frida-ios-dump]: Lottie.framework has been loaded.
[frida-ios-dump]: NFUIKit.framework has been loaded.
[frida-ios-dump]: NFPlaylistGeneration.framework has been loaded.
[frida-ios-dump]: Apollo.framework has been loaded.
[frida-ios-dump]: Aardvark.framework has been loaded.
start dump /var/containers/Bundle/Application/6805F38A-1FE2-44CC-9744-44A966F395A8/Argo.app/Argo
Argo.fid: 29%
| 7.23M/25.3M [00:13<00:31, 598kB/s]

Mobsf – Static Analysis

- Mobile Security Framework (**MobSF**) is an automated, open source, all-in-one mobile application (Android/iOS/**Windows**) pen-testing framework capable of performing static, dynamic and malware analysis.
- https://github.com/MobSF/Mobile-Security-Framework-MobSF

```
root@ubuntu:~#git clone https://github.com/MobSF/Mobile-Security-Framework-MobSF.git

root@ubuntu:~#cd Mobile-Security-Framework-MobSF

root@ubuntu:~#./setup.sh

root@ubuntu:~#./run.sh

As we see it start the server on http://0.0.0.0:8000
```

What is Firebase Database.

- The Firebase Realtime Database is a cloud-hosted database. Data is stored as JSON and synchronized in Realtime to every connected client.
- When you build cross-platform apps with our iOS, Android, and JavaScript SDKs, all of your clients share one Realtime Database instance and automatically receive updates with the newest data.

Misconfiguration at permission

issue exists where if due to Human Error the security rules of the database are left on **"true"** for Both of the values ie read and write.

- Result of this we can write in database if the following misconfiguration is there.

{

".read" :true

".write" :true

}

Step.

1. Reverse the ipa file using frida-dump.
2. Look for string **firebaseio.com**
3. Download script from here
4. Use python script.py and follows the instruction.
5. https://firebase.google.com/docs/database/ios/read-and-write

Insecure-Firebase-Exploit

https://github.com/MuhammadKhizerJaved/Insecure-Firebase-Exploit

```
sunil@ubuntu:~/Desktop/fridascript/Insecure-Firebase-Exploit$ python Firebase-Write-Permission-Exploit.py

             ############ "Firebase Database Permissions Vulnerability Exploit ############
Usage   : Provide target dB name, filename to be create, information to write
Blog    : Read Full Blog about
Url     : https://blog.securitybreached.org/2020/02/04/exploiting-insecure-firebase-database-bugbounty
Authors : Muhammad Khizer Javed, Daniyal Nasir
Update  : Haroon Awan

[>] Input Settings for exploit

[+] Enter firebase DB name : paytm-hero
[+] Enter your filename    : file1
[+] Enter your name        : sunil
[+] Enter your email       : test@gmail.com
[+] Enter your Blog        : https://techfrendz007.blogspot.com
[>] Verfying if Exploit Successful
```

Insecure Firebase Exploit

Tools

1. Fireprint

- pip install json2html
- apktool.jar
- grep and awk
- chmod +x Fireprint.py

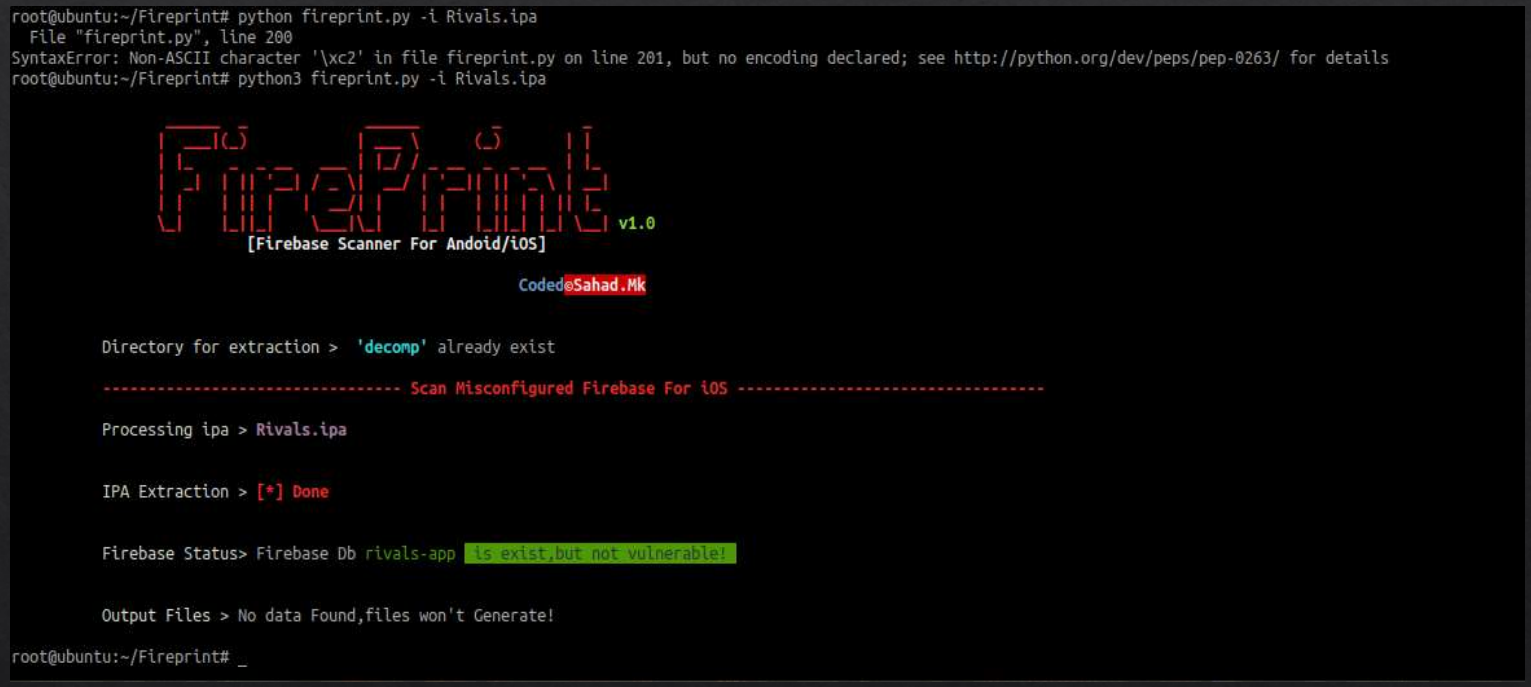
use – python3 Fireprint.py –a Sunil.ipa

2. Python Script

https://github.com/sahad-mk/Fireprint

Fireprint

https://github.com/sahad-mk/Fireprint



https://github.com/shivsahni/FireBaseScanner

Firebase Scanner

Binary Protection

- ✓ Lack of binary protection within the app can help an attacker to analyze the code by reversing the app using tool like IDA pro and even tamper it to do permeant changes.
- ✓ Where we need to look for this
  - ✓ Check for PIE Flag, ARC and stack smashing protection
  - ✓ Check for the shared Library used by the app. Is there any existing bug?
  - ✓ Check whether the app binary is a fat or not
  - ✓ Check for the signed certificate
  - ✓ Check whether the classes are obfuscated or not.

Application Patching (Theory)

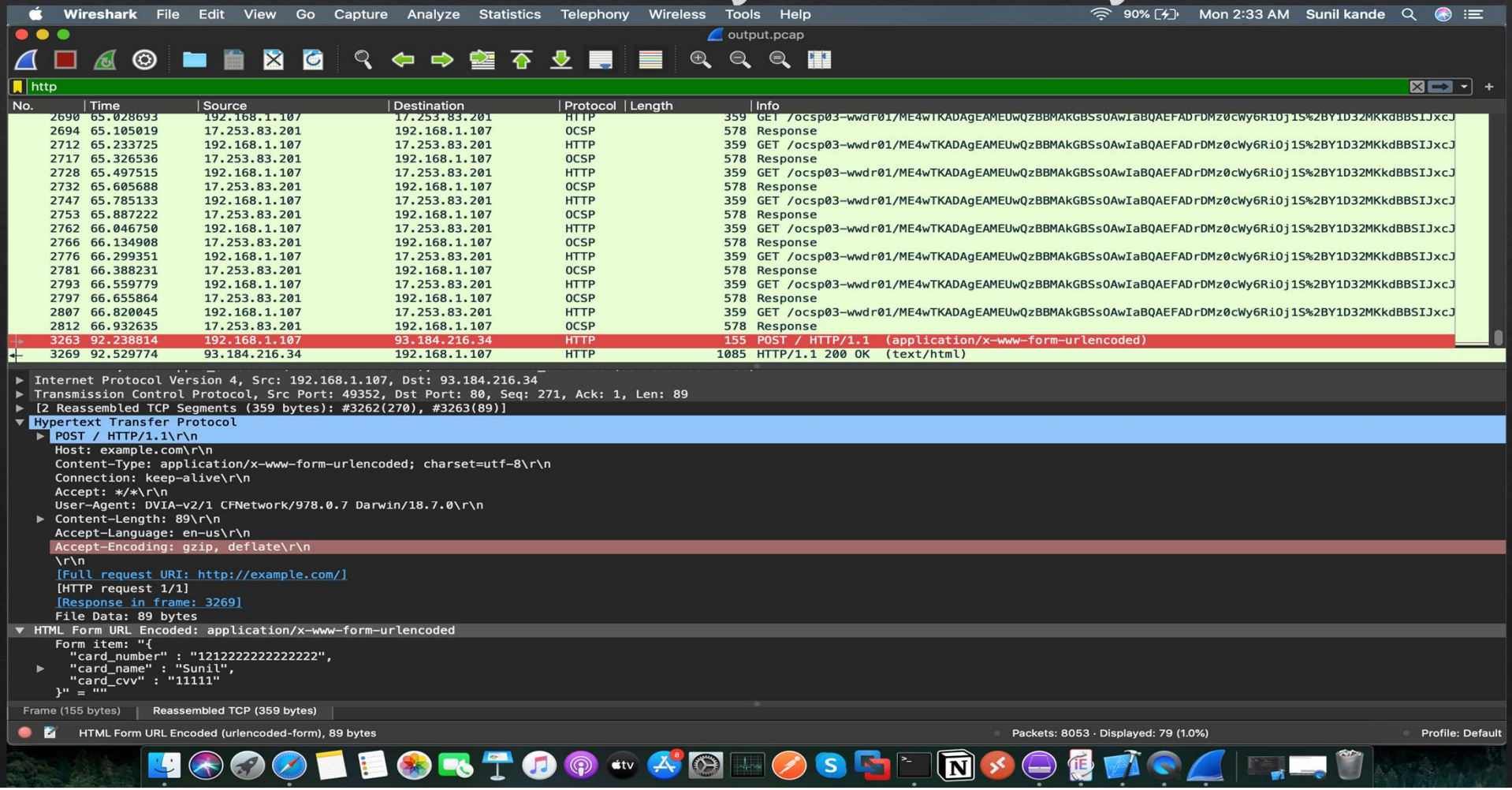
- ✓ Advantage of the application patching is that once the change has been made in the application binary, its permanent.
- ✓ Don't need to do again and again example SSL Pining, Root detection bypass etc.
- ✓ We need IDA Pro or Hooper to reverse the IPA file tamper the code and build it again using xcode or objection tool.
- ✓ Install that app in jailbroken device and use for testing.

Webview Issues

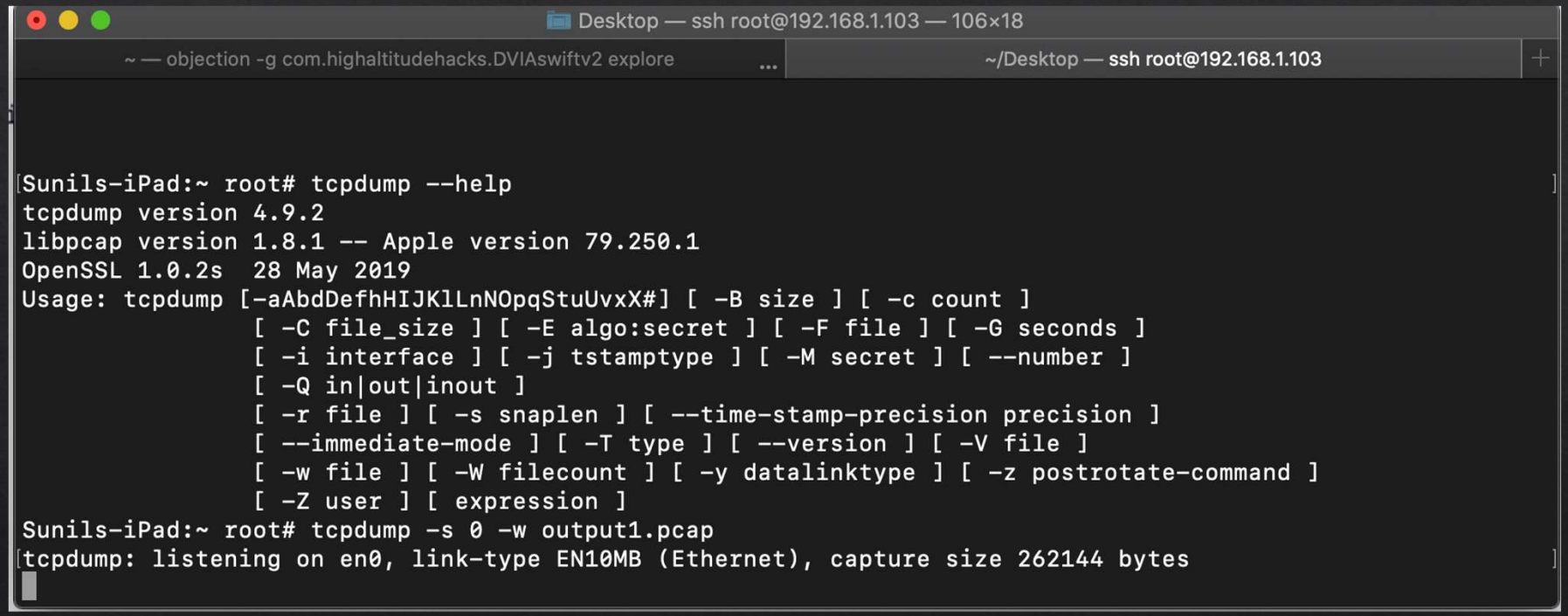
- ✓ URL schemes are used by applications to communicate with each other. Every application can register for a particular url scheme.
- ✓ For e.g, the whats app registers for the url scheme *whatsapp*. This means that any url starting with whatsapp:// protocol will open up the whatsapp application
- ✓ The first step is to find the actual url scheme an application is registered to. This information can be found by looking at the Info.plist file
- ✓ if an application doesn't validate an incoming url properly, it might lead to a security vulnerability

- ✓ https://www.sans.org/blog/insecure-handling-of-url-schemes-in-apples-ios/

Network Layer Security



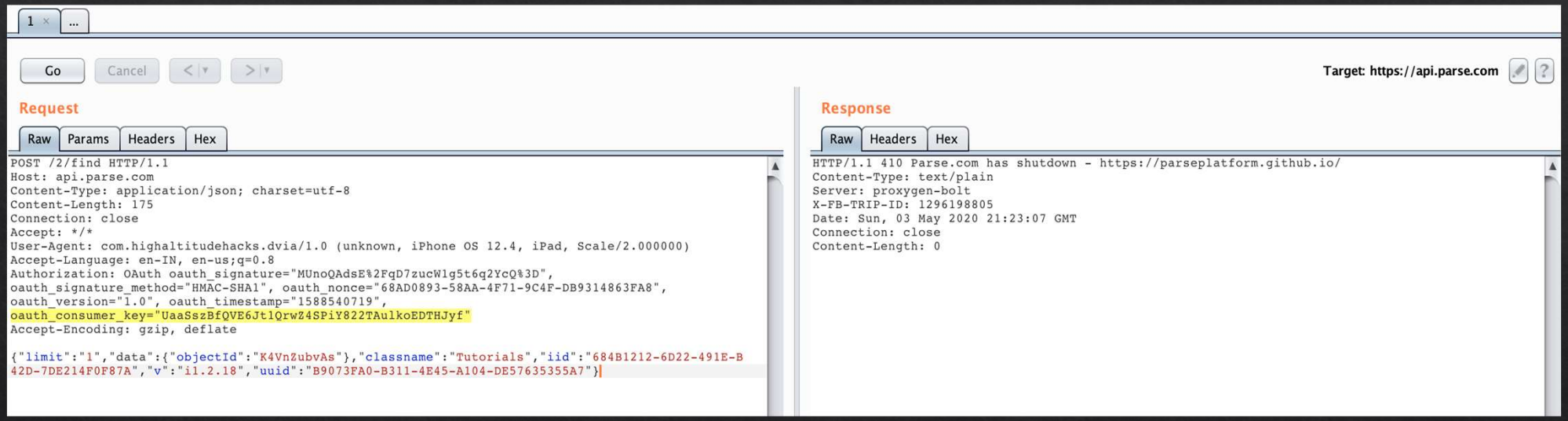
TcpDump and Wireshark



Sensitive Info In Memory

```
[Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Cookies root# strings Cookies.binarycookies ]
cook
Ahighaltitudehacks.com
username
admin123
Ahighaltitudehacks.com
password
dvpassword
Kbplist00
NSHTTPCookieAcceptPolicy
Sunils-iPad:/var/mobile/Containers/Data/Application/210E4318-F2AB-41F8-AC37-094CEDF1D5AF/Library/Cookies root#
```

Data Leakage To Third Party



Ref - https://www.youtube.com/watch?v=X3YxJFiTa6c

Some Useful Resources

- https://appsec-labs.com/ios-attacks-tests/
- https://github.com/sjehutch/development-best-practices/wiki/Pen-test-checklist-IOS
- http://www.droidsec.cn/ios-application-security-testing-cheat-sheet/
- https://docs.google.com/document/d/1N7zMXlFHtWfc00xa6lRHnVB60U4BZO4SbUrWYMbojVM/edit
- https://www.simform.com/mobile-application-security-data-vulnerabilities/