App Communication Testing

News

- https://blog.pradeo.com/accessibility-services-mobile-analysis-malware

Objectives

- Implement testing tools
- Test app functions

What to Investigate

- Protocols being used – is it encrypted?
- Servers being contacted – Are they trusted?
  - Identify ones from MobSF report
- Data being sent – Is it appropriate for app?
- Exposed passwords, tokens, cookies, API keys.

Key Challenges

- Decrypting data
  - Wi-Fi encryption – use the key and Wireshark
  - https/tls encryption – proxy server
- Monitoring traffic over cellular interfaces
- Monitoring SMS/MSM

App Packet Capture

- For iOS – rvictl is the solution
- Android is a bit more complicated
  - Tools like Burp Suite, Fiddler, Bettercap can provide proxy and MITM capabilities

IOS Packet Capture

- Capture packets through a Mac using Xcode rvictl command line utility
- Connect iPhone to Mac through USB
- Create and start a virtual interface with rvictl
- Use tcpdump to capture packets and save pcap file
- Use Wireshark to analyze
- Captures wifi and LTE
- Details: https://www.agnosticdev.com/blog-entry/networking-ios/capturing-packet-trace-ios-device
- Demo: https://www.youtube.com/watch?v=vE8Xu97SSQg

Android Communication Type

- Web browser traffic
  - Some apps use a WebView as their method of communication with server
  - Can be captured with Chrome Inspect
- HTTP
  - Easy to capture with a proxy
- HTTPS
  - Can capture with a proxy if keys are accepted
  - May require modification of app to accept user certificates
- Other protocols
  - Requires Burp add-on or Bettercap

Android WebView Communication

- Many apps use WebView for multiplatform convenience and standard UI
- Examples:
  - Gmail
  - Uber
  - Burger King
- WebViews indicate HTTP/HTTPS Traffic
- WebViews can be vulnerable to web attacks like js injection or XSS

Android WebView Communication

- For Android, see if your app uses WebView and view cookie flags
  - Activate remote debugging on device and connect to computer with USB or activate in emulator
    - Activate developer mode **settings - about phone (or emulated device) – build number *tap seven times***
    - If real device - **Settings – System – Advanced – Developer Options – USB Debugging (also Wireless debugging)**
  - On host computer - start Chrome and browse to **`chrome://inspect/#devices`**
  - Start your app and communicate with server
    - Can test with Webview Test app in play store
  - If your app is using WebView it will show a section for your app
  - https://developer.chrome.com/docs/devtools/remote-debugging/webviews/

WebView Demo

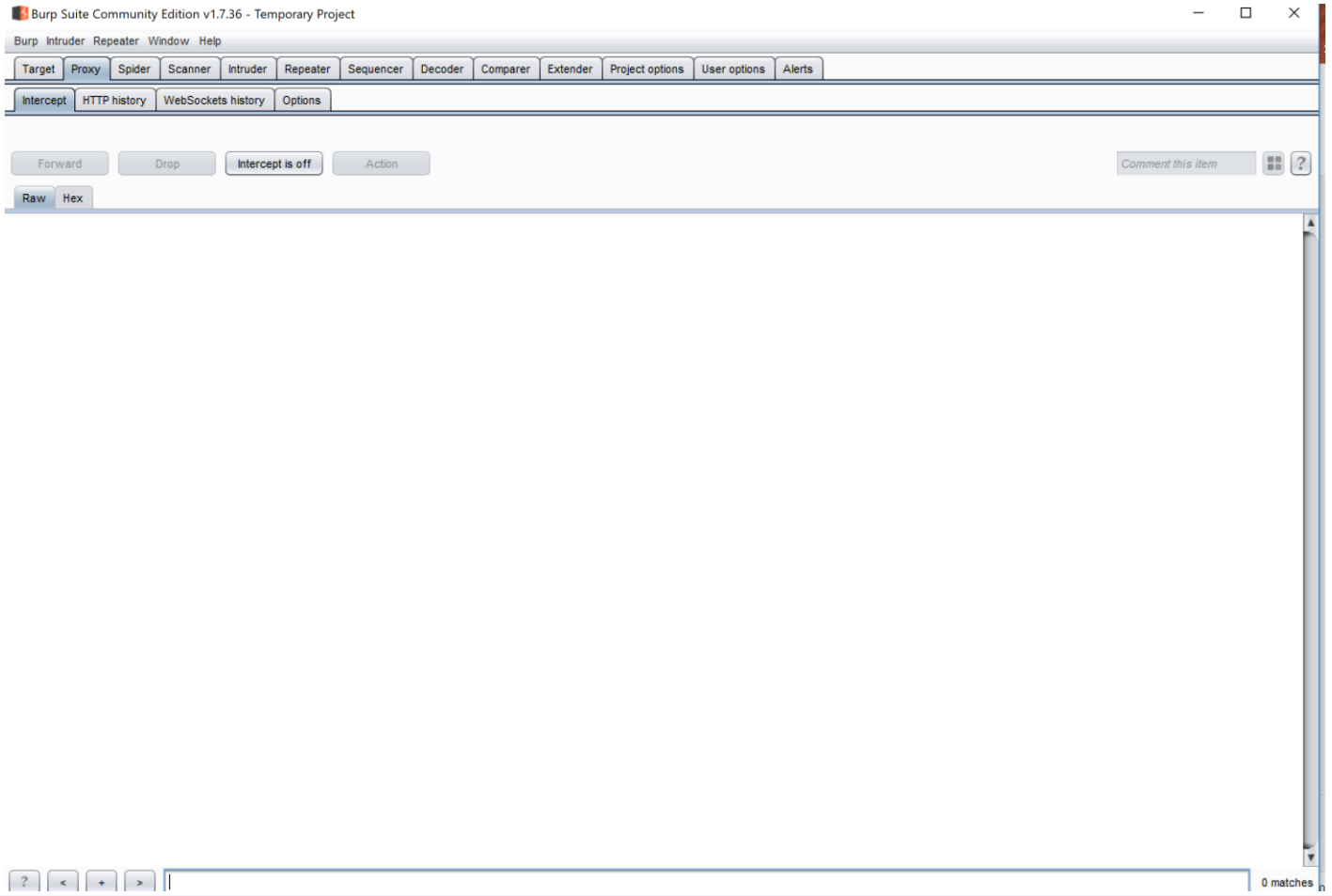
- Download WebviewTest.apk from Canvas and install on emulator
- Start Chrome browser find WebViews and debug
- Note casting tab in debugger
- Since traffic is viewed from device, it will be decrypted

Capture Mobile Traffic With a Proxy

- Use Burp Suite
  - https://support.portswigger.net
- Only proxies HTTP and HTTPS traffic, will not see other protocols
  - Non-HTTP Protocol Extension Proxy ***(NoPE Proxy)***
- Browser should accept Burp certificate if it is installed on the device
  - Other apps may not

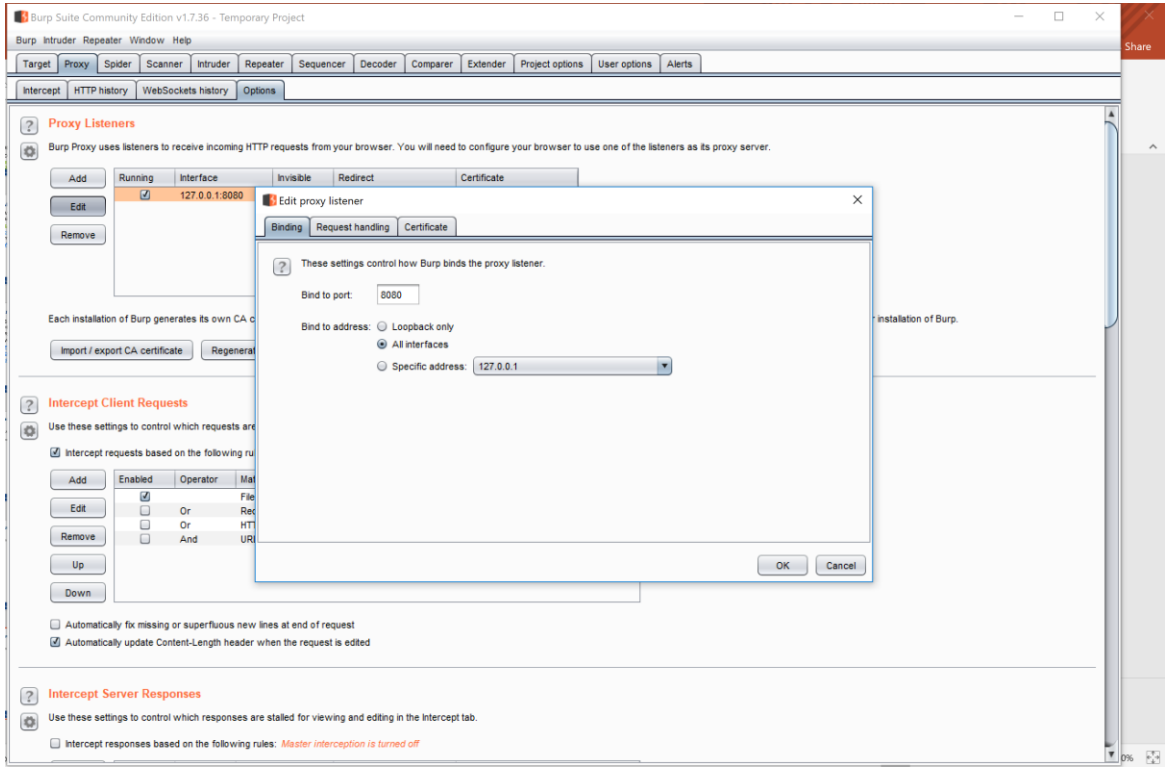
Install and start Burp Suite

- In Proxy-Intercept tab, turn Intercept off



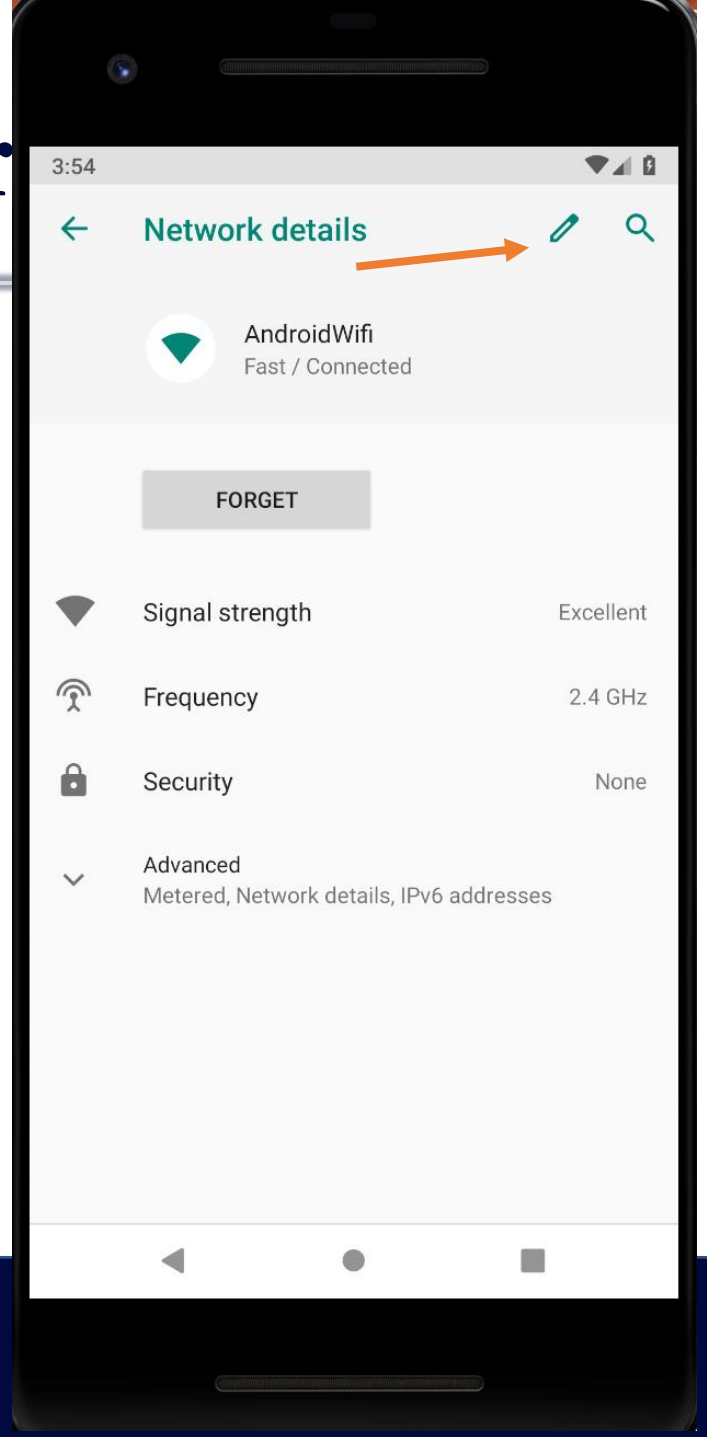
Burp Suite Proxy Config

- In Proxy – Options tab select loopback interface and select edit
  - Select all interfaces
  - On Certificates tab make sure selection is to generate certificates per host
  - In dashboard select Running



Set Proxy Server on Emulator

- Go to Network and Internet Settings
- Click on Wi-Fi network, then on gear for settings
- Click on pencil to edit network
- Advanced Options – Proxy – Manual
  - Enter IP address of your computer's wifi interface and Burp port number
  - Save



Ensure Traffic is Proxied

- Open Chrome Browser and go to http://neversll.com
  - Site should open
  - Go to Burp HTTP tab and see traffic to neverssl
- In Chrome go to Yahoo.com
  - Should fail – note traffic in Burp
  - Select advanced and continue – note traffic in Burp shows TLS

Using Proxy with HTTPS

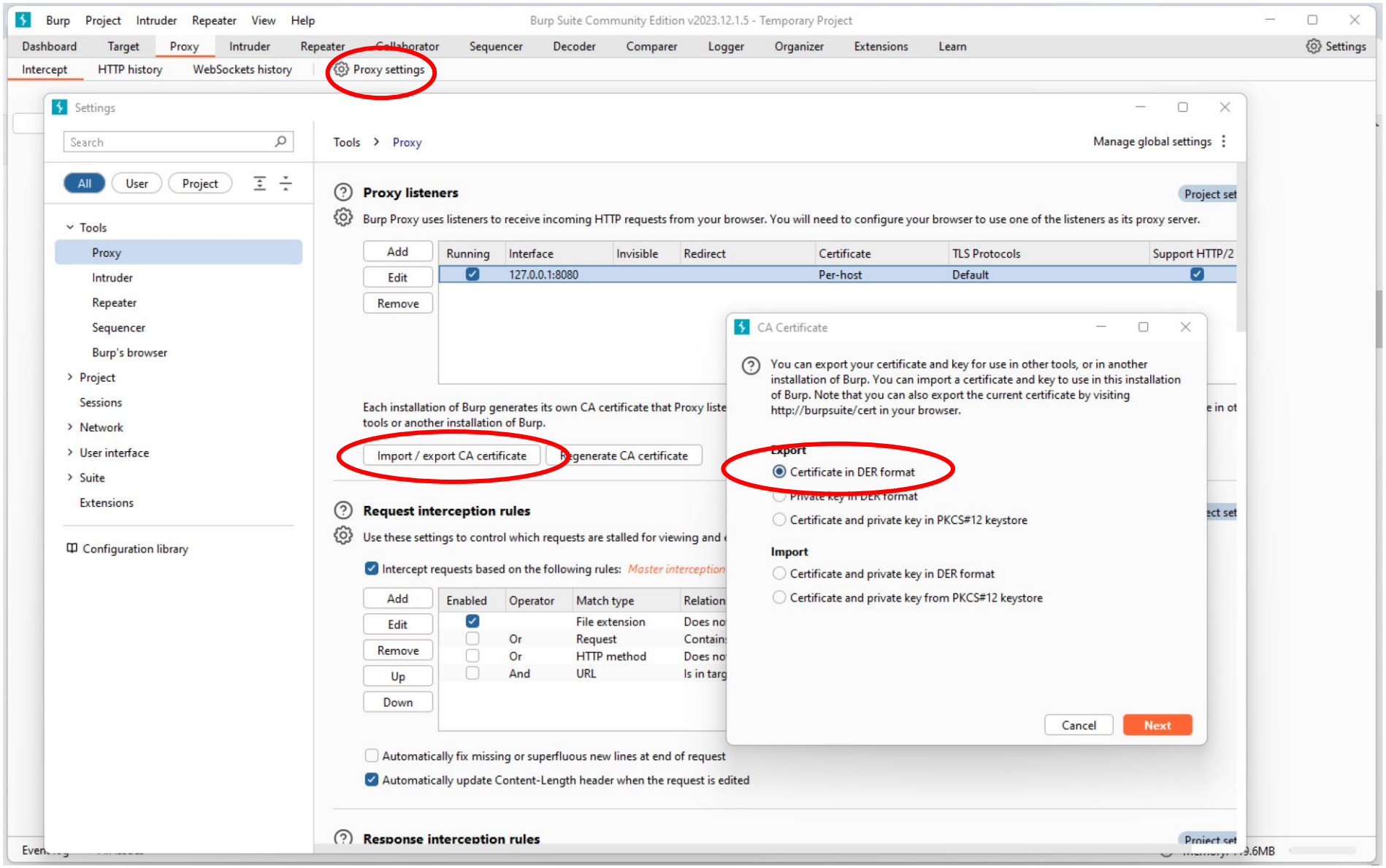
Generate and Install Burp Certificate

- You can only download the Burp certificate if you are going through the proxy, so go to the Proxy tab and click the button to Open Browser to get the certificate in detailed instructions at:
  - https://portswigger.net/support/installing-burp-suites-ca-certificate-in-an-android-device
- For iOS apps, install Burp Mobile Assistant
  - https://portswigger.net/burp/documentation/desktop/mobile/config-ios-device
- For Android 7 and up – apps don't trust user CA certificates and may have to be modified as described later.

Generate Burp Certificate in DER format

Save as cacert.der

Rename as cacert.crt



Add Certificate to Emulator

- Drag cacert.crt and drop on emulator
- Open settings
- Go to 'Security' – 'Advanced'
- Go to 'Encryption & Credentials'
- Go to 'Install Certificate'
- Select 'CA Certificate' from the list of types available
- Accept a large scary warning
- Browse to the certificate file on the device and open it

Confirm Certificate Installation

- As of this writing, the Chrome browser will accept the certificate
- Confirm the certificate is installed and working
  - In Emulator open Chrome, go to http://neverssl.com to confirm working connection
  - In Chrome go to any encrypted site like Yahoo.com and confirm it opens

Using Burp Certificate With Apps

- Check MobSF Network Security Section
  - If system certificates are trusted, modify app to trust user certificates

NO	SCOPE	SEVERITY	DESCRIPTION
1	*	secure	Base config is configured to disallow clear text traffic to all domains.
2	*	warning	Base config is configured to trust system certificates.

Modify App to Trust User Certificates

- Get apk from device or emulator:
  - Adb shell pm list packages
  - Adb shell pm path com.appname
  - Copy long path name to adb pull command
  - Rename base.apk to long apk name
- Use apktool to decompile
  - https://apktool.org/docs/install/
  - Can be installed on Linux with sudo apt install apktool
- Edit res/xml/network_security_config.xml and add:
  - <certificates src="user"/> under trust-anchors
  - https://developer.android.com/training/articles/security-config

Creating network_security_config.xml

If network_security_config.xml doesn't exist, create it

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <domain-config>
        <domain includeSubdomains="true">example.com</domain>
        <trust-anchors>
            <certificates src="user"/>
        </trust-anchors>
    </domain-config>
</network-security-config>
```

Add network_security_config to manifest

- <manifest ... >
- <application android:networkSecurityConfig="@xml/network_security_config"
- ... >
- ...
- </application>
- </manifest>

Resign and Reinstall app

- Use apktool to rebuild
  - if error, check to see if message calls out your xml file and check syntax
- Navigate to location of apksigner and create a key
  - c:\Users***username***\AppData\Local\Android\Sdk\build-tools***version#***
  - keytool -genkeypair -v -keystore <filename>.keystore -alias <key-name> -keyalg RSA \ -keysize 2048 -validity 10000
- Insure apk is formatted correctly:
  - zipalign -f -v 4 mono.samples.helloworld-unsigned.apk helloworld.apk
- Sign the app
  - jarsigner -keystore mykey.keystore app_name.apk
- Adb install

App traffic still may not show!?!

- App may be using SSL Pinning
  - Check MobSF Network Security Section

NO	SCOPE	SEVERITY	DESCRIPTION
1	sha256.badssl.com	info	Domain config is configured to trust bundled certs @raw/lets_encrypt_isrg_root.
2	sha256.badssl.com	secure	Certificate pinning does not have an expiry. Ensure that pins are updated before certificate expire. [Pin: NgJeUutmfGsIONh0XaovCA5VJ05uv2gCb27pUOpTPxU= Digest: SHA-256]

https://cheatsheetseries.owasp.org/cheatsheets/Pinning_Cheat_Sheet.html

Bypassing SSL Pinning

- The OWASP Mobile Application Security Testing guide provides options
- Tutorial: https://www.hackingarticles.in/android-hooking-and-sslpinning-using-objection-framework/
- Not required for this class, but highly encouraged if you have the tools

App traffic still may not show!?!

- App may be using other protocols besides HTTP
  - Try Non-HTTP Protocol Extension Proxy ***(NoPE Proxy)***
  - Other methods are beyond the scope of class
- Custom app platform like Xamarin may be unaware of system proxy
- App may be intentionally blocking proxy
  - Testing of both of these is beyond our scope but could be done looking at source code.

Documenting Results in MASVS

Include screenshots to prove your findings/issues

MASVS-PLATFORM-2

Control

- The app uses WebViews securely.

Description

- WebViews are typically used by apps that have a need for increased control over the UI. This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code).

Testing

- Note presence or absence of WebViews for later detailed testing

MASVS-NETWORK-1

Control

- The app secures all network traffic according to the current best practices.

Description

- Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries. This control ensures that the app is in fact setting up secure connections in any situation.

MASVS-NETWORK-1 Tests

- MASTG-TEST-0019 Testing Data Encryption on the Network
  - View ALL app traffic in Burp and verify that it is encrypted.
- MASTG-TEST-0020 Testing the TLS Settings
  - View traffic and verify TLS version > 1.1
- MASTG-TEST-0021 Testing Endpoint Identify Verification
  - Check MANIFEST.XML for network-security-config section with custom trust anchors
  - Attempt to run app with Burp certificate
- MASTG-TEST-0023 Testing the Security Provider
  - Beyond our scope, but you can look at source code in Module 9

MASVS-NETWORK-2

Control

The app performs identity pinning for all remote endpoints under the developer's control.

Description

Instead of trusting all the default root CAs of the framework or device, this control will make sure that only very specific CAs are trusted. This practice is typically called certificate pinning or public key pinning.

MASVS-NETWORK-2 Testing

- MASTG-TEST-0022 Testing Custom Certificate Stores and Certificate Pinning
  - Check MobSF report
  - Do extra credit app modification to see if traffic still fails

MASVS-AUTH-1

Control

The app uses secure authentication and authorization protocols and follows the relevant best practices.

Description

Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols.

Testing

Note if authentication process is encrypted

Provide Evidence!

- Capture screen shots of each finding
  - Show things that are there AND not there
- Name shots well and organize in folders by MASVS
- Paste into your MASVS journal you are building

Summary

- Packet capture on Android and iOS
- Objectives, tools and processes for testing mobile app communication
- Documenting findings in MASVS