



Mobile Hacking Workshop – iOS

May 28th 2020

Agenda

1. Basics
2. Testing environment
3. Static analysis – 2 labs
4. Data Security – 3 labs
5. Execution analysis – 2 labs
6. Transport Security – 1 lab

Basics

Main steps of a security iOS application assessment

1. **Review** the codebase or **reverse engineer** the binary
2. Run the app on a **jailbroken** device
3. **Inspect** the app via instrumentation
4. **Manipulate** the runtime
5. **MiTM** all the network communications

OWASP Mobile Security Testing Guide

GitHub, Inc. [US] | <https://github.com/OWASP/owasp-mstg>



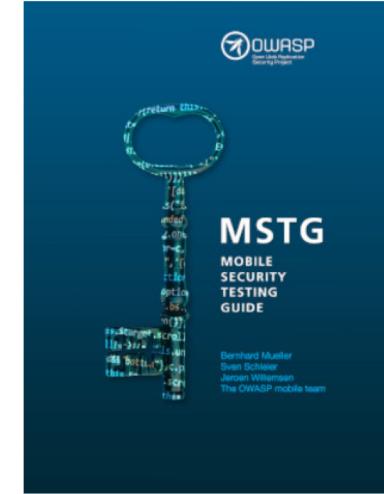
README.md

OWASP Mobile Security Testing Guide

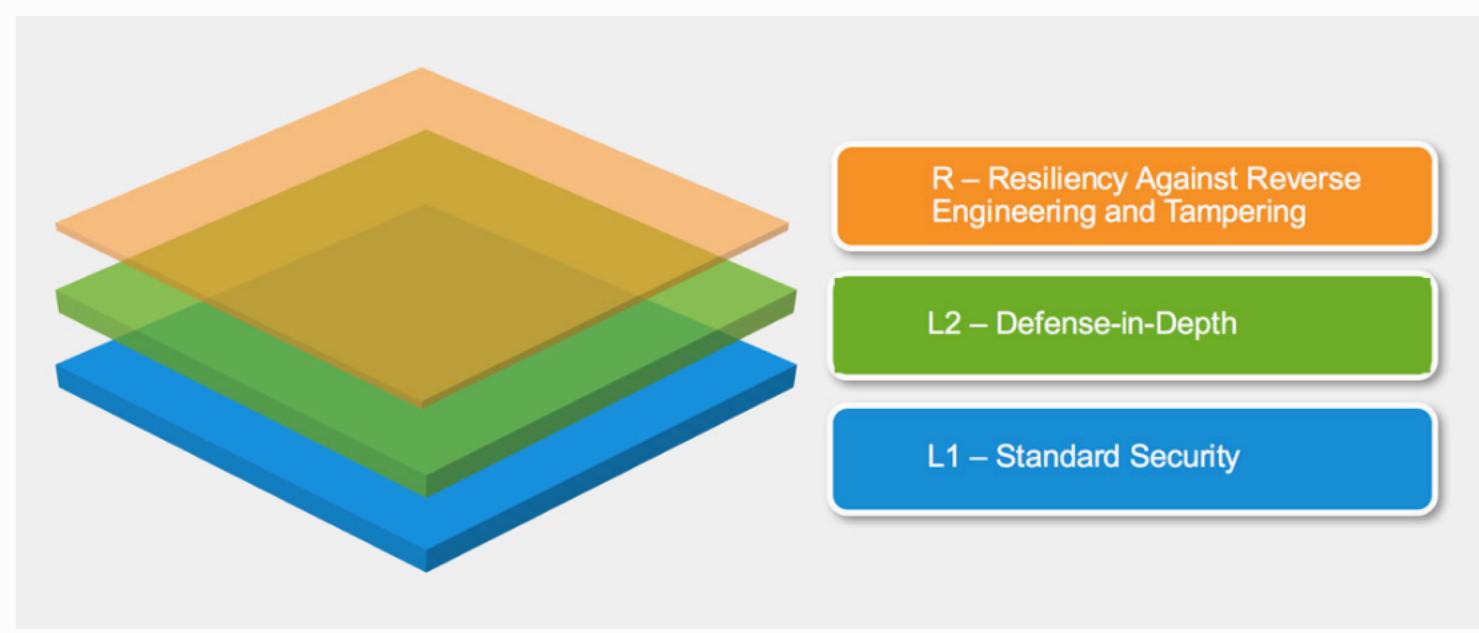
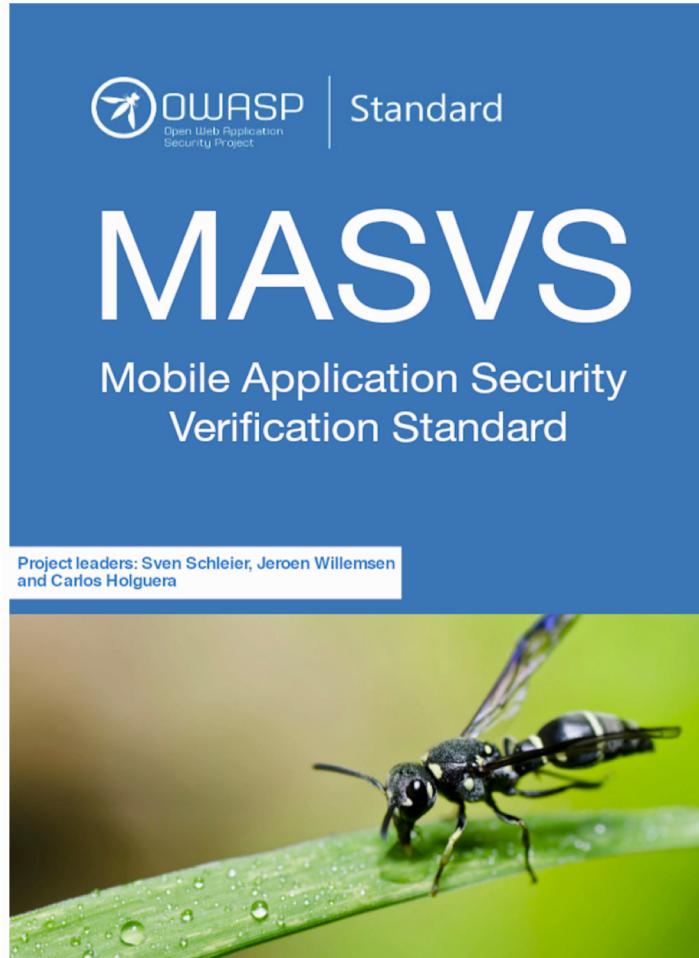


owasp lab project build passing

This is the official GitHub Repository of the OWASP Mobile Security Testing Guide (MSTG). The MSTG is a comprehensive manual for mobile app security testing and reverse engineering. It describes technical processes for verifying the controls listed in the [OWASP Mobile Application Verification Standard \(MASVS\)](#). You can also read the MSTG on [Gitbook](#) or download it as an [e-book](#).



OWASP Mobile AppSec Verification (standard)



**Last version:
version 1.2 (march 2020)**

OWASP Mobile AppSec Checklist

| Mobile Application Security Requirements - iOS | | | | | | |
|--|---|---------|---------|--------|--|--|
| ID | Detailed Verification Requirement | Level 1 | Level 2 | Status | Testing Procedure(s) | |
| V1 | Architecture, design and threat modelling | | | | | |
| 1.1 | All app components are identified and known to be needed. | ✓ | ✓ | - | | |
| 1.2 | Security controls are never enforced only on the client side, but on the respective remote endpoints. | ✓ | ✓ | - | | |
| 1.3 | A high-level architecture for the mobile app and all connected remote services has been defined and security has been addressed in that architecture. | ✓ | ✓ | - | | |
| 1.4 | Data considered sensitive in the context of the mobile app is clearly identified. | ✓ | ✓ | - | | |
| 1.5 | All app components are defined in terms of the business functions and/or security functions they provide. | ✓ | ✓ | N/A - | | |
| 1.6 | A threat model for the mobile app and the associated remote services has been produced that identifies potential threats and countermeasures. | ✓ | ✓ | N/A - | | |
| 1.7 | All security controls have a centralized implementation. | ✓ | ✓ | N/A - | | |
| 1.8 | There is an explicit policy for how cryptographic keys (if any) are managed, and the lifecycle of cryptographic keys is enforced. Ideally, follow a key management standard such as NIST SP 800-57. | ✓ | ✓ | N/A - | | |
| 1.9 | A mechanism for enforcing updates of the mobile app exists. | ✓ | ✓ | N/A - | | |
| 1.10 | Security is addressed within all parts of the software development lifecycle. | ✓ | ✓ | N/A - | | |
| V2 | Data Storage and Privacy | | | | | |
| 2.1 | System credential storage facilities are used appropriately to store sensitive data, such as PII, user credentials or cryptographic keys. | ✓ | ✓ | | Testing For Sensitive Data in Local Data Storage | |
| 2.2 | No sensitive data should be stored outside of the app container or system credential storage facilities. | ✓ | ✓ | | Testing For Sensitive Data in Local Data Storage | |
| 2.3 | No sensitive data is written to application logs. | ✓ | ✓ | | Testing For Sensitive Data in Logs | |
| 2.4 | No sensitive data is shared with third parties unless it is a necessary part of the architecture. | ✓ | ✓ | | Testing Whether Sensitive Data Is Sent To Third Parties | |
| 2.5 | The keyboard cache is disabled on text inputs that process sensitive data. | ✓ | ✓ | | Testing Whether the Keyboard Cache Is Disabled for Text Input Fields | |
| 2.6 | No sensitive data is exposed via IPC mechanisms. | ✓ | ✓ | | Testing Whether Sensitive Data Is Exposed via IPC Mechanisms | |
| 2.7 | No sensitive data, such as passwords or pins, is exposed through the user interface. | ✓ | ✓ | | Testing for Sensitive Data Disclosure Through the User Interface | |
| 2.8 | No sensitive data is included in backups generated by the mobile operating system. | ✓ | ✓ | N/A | Testing for Sensitive Data in Backups | |
| 2.9 | The app removes sensitive data from views when backgrounded. | ✓ | ✓ | N/A | Testing for Sensitive Information in Auto-Generated Screenshots | |
| 2.10 | The app does not hold sensitive data in memory longer than necessary, and memory is cleared explicitly after use. | ✓ | ✓ | N/A | Testing for Sensitive Data in Memory | |
| 2.11 | The app enforces a minimum device-access-security policy, such as requiring the user to set a device passcode. | ✓ | ✓ | N/A | Testing Local Authentication | |
| 2.12 | The app educates the user about the types of personally identifiable information processed, as well as security best practices the user should follow in using the app. | ✓ | ✓ | N/A | Testing user education | |
| V3 | Cryptography | | | | | |
| 3.1 | The app does not rely on symmetric cryptography with hardcoded keys as a sole method of encryption. | ✓ | ✓ | | Verifying Key Management | |

Last version:
version 1.1.3 (August 2019)
So not aligned with the standard !

OWASP - Top10 Mobile Risks

OWASP M1
Improper
Platform Usage

OWASP M2
Insecure
Data Storage

OWASP M3
Insecure
Communication

OWASP M4
Insecure
Authentication

OWASP M5
Insufficient
Cryptography

OWASP M6
Insecure
Authorization

OWASP M7
Client code
quality

OWASP M8
Code
Tampering

OWASP M9
Reverse
engineering

OWASP M10
Extraneous
Functionality

Filesystem: System applications

- /Applications

```
iPhone:/Applications root# ls -al
total 0
drwxr-xr-x  76 root wheel 2432 May 31 06:20 .
drwxr-xr-x  27 root wheel  864 May 31 06:18 ..
drwxrwxr-x  46 root admin 1472 May 21 20:47 AXUIViewService.app/
drwxrwxr-x   7 root admin  224 May 21 20:47 AccountAuthenticationDialog.app/
drwxrwxr-x   7 root admin  224 May 21 20:47 ActivityMessagesApp.app/
drwxrwxr-x   9 root admin  288 May 21 20:47 AdPlatformsDiagnostics.app/
drwxrwxr-x  52 root admin 1664 May 21 20:47 AppStore.app/
drwxrwxr-x  47 root admin 1504 May 21 20:47 AskPermissionUI.app/
drwxrwxr-x   7 root admin  224 May 21 20:47 BusinessExtensionsWrapper.app/
drwxrwxr-x  46 root admin 1472 May 21 20:47 CTCarrierSpaceAuth.app/
drwxrwxr-x  58 root admin 1856 May 21 20:47 Camera.app/
drwxrwxr-x  47 root admin 1504 May 21 20:47 CheckerBoard.app/
drwxrwxr-x  46 root admin 1472 May 21 20:47 CompassCalibrationViewService.app/
drwxrwxr-x   8 root admin  256 May 21 20:46 ContinuityCamera.app/
drwxrwxr-x  49 root admin 1568 May 21 20:46 CoreAuthUI.app/
drwxr-xr-x 124 root wheel 3968 May 31 01:37 Cydia.app/
drwxrwxr-x  46 root admin 1472 May 21 20:46 DDACTIONSERVICE.app/
```

Filesystem: User applications

- /private/var/containers/Bundle/Application

```
iPhone:/private/var/containers/Bundle/Application root# ls -al
total 0
drwxr-xr-x 29 _installld _installld 928 Jun 15 09:48 ../
drwxr-xr-x  6 _installld _installld 192 May 31 01:38 ...
drwxr-xr-x  5 _installld _installld 160 Jun 10 03:54 07B03D18-01A1-4F1E-A355-F000AC9B9F35/
drwxr-xr-x  5 _installld _installld 160 May 31 01:39 121F8420-4F80-4398-8C22-240CEFEF6F54/
drwxr-xr-x  5 _installld _installld 160 May 31 01:39 1E1ACAAC-CD00-47D2-BD0D-68A23A68A85A/
drwxr-xr-x  5 _installld _installld 160 May 31 02:19 277F19A6-D521-48ED-B75E-9D1E8FCA8C90/
drwxr-xr-x  5 _installld _installld 160 May 31 01:52 27FFB5C0-7872-4552-894D-D2374A0ACDD9/
drwxr-xr-x  5 _installld _installld 160 Jun 10 03:51 2E2449F1-AE2A-44A1-8A71-2FBF132852BD/
drwxr-xr-x  5 _installld _installld 160 May 31 01:38 43B016BC-5984-45A5-A4A5-B315E5046993/
drwxr-xr-x  5 _installld _installld 160 May 31 01:39 62784C25-81EB-4CCA-9BDE-C71AE162EA0A/
drwxr-xr-x  5 _installld _installld 160 May 31 01:39 65A879DB-CC9B-4AC6-8DB9-EFABC3DCDF90/
drwxr-xr-x  5 _installld _installld 160 May 31 01:39 7D73F79E-78CF-487B-8C9E-7A3B15CE13E0/
drwxr-xr-x  5 _installld _installld 160 May 31 01:38 81A56A73-B663-4566-8687-2EBE4C04ADD7/
```

Filesystem: User applications: Bundle Directory

- /private/var/containers/Bundle/Application/UUID

```
-rw-r--r-- 1 _installd _installd 2778 Jul  7 2017 about.html
-rw-r--r-- 1 _installd _installd  372 Jul  7 2017 archived-expanded entitlements.xcent
-rw-r--r-- 1 _installd _installd 4096 Jul  7 2017 articles.sqlite
-rw-r--r-- 1 _installd _installd 8167 Jul  7 2017 embedded.mobileprovision
drwxr-xr-x 5 _installd _installd 160 Jul  7 2017 en.lproj/
-rwxr-xr-x 1 _installd _installd 6136800 May 31 01:52 iGoat*
-rw-r--r-- 1 _installd _installd  735 Jul  7 2017 iGoatSSLServer.der
-rw-r--r-- 1 _installd _installd  283 Jul  7 2017 igoat.css
-rw-r--r-- 1 _installd _installd 22687 Jul  7 2017 iphone.png
-rw-r--r-- 1 _installd _installd  327 Jul  7 2017 mansi.html
-rw-r--r-- 1 _installd _installd  463 Jul  7 2017 rutger.html
-rw-r--r-- 1 _installd _installd  817 Jul  7 2017 splash.html
iPhone:/private/var/containers/Bundle/Application/27FFB5C0-7872-4552-894D-D2374A0ACDD9/iGoat.app root#
```

- Binary and static files (signed)

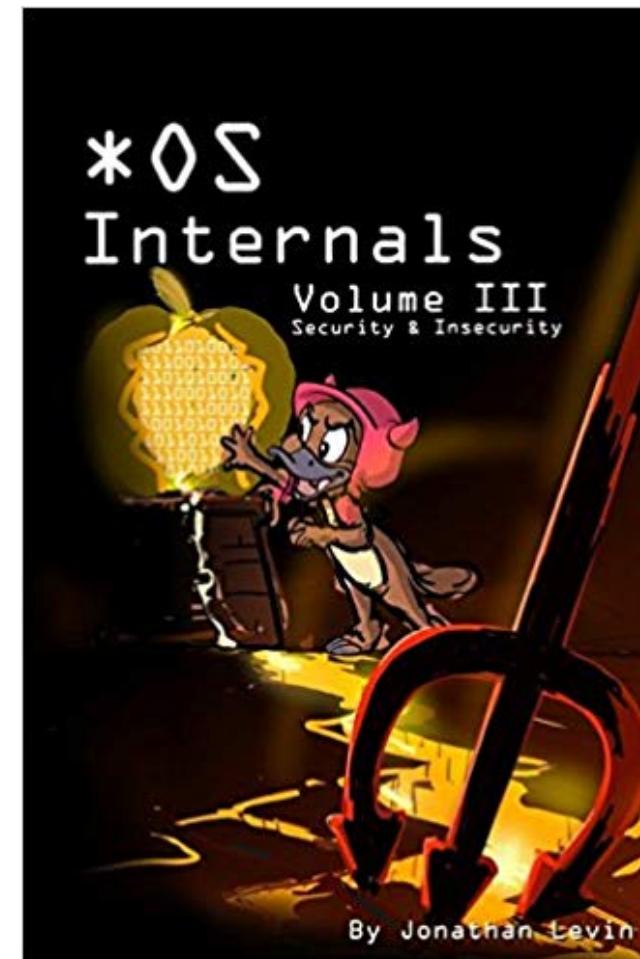
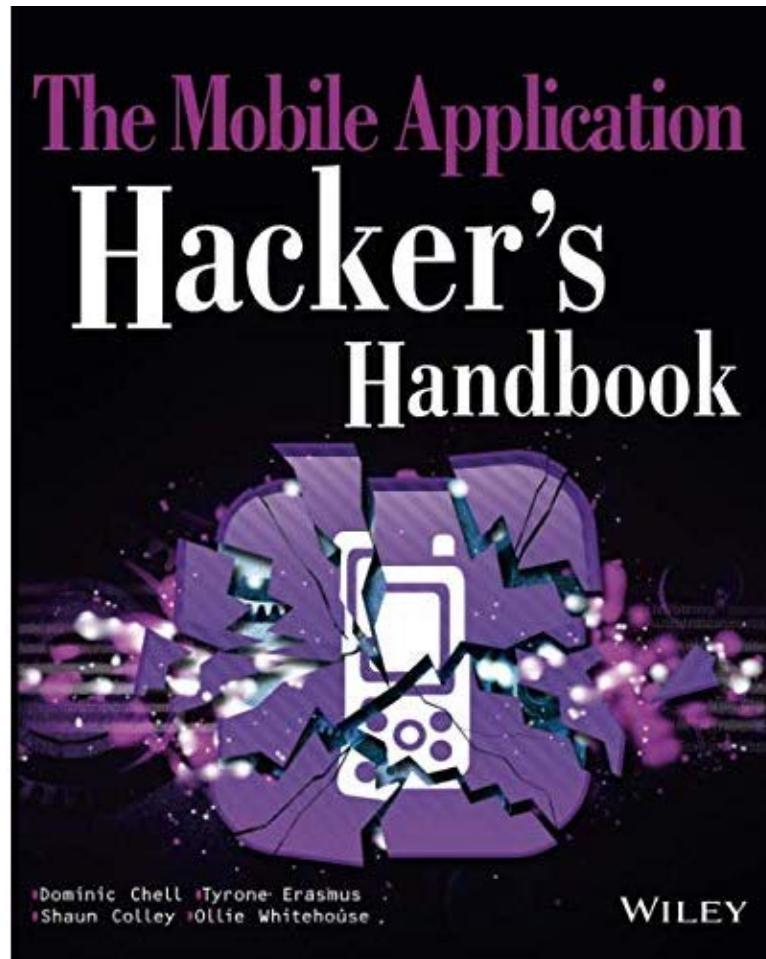
Filesystem: User applications: Data Directory

- /private/var/mobile/Containers/Data/Application/UUID

```
iPhone:/private/var/mobile/Containers/Data/Application/AA7D5264-12B6-4109-A397-C007299AB958 root# ls -al
total 4
drwxr-xr-x  7 mobile  mobile  224 May 31 01:52 .
drwxr-xr-x 81 mobile  mobile 2592 Jun 15 09:48 ../
-rw-r--r--  1 root   mobile  211 May 31 01:52 .com.apple.mobile_container_manager.metadata.plist
drwxr-xr-x  3 mobile  mobile   96 Jun 11 10:45 Documents/
drwxr-xr-x  5 mobile  mobile  160 Jun 11 02:14 Library/
drwxr-xr-x  2 mobile  mobile   64 May 31 01:52 SystemData/
drwxr-xr-x 25 mobile  mobile  800 Jun 15 16:25 tmp/
iPhone:/private/var/mobile/Containers/Data/Application/AA7D5264-12B6-4109-A397-C007299AB958 root# du -ah
12K    ./Documents/credentials.sqlite
12K    ./Documents
4.0K   ./com.apple.mobile_container_manager.metadata.plist
52K    ./Library/Caches/com.swaroop.iGoat/Cache.db
0      ./Library/Caches/com.swaroop.iGoat/Cache.db-wal
32K    ./Library/Caches/com.swaroop.iGoat/Cache.db-shm
84K    ./Library/Caches/com.swaroop.iGoat
40K    ./Library/Caches/Snapshots/com.swaroop.iGoat/600E684E-7B24-4386-947A-FDF646E30248@2x.atx
36K    ./Library/Caches/Snapshots/com.swaroop.iGoat/downscaled/D29866C1-7955-4271-B34A-5F63AF19BB65@2x.atx
36K    ./Library/Caches/Snapshots/com.swaroop.iGoat/downscaled
76K    ./Library/Caches/Snapshots/com.swaroop.iGoat
76K    ./Library/Caches/Snapshots
160K   ./Library/Caches
0      ./Library/Preferences
4.0K   ./Library/Cookies/com.swaroop.iGoat.binarycookies
```

- User data, settings, cookies, cached & temp files

To go further



Testing environment

Setup – Intro

- For this workshop, we are going to use

- A VM: **Mobexler Lite**

- <https://mobexler.com/download.htm>

- With tools:

- Cydia Impactor
 - Frida
 - Hopper
 - Objection

A **jailbroken device**

With vulnerable apps:

- DVIA
- iGoat

Setup – Mobexler

Default password: 12345

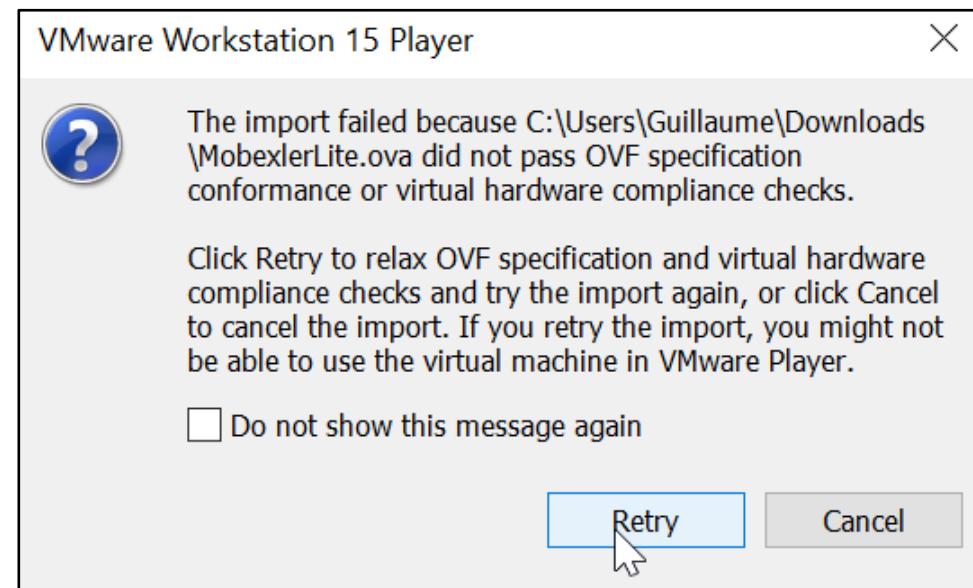
- “A Mobile Application Penetration Testing Platform”
- Similar to a Kali Linux machine but focused on mobile apps
 - Android and iOS
- Setup Guide
 - <https://mobexler.com/setup.htm>
- This machine includes most common tools for testing
 - Burp Suite, **Cydia Impactor**, **Frida**, **Ghidra**, **Hopper**, **Objection**, **MobSF**, etc.
 - <https://mobexler.com/tools.htm>

Setup – Mobexler

- Mobexler and VMWare
 - For this workshop, use only the Mobexler Lite version
 - <https://drive.google.com/file/d/1r2exB-4OyBW0vRKFAPY7ls7ezX2ioXtl/view?usp=sharing>
 - **Don't use Virtual Box!!! Use VMWare Player or Workstation**
 - <https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html>

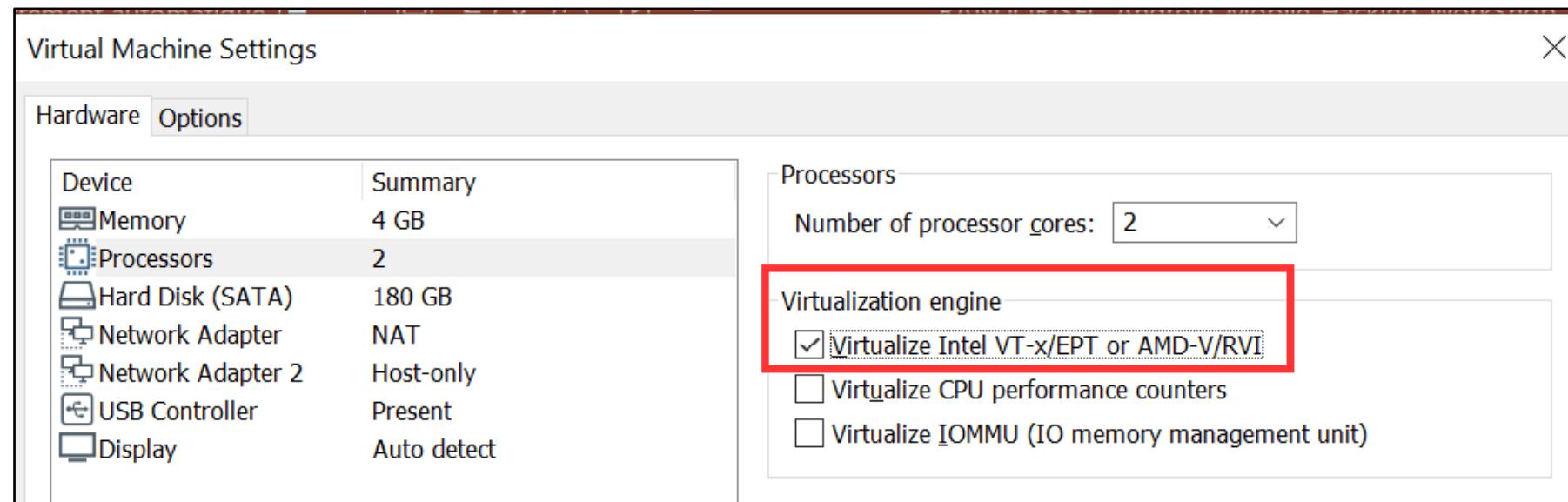
Setup – Mobexler

- Import the OVA file using VMWare
 - If an error message appears saying the OVA file didn't pass the OVF specifications, please click retry. It should work ☺



Setup – Mobexler

- Edit the Virtual Machine hardware settings
 - Memory: At least 4 Go
 - Processors: Enable "Virtualize Intel VT-x/EPT or AMD-V/RVI"



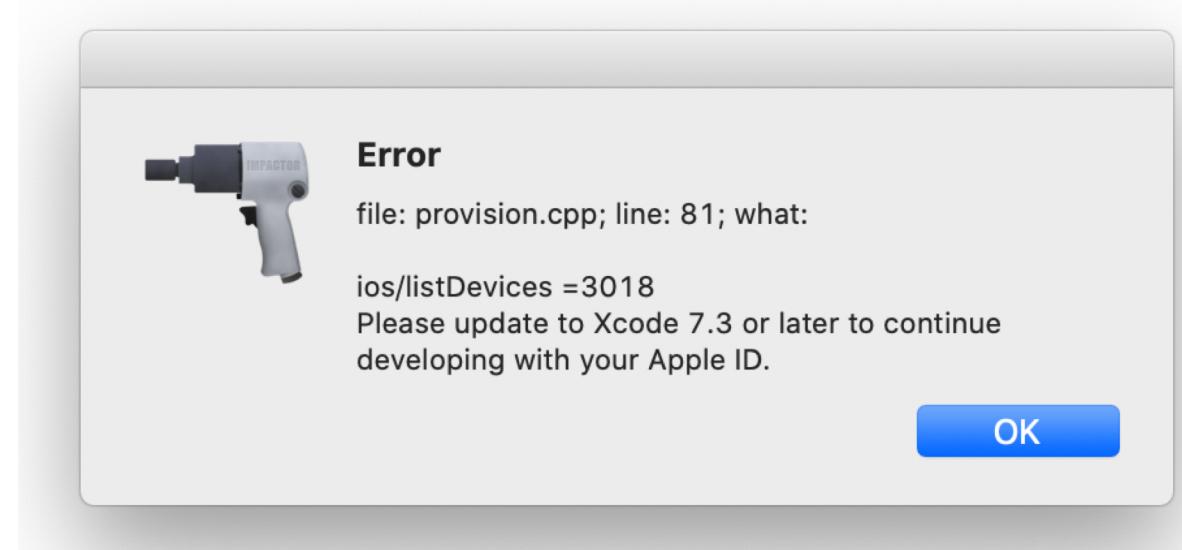
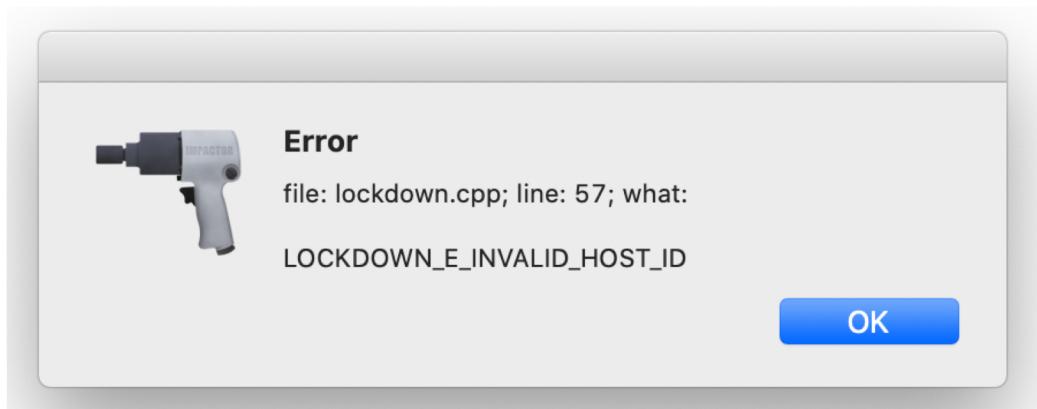
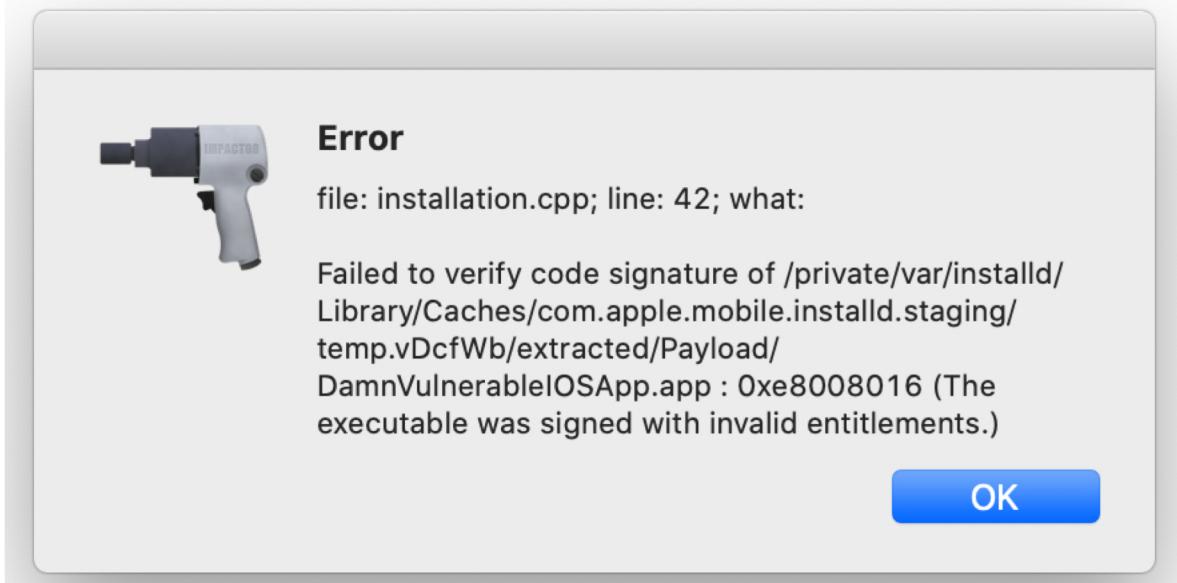
Setup – Mobexler

- Install VMWare Tools (Optional)
 - Install the VMWare tools
 - <https://linuxconfig.org/install-vmware-tools-on-ubuntu-18-04-bionic-beaver-linux>
 - sudo apt install open-vm-tools-desktop
- Install Rvi_capture (Optional)
 - [https://github.com/gh2o/rvi capture](https://github.com/gh2o/rvi_capture)

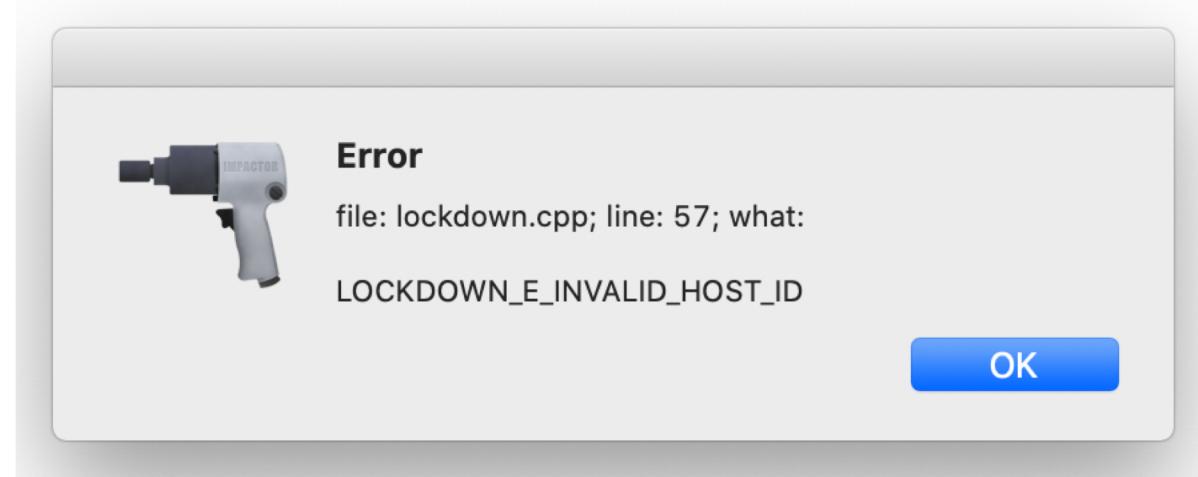
Setup – Vulnerable apps

- Vulnerable Apps
 - Install the following IPA on your jailbroken device using Cydia Impactor:
 - <https://github.com/prateek147/DVIA/blob/master/DamnVulnerableiOSApp.ipa>
 - <https://github.com/OWASP/igoat/blob/develop/iGoat.ipa>

Signing problems

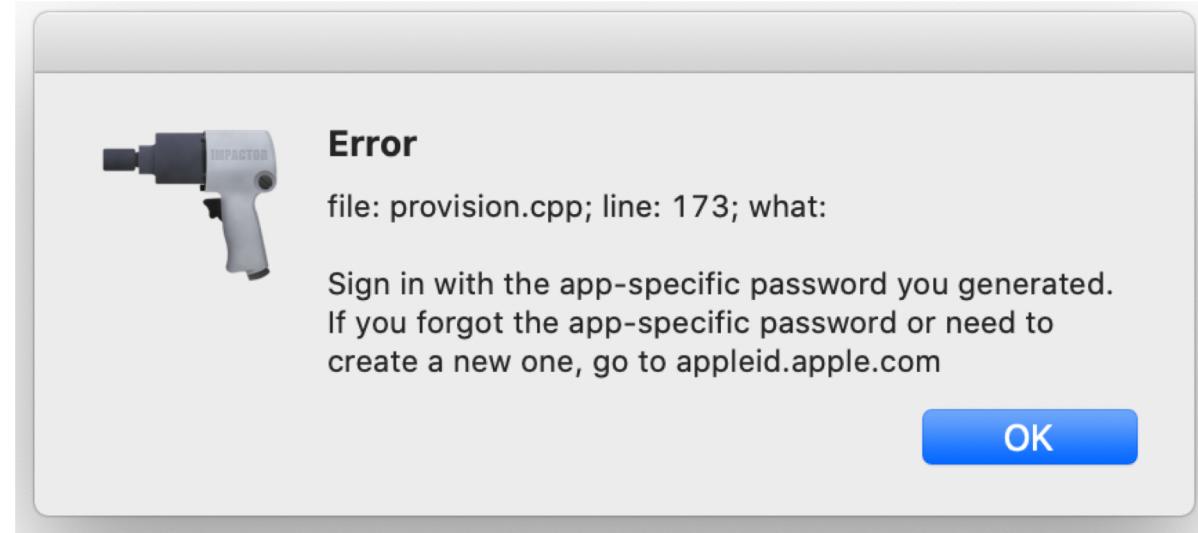


Signing problems: *file: lockdown.cpp; line: 57*



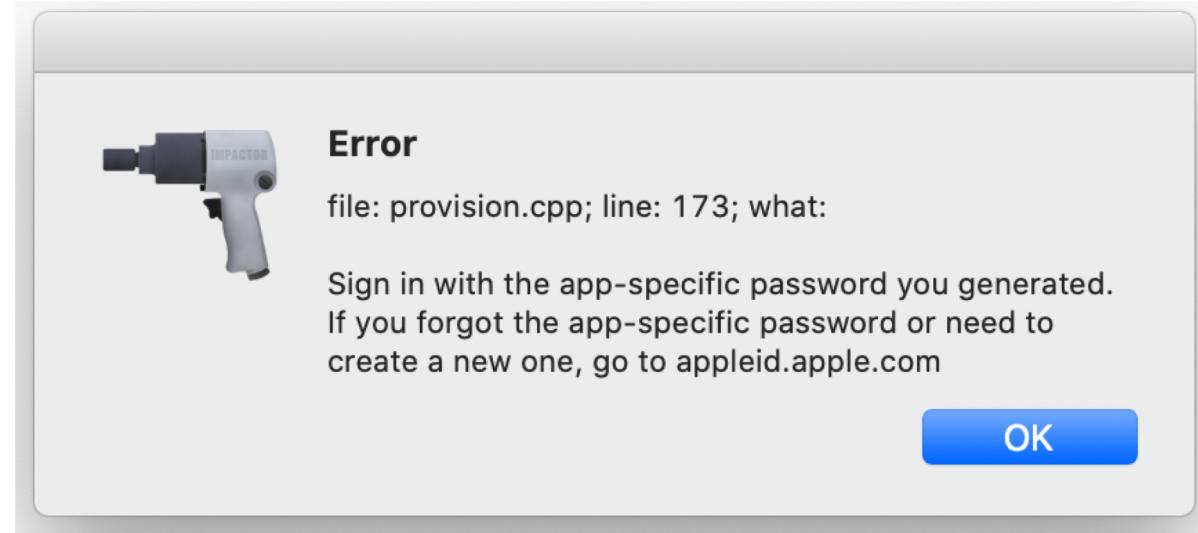
- Trust This Computer ?
- **Select « Trust »**

Signing problems: *file: provision.cpp; line: 81*



- Go to: <https://developer.apple.com/programs/enroll/>
- **Enroll on the Apple Developer Program**

Signing problems: *file: provision.cpp; line: 173*

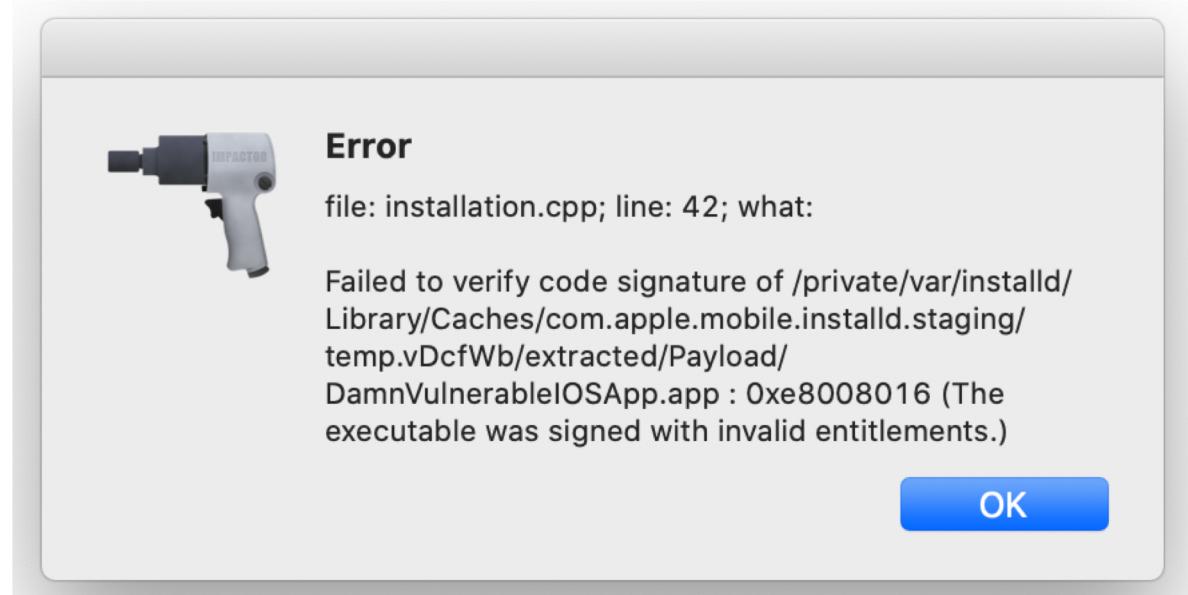


- Go to: <https://appleid.apple.com/account/manage>
- Enable 2FA
- **Generate app-specific password**

Signing problems: *file: installation.cpp; line: 42*



Need a macOS



- Go to: <https://developer.apple.com/>
- Create a **provisioning profile**
- Sign your IPA with it

Signing problems: *file: installation.cpp; line: 42*



Certificates, Identifiers & Profiles

Need a macOS

[◀ All Profiles](#)

Register a New Provisioning Profile

Development

- iOS App Development**
Create a provisioning profile to install development apps on test devices.
- tvOS App Development**
Create a provisioning profile to install development apps on tvOS test devices.
- macOS App Development**
Create a provisioning profile to install development apps on test devices.

Distribution

- Ad Hoc**
Create a distribution provisioning profile to install your app on a limited number of registered devices.

Signing problems: *file: installation.cpp; line: 42*



Need a macOS

```
applesign -i <identity> -m <provision_profile> -o <signed.ipa> <original.ipa>
```

```
Unzipping /Users/davydouhine/Documents/iOS/ipa/dvia.ipa
Payload found
Main IPA executable is not encrypted
Embedding new mobileprovision
{"application-identifier":"T9T8LG2CVR.*", "keychain-access-groups":["T9T8LG2CVR.*"], "get-task-allow":false, "com.apple.developer.team-identifier":"T9T8LG2CVR"}
Updated binary entitlements/Users/davydouhine/Documents/iOS/ipa/dvia.ipa.ae9b005e-61f9-4a59-84ce-67f61334c2c0/Payload/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp.entitlements
Signing libraries and frameworks
Executable found at /Users/davydouhine/Documents/iOS/ipa/dvia.ipa.ae9b005e-61f9-4a59-84ce-67f61334c2c0/Payload/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
Resolving signing order using layered list
Signed /Users/davydouhine/Documents/iOS/ipa/dvia.ipa.ae9b005e-61f9-4a59-84ce-67f61334c2c0/Payload/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
Verifying /Users/davydouhine/Documents/iOS/ipa/dvia.ipa.ae9b005e-61f9-4a59-84ce-67f61334c2c0/Payload/DamnVulnerableIOSApp.app/DamnVulnerableIOSApp
Zipifying into /Users/davydouhine/Documents/iOS/ipa/dvia_signed.ipa ...
Cleaning up /Users/davydouhine/Documents/iOS/ipa/dvia.ipa.ae9b005e-61f9-4a59-84ce-67f61334c2c0
Target is now signed: dvia_signed.ipa
```

Static analysis

Static analysis

- Needs:
 - full source code !
 - tools (Checkmarx , Sonar, Clang Static Analyzer, ...) !
- But
 - a simple “grep” can help
 - even **without the full source code a lot can be done by reverse engineering the static files (binary, etc.)**

OWASP M2
Insecure
Data Storage

OWASP M3
Insecure
Communication

OWASP M5
Insufficient
Cryptography

OWASP M7
Client code
quality

OWASP M10
Extraneous
Functionality

Static analysis

To detect:

- Caching
- Cleartext credentials storage
- Back-end servers
- Bad pasteboard management
- SQL injection
- Code injection
- Verbose logging
- Vulnerable C function
- Custom URL schemes aka Deeplinks (e.g: cydia://)

OWASP M2
Insecure
Data Storage

OWASP M3
Insecure
Communication

OWASP M5
Insufficient
Cryptography

OWASP M7
Client code
quality

OWASP M10
Extraneous
Functionality

Lab1: App metadata

Goal: analyze Info.plist

OWASP M2
Insecure data
storage



Steps:

1. Unzip Headbook.ipa on your VM (you don't need to install it on your device)
2. Analyze Info.plist
3. Submit the flag to <https://ctf.ivrodriguez.com>

Lab2: Custom URL Scheme (Deep link)

Goal: find and use DVIA deep link

OWASP M2
Insecure data
storage



Steps:

1. Open DVIA
2. Go to “**Security Decisions via untrusted input**”
3. Click on “Start Challenge”

Lab2: Custom URL Scheme (Deep link)

Goal: find and use DVIA deep link

OWASP M2
Insecure data
storage



Testing Custom URL Schemes (MSTG-PLATFORM-3)

Overview

Custom URL schemes [allow apps to communicate via a custom protocol](#). An app must declare support for the schemes and handle incoming URLs that use those schemes.

Apple warns about the improper use of custom URL schemes in the [Apple Developer Documentation](#):

URL schemes 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 do not risk the user's data. For example, do not allow other apps 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.

Source: <https://github.com/OWASP/owasp-mstg/blob/master/Document/0x06h-Testing-Platform-Interaction.md>

Lab2: Custom URL Scheme (Deep link)

Goal: find and use DVIA deep link

OWASP M2
Insecure data
storage



Steps:

1. Analyze Info.plist

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleTypeRole</key>
        <string>None</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>dvia</string>
        </array>
    </dict>
</array>
```

Lab2: Custom URL Scheme (Deep link)

Goal: find and use DVIA deep link

OWASP M2
Insecure data
storage



Steps:

2. Analyze DVIA binary

A screenshot of a debugger interface showing a search result for "openURL". The search bar at the top contains "openURL". Below the search bar is a "Tag Scope" button. The main area displays a table with three columns: "Address", "Type", and "Name". A single row is shown, with the "Address" column containing "0xfacc", the "Type" column containing "P", and the "Name" column containing "-[AppDelegate application:openURL:sourceApplication:annotation:]".

| Address | Type | Name |
|---------|------|--|
| 0xfacc | P | -[AppDelegate application:openURL:sourceApplication:annotation:] |

Lab2: Custom URL Scheme (Deep link)

Goal: find and use DVIA deep link

OWASP M2
Insecure data storage



```
/* @class AppDelegate */
-(char)application:(void *)arg2 openURL:(void *)arg3 sourceApplication:(void *)arg4 annotation:(void *)arg5 {
    r7 = (sp - 0x14) + 0xc;
    sp = sp - 0x38;
    r6 = self;
    r11 = [arg3 retain];
    r7 = r7;
    r5 = [[r11 absoluteString] retain];
    if (r5 != 0x0) {
        [sp + 0x10 rangeOfString:r5, @"/call_number/"];
        if (var_20 == (0x80000000 ^ 0xffffffff)) {
            r4 = 0x0;
        }
    } else {
        r10 = [[r6 getParameters:r11] retain];
        r0 = [r10 objectForKey:@"phone"];
        r7 = r7;
        r0 = [r0 retain];
        [r0 release];
        if (r0 != 0x0) {
            var_24 = [UIAlertView alloc];
            r8 = [[r10 objectForKey:@"phone"] retain];
            r4 = [[NSString stringWithFormat:@"Calling %@ without validation. Ring Ring !"] retain];
            r6 = [var_24 initWithTitle:@"Success" message:r4 delegate:0x0 cancelButtonTitle:@"OK" otherButtonTitles:0x0];
            [r6 show];
            [r6 release];
            [r4 release];
            [r8 release];
        }
        [r10 release];
        r4 = 0x1;
    }
}
```

Data security

Lab3: Cleartext Property List Files (Plist)

Goal: find credentials stored in cleartext

OWASP M2
Insecure data
storage



Steps:

1. Open DVIA
2. Go to “**Insecure Data Storage**”
3. Click “Plist”
4. Enter credentials
5. Click “Save in Plist file”
6. The credentials are stored on your device
7. Get them with Objection

Lab4: Keychain access

Goal: find credentials stored in the keychain

OWASP M2
Insecure data
storage



Steps:

1. Open DVIA
2. Go to “**Insecure Data Storage**”
3. Click “Keychain”
4. Enter credentials
5. Click “Save in Keychain”
6. The credentials are stored on your device
7. Get them with Objection

Syslog

```

Jan 6 17:35:45 iPad-de-davy kernel[0] <Notice>: xpcproxy[80834] Builtin profile: container (sandbox)
Jan 6 17:35:45 iPad-de-davy kernel[0] <Notice>: xpcproxy[80834] Container: /private/var/mobile/Containers/Data/Application/237ED97E-6F71-4650-B215-55C8E3831738
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Notice>: MS:Notice: Injecting: com.outfit7.mytalkingtom [mytalkingtom] (1280.38)
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Error>: MS:Error: unable to open() binary file
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Notice>: MS:Notice: Loading: /Library/MobileSubstrate/DynamicLibraries/SSLKillSwitch2.dylib
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Warning>: === SSL Kill Switch 2: Preference set to 1.
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Warning>: === SSL Kill Switch 2: Substrate hook enabled.
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Error>: MS:Error: binary does not support this cpu type
Jan 6 17:35:46 iPad-de-davy mytalkingtom[80834] <Error>: MS:Error: failure to check trustme.dylib
Jan 6 17:35:46 iPad-de-davy gamecontrollerd[80836] <Notice>: MS:Notice: Injecting: (null) [gamecontrollerd] (1280.38)
Jan 6 17:35:47 iPad-de-davy gamecontrollerd[80836] <Error>: MS:Error: unable to open() binary file
Jan 6 17:35:47 iPad-de-davy gamecontrollerd[80836] <Warning>: === SSL Kill Switch 2: Preference set to 1.
Jan 6 17:35:47 iPad-de-davy gamecontrollerd[80836] <Warning>: === SSL Kill Switch 2: Substrate hook enabled.
Jan 6 17:35:47 iPad-de-davy gamecontrollerd[80836] <Error>: MS:Error: binary does not support this cpu type
Jan 6 17:35:47 iPad-de-davy gamecontrollerd[80836] <Error>: MS:Error: failure to check trustme.dylib
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: -> registered mono modules 0x10208f9f0
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: You've implemented -[ application:didReceiveRemoteNotification:fetchCompletionO
ote-notification" to the list of your supported UIBackgroundModes in your Info.plist.
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Notice>: <FIRAnalytics/INFO> Firebase Analytics v.3404000 started
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Notice>: <FIRAnalytics/INFO> Successfully created Firebase Analytics App Delegate Proxy automatically. To disable
egateProxyEnabled to NO in the Info.plist
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: AppsFlyer SDK version 4.6.3 started build (521)
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: AppInit: r=1
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: O7FunNetworkLib Version: 5.10.0
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: *** -[NSKeyedUnarchiver initForReadingWithData:]: data is NULL
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Warning>: -canOpenURL: failed for URL: "fb://" - error: "(null)"
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Notice>: <FIRAnalytics/INFO> Firebase Analytics enabled
Jan 6 17:35:47 iPad-de-davy mytalkingtom[80834] <Notice>: <FIRAnalytics/INFO> Firebase Analytics enabled

```

Syslog

To read the syslog you first need to get the device UDID

OWASP M2
Insecure data
storage



Get device UDID (serial number) on Linux:

```
lsusb -s :`lsusb | grep iPhone | cut -d ' ' -f 4 | sed  
's/://'\` -v | grep iSerial | awk '{print $3}''
```

Get device UDID (serial number) on macOS:

```
idevice_id -l
```

or

```
ioreg -p IOUSB -l | grep "USB Serial"
```

(Don't copy the lsusb command from the slide, check
[*ios_cheatsheet.txt*](#))

Lab5: Syslog

Goal: get info stored in the syslog

OWASP M2
Insecure data
storage



Steps:

1. Open **iGoat**
2. Go to “Key Management / **Random Key Generation**”
3. Find the encryption key
4. Get the syslog with Impactor:

Linux:

```
./Impactor idevicesyslog -u <UDID>
```

macOS:

```
./Applications/Impactor.app/Contents/MacOS/Impactor  
idevicesyslog -u <UDID>
```

Execution analysis

Hooking

cycrypt

Cycript allows developers to explore and modify running applications on either iOS or Mac OS X using a hybrid of Objective-C++ and JavaScript syntax through an interactive console that features syntax highlighting and tab completion.

(It also runs standalone on Android and Linux and provides access to Java, but without injection.)

FRIDA

[OVERVIEW](#) [DOCS](#) [NEWS](#) [CODE](#) [CONTACT](#)

Inject JavaScript to explore native apps on Windows, macOS, Linux, iOS, Android, and QNX.

Frida

```
bibi:~ root# frida -U -p 81987
    ___
   / _ \ |  Frida 10.2.1 - A world-class dynamic instrumentation framework
  | ( ) |
 > _ _ |  Commands:
/_/ \_ \|  help      -> Displays the help system
 . . . . object?   -> Display information about 'object'
 . . . . exit/quit -> Exit
 . . . .
 . . . . More info at http://www.frida.re/docs/home/
```

```
[USB:::iPad 4:::81987]->
```

- ApiResolver
- Arm64Relocator
- Arm64Writer
- Array
- ArrayBuffer
- Backtracer
- Boolean

Frida: native API tracing

```
bibi:~ root# frida-trace -U -p 82924 -i "*URL*"  
Instrumenting functions...  
CaptiveCopyWiFiLandingPageURL: Auto-generated handler at "/private/var/root/__handlers__/CaptiveNetwork/CaptiveCopyWiFiLandingPageURL.js"  
CFURLCreateBookmarkData: Auto-generated handler at "/private/var/root/__handlers__/Foundation/CFURLCreateBookmarkData.js"  
_CFURLCreateDisplayPathComponentsArray: Auto-generated handler at "/private/var/root/__handlers__/Foundation/_CFURLCreateDisplayPathComponentsArray.js"  
_CFBundleCreateWithExecutableURLIfMightBeBundle: Auto-generated handler at "/private/var/root/__handlers__/Foundation/_CFBundleCreateWithExecutableURL_3b8f9203.js"  
_CFURLCreateByResolvingAliasFile: Auto-generated handler at "/private/var/root/__handlers__/Foundation/_CFURLCreateByResolvingAliasFile.js"  
_CFURLCreateFromComponents: Auto-generated handler at "/private/var/root/__handlers__/Foundation/_CFURLCreateFromComponents.js"  
CFBundleCopyExecutableURL: Auto-generated handler at "/private/var/root/__handlers__/Foundation/CFBundleCopyExecutableURL.js"  
_CFURLIsItemPromiseAtURL: Auto-generated handler at "/private/var/root/__handlers__/Foundation/_CFURLIsItemPromiseAtURL.js"  
CFCopyHomeDirectoryURL: Auto-generated handler at "/private/var/root/__handlers__/Foundation/CFCopyHomeDirectoryURL.js"  
CFURLCreateFilePathURL: Auto-generated handler at "/private/var/root/__handlers__/Foundation/CFURLCreateFilePathURL.js"
```

(...)

```
CMByteStreamCreateForFileURL: Auto-generated handler at "/private/var/root/__handlers__/CoreMedia/CMByteStreamCreateForFileURL.js"  
FigNote_FlushRunningLogAndCopyURLContainingLogs: Auto-generated handler at "/private/var/root/__handlers__/CoreMedia/FigNote_FlushRunningLogAndCopyUR_0e23347a.js"  
Started tracing 1165 functions. Press Ctrl+C to stop.
```

/* TID 0xc07 */

```
19170 ms CFURLCopyScheme()  
19171 ms CFURLCopyAbsoluteURL()  
19171 ms CFURLCopyFileSystemPath()  
19182 ms CFURLCopyScheme()  
19184 ms CFURLCopyAbsoluteURL()  
19184 ms CFURLCopyFileSystemPath()  
19214 ms CFBundleCopyResourceURL()  
19215 ms | CFURLCreateWithFileSystemPath()  
19215 ms | | CFURLCreateWithFileSystemPathRelativeToBase()
```

Frida: CodeShare

Frida CodeShare [Twitter](#) [GitHub](#)

[Log In](#)

Unleash the power of Frida.

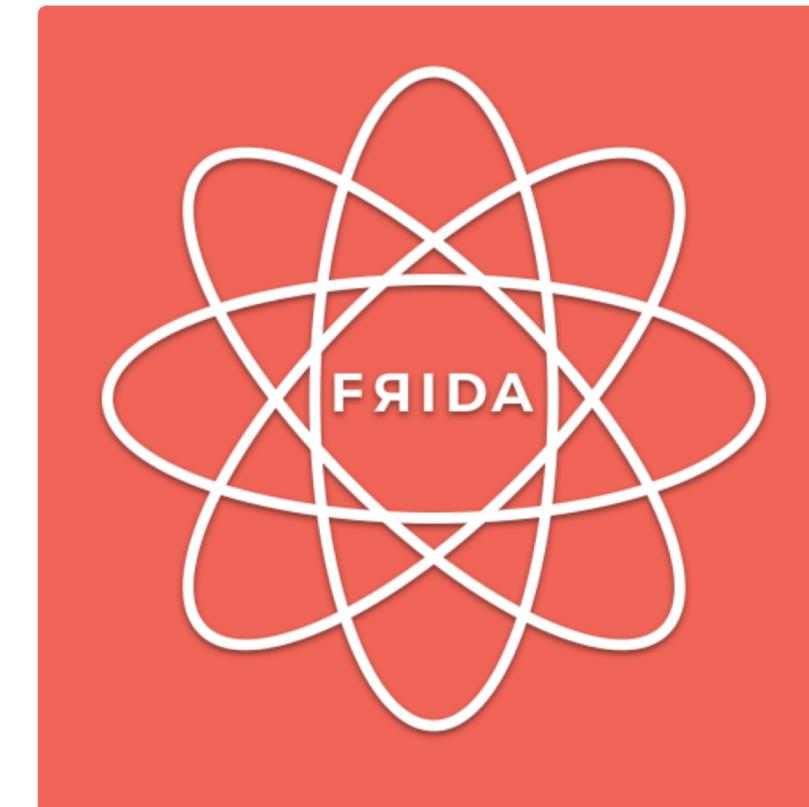
"If I have seen further, it is by standing on the shoulders of giants."

-Sir Issac Newton

The Frida CodeShare project is comprised of developers from around the world working together with one goal - push [Frida](#) to its limits in new and innovative ways.

Frida has [amazing potential](#), but needed a better forum to share ideas, so we've put together CodeShare to help stoke innovation and push Frida to its highest potential!

[BROWSE CODE](#)



Frida: CodeShare

iOS DataProtection

thumb up 7 | eye 4K

Uploaded by: [@ay-kay](#)

List iOS file data protection classes (NSFileProtectionKey) of an app

[PROJECT PAGE](#)

aesinfo

thumb up 7 | eye 5K

Uploaded by: [@dzonterzy](#)

Show useful info about AES encryption/decryption at application runtime

[PROJECT PAGE](#)

frida-multiple-unpinning

thumb up 5 | eye 4K

Uploaded by: [@akabel](#)

Another Android ssl certificate pinning bypass script for various methods
<https://gist.github.com/akabe1/5632cbc1cd49f0237cbd0a93bc8e4452>

[PROJECT PAGE](#)

ObjC method observer

thumb up 4 | eye 8K

Uploaded by: [@mrmacete](#)

Observe all method calls to a specific class (e.g. observeClass('LicenseManager')) , or dynamically resolve methods to observe using ApiResolver (e.g. observeSomething('*[* *Password:*'])). The script tries to do its best to resolve and display input parameters and return value. Each call log comes with its stacktrace.

[PROJECT PAGE](#)

Frida: ObjC-method-observer

```
bibi:~ root# frida -U iGoat --codeshare mrmacete/objc-method-observer
    ___
   / _ \  Frida 10.2.1 - A world-class dynamic instrumentation framework
  | C_ \ |
 > _ | Commands:
 / \_ |     help      -> Displays the help system
 . . . . object?    -> Display information about 'object'
 . . . . exit/quit -> Exit
 . . . .
 . . . . More info at http://www.frida.re/docs/home/

[USB::iPad 4::iGoat]-> observeSomething('*[* *Pin:*]');
Observing -[_MKPlaceViewController actionsViewControllerRemovePin:]
Observing -[_MKPlaceViewController setShowRemovePin:]
Observing -[MKPlaceHeaderBackgroundView_Snapshot setShowsPin:]
Observing -[BruteForceRuntimeVC validatePin:]
Observing -[MKPlaceActionsViewController setShowRemovePin:]
undefined
[USB::iPad 4::iGoat]-> (0x140264960)  -[BruteForceRuntimeVC validatePin:]
validatePin: 1111
0x100166a30 iGoat!-[BruteForceRuntimeVC buttonClick:]
0x1877e0ad0 UIKit!-[UIApplication sendAction:to:from:forEvent:]
0x1877e0a4c UIKit!-[UIControl sendAction:to:forEvent:]
0x1877c8740 UIKit!-[UIControl _sendActionsForEvents:withEvent:]
0x1877e033c UIKit!-[UIControl touchesEnded:withEvent:]
0x187798b0c UIKit!_UIGestureRecognizerUpdate
0x18261c7b0 CoreFoundation!__CFRUNLOOP_IS_CALLING_OUT_TO_AN_OBSERVER_CALLBACK_FUNCTION__
0x18261a554 CoreFoundation!__CFRunLoopDoObservers
0x18261a984 CoreFoundation!__CFRunLoopRun
0x182544d10 CoreFoundation!CFRunLoopRunSpecific
0x183e2c088 GraphicsServices!GSEventRunModal
0x187811f70 UIKit!UIApplicationMain
0x10015ed74 iGoat!main
0x1820e28b8 libdyld.dylib!start
RET: nil
```

Frida: ObjC-method-observer

```
(0x1405232e0) -[BruteForceRuntimeVC validatePin:]  
validatePin: 1181  
0x100166a30 iGoat!-[BruteForceRuntimeVC buttonClick:]  
0x1877e0ad0 UIKit!-[UIApplication sendAction:to:from:forEvent:]  
0x1877e0a4c UIKit!-[UIControl sendAction:to:forEvent:]  
0x1877c8740 UIKit!-[UIControl _sendActionsForEvents:withEvent:]  
0x1877e033c UIKit!-[UIControl touchesEnded:withEvent:]  
0x1877dff6c UIKit!-[UIWindow _sendTouchesForEvent:]  
0x1877d8b08 UIKit!-[UIWindow sendEvent:]  
0x1877a8f4c UIKit!-[UIApplication sendEvent:]  
0x1877a7528 UIKit!_UIApplicationHandleEventQueue  
0x18261d124 CoreFoundation!__CFRUNLOOP_IS_CALLING_OUT_TO_A_SOURCE0_PERFORM_FUNCTION__  
0x18261cbb8 CoreFoundation!__CFRunLoopDoSources0  
0x18261a8b8 CoreFoundation!__CFRunLoopRun  
0x182544d10 CoreFoundation!CFRunLoopRunSpecific  
0x183e2c088 GraphicsServices!GSEventRunModal  
0x187811f70 UIKit!UIApplicationMain  
0x10015ed74 iGoat!main  
RET: 0x1
```

Lab6: Frida PIN bypass

OWASP M8
code
tampering



Goal: bypass a security check

Steps:

1. Open DVIA
2. Go to “**Runtime Manipulation**” / “Start Challenge”
3. Find the class and the method used to check the PIN code
4. Instrument DVIA using Frida to bypass the PIN code check

Lab6: Frida PIN bypass

OWASP M8
code
tampering



Steps (detailed):

1. Use **frida** and **ObjC-Method-Observer**:

`frida -FU -c mrmacet/objc-method-observer` to confirm class/method used for the PIN check

2. Use **frida-trace**:

`frida-trace -FU -m " - [RuntimeManipulationDetailsVC isLoginValidated]"` to add “backtrace” and confirm again the class/method

3. Use the JS handler created by **frida-trace** to modify the method result

Add `retval.replace(0);` to the **onLeave** function

Objection

```

root@kali:~# objection -g "iGoat" explore
Using USB device `iOS Device`
Agent injected and responds ok!

   _[ _|_ |_ _ _ _[ _|_ |_ _ _ 
  | . | . | | -| _| _| | . | | |
  |__|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_
  |__|_(object)inject(ion) v1.9.2

  Runtime Mobile Exploration
  by: @leonjza from @sensepost

[tab] for command suggestions
com.swaroop.iGoat on (iPhone: 12.1) [usb] # env

Name          Path
-----
BundlePath    /var/containers/Bundle/Application/ABDD92BD-D5C9-498E-B9E2-CE87CE854C82/iGoat.app
CachesDirectory /var/mobile/Containers/Data/Application/D56A459C-1C49-4B61-A0B4-4CC01537A5B2/Library/Caches
DocumentDirectory /var/mobile/Containers/Data/Application/D56A459C-1C49-4B61-A0B4-4CC01537A5B2/Documents
LibraryDirectory /var/mobile/Containers/Data/Application/D56A459C-1C49-4B61-A0B4-4CC01537A5B2/Library
com.swaroop.iGoat on (iPhone: 12.1) [usb] # ios
      import Import fridascript from a full path and run it
      ios Commands specific to iOS
      jobs Work with objection jobs

```

Lab7: Objection PIN bypass

OWASP M8
code
tampering



Goal: bypass a security check

Steps:

1. Open DVIA
2. Go to “**Runtime Manipulation**” / “Start Challenge”
3. Instrument DVIA using Objection to bypass the PIN code check

Lab7: Objection PIN bypass

OWASP M8
code
tampering



Steps (detailed):

Use **objection** and:

ios hooking list class_methods to list class/method used for the PIN check

ios hooking watch method to add “backtrace” and confirm again the class/method

ios hooking set return_value to modify the method result

Transport security

Lab8: network capture with rvictl

Goal: analyze network communications

Steps:

1. Connect a device using USB and click on “Trust”

2. Get device UDID (serial number)

3. Create redirection:

```
rvictl -s 81e9e294d31bf751b778c7abb6bf83ee6dec8fd
```

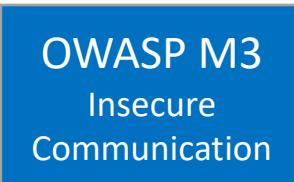
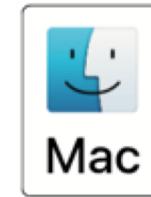
4. Capture network traffic with tcpdump or wireshark

```
tcpdump -Xni rvi0
```

5. Launch DVIA, go to “**Transport Layer Security**” / “Send over HTTP”

6. Delete redirection:

```
rvictl -x 81e9e294d31bf751b778c7abb6bf83ee6dec8fd
```



Only for macOS - sorry

Lab8: network capture with rvi_capture

Goal: analyze network communications

For Linux :)

OWASP M3
Insecure
Communication



Steps:

1. Connect a device using USB and click on “Trust”
2. Get device UDID (serial number)
(using the lsusb command in *ios_cheatsheet.txt*)
3. Launch capture:

```
./rvi_capture.py --udid <UDID> iPhone.pcap
```

4. Launch DVIA, go to “Transport Layer Security” / “Send over HTTP”
5. Stop capture and analyze network traffic with tcpdump or wireshark

```
tcpdump -Xnr iPhone.pcap
```

Questions

