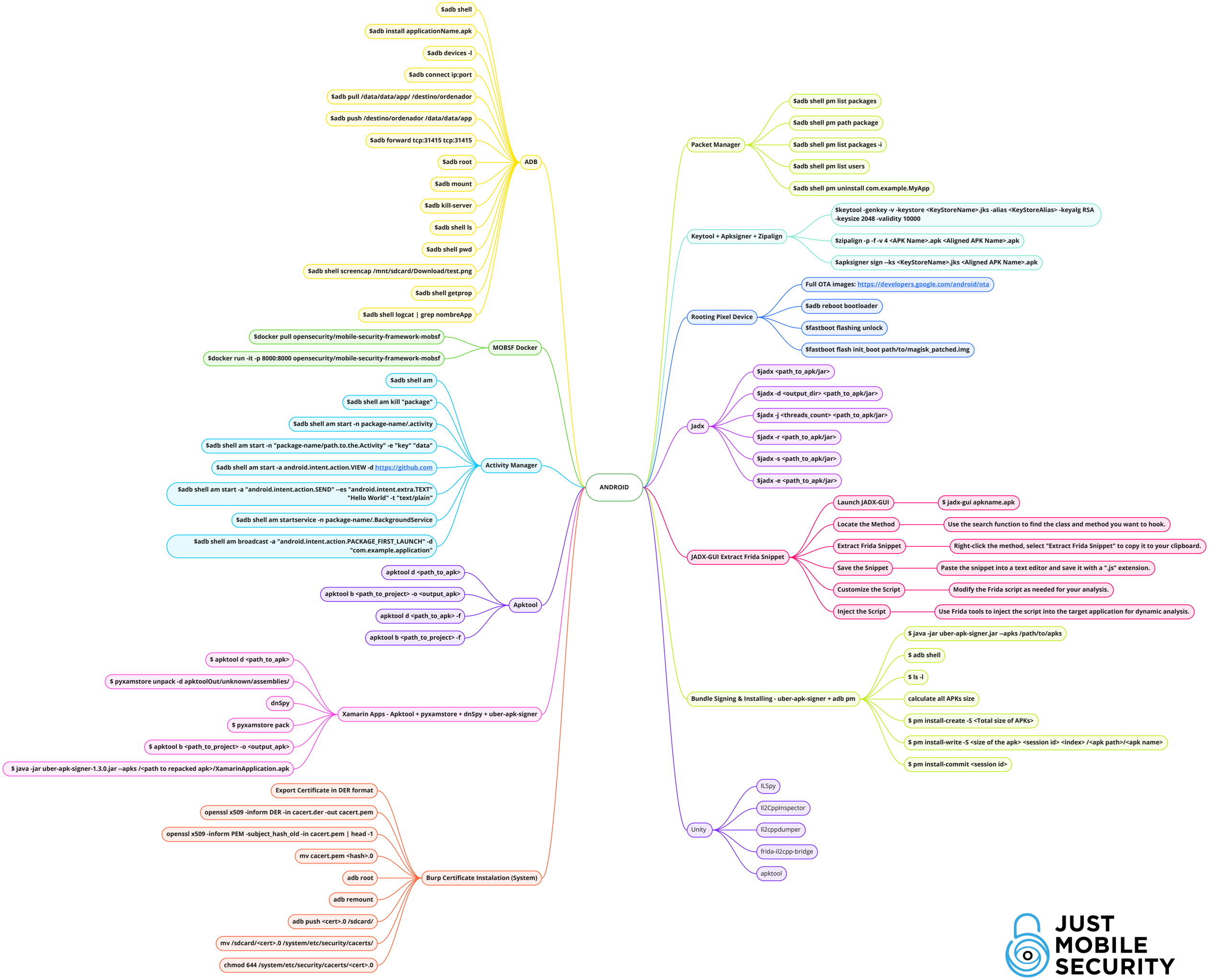
Android Cheatsheet MindMap



JUST MOBILE SECURITY

ADB

$adb shell
Opens a remote shell on the device/emulator.
$adb install applicationName.apk
Installs an Android application (.apk) on the device/emulator.
$adb devices -l
Lists the connected devices/emulators with detailed information.
$adb connect ip:port
Connects to a device/emulator over TCP/IP using the specified IP address and port.

$adb root
Restarts the adbd daemon with root permissions on the device.
$adb mount
Mounts the device's filesystem in read-write mode.
$adb kill-server
Kills the ADB server daemon.
$adb shell ls
Lists files and directories on the device/emulator.

$adb forward tcp:31415 tcp:31415
Forwards TCP traffic from a specified local port to a specified device/emulator port.
$adb pull /data/data/app/ /destino/ordenador
Pulls files from the device/emulator to the computer.
$adb push /destino/ordenador /data/data/app
Pushes files from the computer to the device/emulator.

$adb shell pwd
Prints the current working directory on the device/emulator.
$adb shell screencap /mnt/sdcard/Download/test.png
Takes a screenshot of the device/emulator screen and saves it to the specified location.
$adb shell getprop
Retrieves system properties from the device/emulator.
$adb shell logcat | grep nombreApp
Filters and displays logcat output for a specific app.

Packet Manager

$adb shell pm list packages
Lists all installed packages on the device/emulator.
$adb shell pm path package
Prints the path of the APK file associated with a package.
$adb shell pm list packages -i
Lists installed packages along with their installer package names.
$adb shell pm list users
Lists all users on the device/emulator.
$adb shell pm uninstall com.example.MyApp
Uninstalls the specified package.

Keytool + Apksigner + Zipalign

$keytool -genkey -v -keystore <KeyStoreName>.jks -alias <KeyStoreAlias> -keyalg RSA -keysize 2048 -validity 10000
Generates a new keystore and private key pair.
$zipalign -p -f -v 4 <APK Name>.apk <Aligned APK Name>.apk
Aligns and optimizes an Android APK file.
apksigner sign --ks <KeyStoreName>.jks <Aligned APK Name>.apk
Signs an aligned APK file using the provided keystore.

MOBSF Docker

$docker pull opensecurity/mobile-security-framework-mobsf
Pulls the Docker image of Mobile Security Framework (MobSF) from the Open Security repository.

$docker run -it -p 8000:8000 opensecurity/mobile-security-framework-mobsf
Runs the MobSF Docker container in interactive mode, mapping port 8000 of the host to port 8000 of the container.

Rooting Pixel Device

Full OTA images
https://developers.google.com/android/ota

$adb reboot bootloader
Reboots the connected Android device/emulator into bootloader mode.
$fastboot flashing unlock
Unlocks the bootloader of the device, allowing the installation of custom firmware.
$fastboot flash init_boot path/to/magisk_patched.img
Flashes the Magisk-patched boot image (magisk_patched.img) onto the device.

JUST MOBILE SECURITY

Activity Manager

$adb shell am
Sends an Activity Manager (am) shell command.
$adb shell am kill "package"
Kills the specified package's process.
$adb shell am start -n package-name/.activity
Starts the specified activity of a package.
$adb shell am start -n "package-name/path.to.the.Activity" -e "key" "data"
Starts an activity with extra data specified by key-value pairs.

$adb shell am start -a android.intent.action.VIEW -d https://github.com
Opens a URL in the default browser.
$adb shell am start -a "android.intent.action.SEND" --es "android.intent.extra.TEXT" "Hello World" -t "text/plain"
Initiates a send action with specified text and MIME type.
$adb shell am startservice -n package-name/.BackgroundService
Starts a background service in a specified package.
$adb shell am broadcast -a "android.intent.action.PACKAGE_FIRST_LAUNCH" -d "packagename"
Sends a broadcast intent to the system.

Jadx

Basic Usage:
$jadx <path_to_apk/jar>
Decompiles the specified APK or JAR file and displays the decompiled code.

Advanced Options:
$jadx -d <output_dir> <path_to_apk/jar>
Decompiles the file and saves the output to the specified directory.
$jadx -j <threads_count> <path_to_apk/jar>
Decompiles the file using the specified number of processing threads.
$jadx -r <path_to_apk/jar>
Decompiles the file without extracting resources (disables resources decompilation).
$jadx -s <path_to_apk/jar>
Decompiles the file without generating Java source code (disables source code generation).
$jadx -e <path_to_apk/jar>
Exports the decompiled project as a Gradle project.

Apktool

Basic Usage:
$apktool d <path_to_apk>
Decompiles the specified APK file and extracts its resources and source code.
$apktool b <path_to_project> -o <output_apk>
Rebuilds an APK from a decompiled project located at the specified path and saves it to the specified output file.
Advanced Options:
$apktool d <path_to_apk> -f
Forces overwriting of the output directory if it already exists.
$apktool b <path_to_project> -f
Forces rebuilding of the APK, even if the output file already exists."

Unity

ILSpy
An open-source .NET assembly browser and decompiler for analyzing and debugging .NET code.
Il2CppInspector
A tool to generate C++ headers and reconstruct Unity IL2CPP binaries for reverse engineering.
Il2cppdumper
A Unity IL2CPP binary dumper to extract metadata and reconstruct symbols for analysis.
frida-il2cpp-bridge
A Frida-based toolkit for inspecting and manipulating Unity IL2CPP applications at runtime.

JADX-GUI Extract Frida Snippet

Launch JADX-GUI
Open the APK file using the command **$ jadx-gui apkname.apk**

Locate the Method
Use the search function to find the class and method you want to hook.

Extract Frida Snippet
Right-click the method, select "Extract Frida Snippet" to copy it to your clipboard.

Save the Snippet
Paste the snippet into a text editor and save it with a ".js" extension.

Customize the Script
Modify the Frida script as needed for your analysis.

Inject the Script
Use Frida tools to inject the script into the target application for dynamic analysis.

Xamarin Apps - Apktool + pyxamstore + dnSpy + uber-apk-signer

$ apktool d <path_to_apk>
Decompiles an APK file into a readable and modifiable format for analysis and modification.
$ pyxamstore unpack -d apktoolOut/unknown/assemblies/
Unpacks the assemblies (DLLs) from the decompiled APK using pyxamstore, a tool for working with Xamarin applications.
dnSpy
Launches dnSpy, .NET decompiler and debugger, to analyze the unpacked assemblies for Xamarin applications.

$ pyxamstore pack
Packs the modified assemblies back into a Xamarin application format using pyxamstore.
$ apktool b <path_to_project> -o <output_apk>
Rebuilds the modified project into a new APK file using apktool.
$ java -jar uber-apk-signer-1.3.0.jar --apks /<path to repacked apk>/XamarinApplication.apk
Signs the repacked APK file using uber-apk-signer, a tool for signing Android APKs, ensuring the integrity and authenticity of the APK.

Bundle Signing & Installing - uber-apk-signer + adb pm

$ java -jar uber-apk-signer.jar --apks /path/to/apks
Signs multiple APK files located at the specified path using the Uber APK Signer tool.
$ adb shell
Opens a shell session on the connected Android device/emulator.
$ ls -l
Lists the files and directories in the current directory on the Android device.
calculate all APKs size
Calculates the total size of multiple APK files that are part of an app bundle.

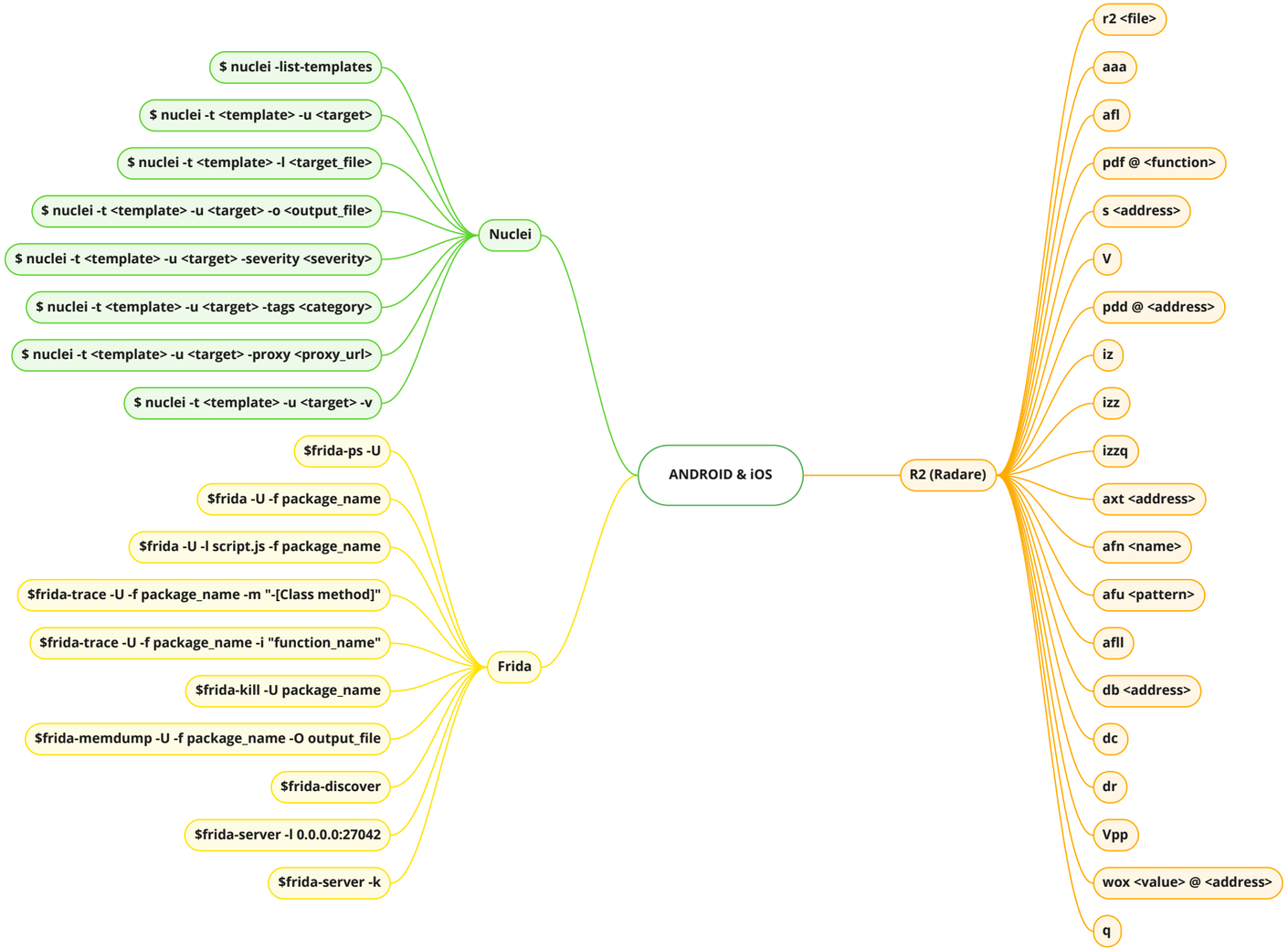
$ pm install-create -S <Total size of APKs>
Initiates the installation session for installing multiple APKs with a specific total size.
$ pm install-write -S <size of the apk> <session id> <index> /<apk path>/<apk name>
Writes individual APK files from an app bundle to the installation session with the specified size, session ID, index, and path. Run the command for each APK.
$ pm install-commit <session id>
Commits the installation session with the specified session ID, installing the APK files on the Android device.

Burp Certificate Instalation (System)

Export Certificate in DER format
Exports the certificate from Burp in DER format.
$openssl x509 -inform DER -in cacert.der -out cacert.pem
Converts the exported certificate (cacert.der) from DER format to PEM format.
$openssl x509 -inform PEM -subject_hash_old -in cacert.pem | head -1
Calculates the subject hash of the PEM-formatted certificate.
$mv cacert.pem <hash>.0
Renames the converted PEM certificate file (cacert.pem) to <hash>.0, where <hash> represents the calculated subject hash.
$adb root
Restarts the adbd daemon with root privileges on the connected Android device/emulator.

$adb remount
Remounts the device's /system partition in read-write mode, allowing modifications.
$adb push <cert>.0 /sdcard/
Copies the renamed certificate file (<hash>.0) to the device's internal storage (/sdcard/).
$mv /sdcard/<cert>.0 /system/etc/security/cacerts/
Moves the certificate file from the internal storage to the /system/etc/security/cacerts/ directory.
$chmod 644 /system/etc/security/cacerts/<cert>.0
Sets the appropriate file permissions (644) for the certificate file in the /system/etc/security/cacerts/ directory.

Android & iOS Cheatsheet MindMap



JUST MOBILE SECURITY

Android & iOS Cheatsheet

Frida

$frida-ps -U
Lists all running processes on the connected Android device. (Displays process names and their corresponding process identifiers (PIDs).)

$frida -U -f package_name
Attaches to a running process for dynamic analysis.

$frida -U -l script.js -f package_name
Injects and runs a Frida script into the specified package for dynamic analysis.

$frida-trace -U -f package_name -m "-[Class method]"
Traces a specific method of a class for detailed analysis.

$frida-trace -U -f package_name -i "function_name"
Traces a specific function for detailed analysis.

$frida-kill -U package_name
Terminates the specified package forcefully.

$frida-memdump -U -f package_name -O output_file
Dumps the memory of a specific process to a file.

$frida-discover
Discovers and lists nearby Frida server devices.

$frida-server -l 0.0.0.0:27042
Starts the Frida server on a specific host and port for remote device communication.

Nuclei

$ nuclei -list-templates
Lists all available Nuclei templates.
$ nuclei -t <template> -u <target>
Runs a specific template against a target URL.
$ nuclei -t <template> -l <target_file>
Runs a specific template against a list of targets from a file.
$ nuclei -t <template> -u <target> -o <output_file>
Saves the results of a scan to a specified output file.

$ nuclei -t <template> -u <target> -severity <severity>
Filters templates based on the specified severity level.
$ nuclei -t <template> -u <target> -tags <category>
Filters templates based on the specified category.
$ nuclei -t <template> -u <target> -proxy <proxy_url>
Sets the proxy URL for making requests.
$ nuclei -t <template> -u <target> -v
Enables verbose output for detailed scanning information.

R2 (Radare)

r2 <file>
Opens the specified file in Radare2 for analysis.
aaa
Analyzes the binary, performing several automated analysis tasks, such as function detection, basic block identification, and more.
afl
Lists all functions in the binary.
pdf @ <function>
Disassembles the specified function and displays it in the default output format.
s <address>
Seeks to the specified address in the binary.

axt <address>
Cross-references the specified address, showing all references to it.
afn <name>
Searches for a function with the specified name.
afu <pattern>
Searches for functions matching the specified pattern.
afll
Lists all local variables for the current function.
db <address>
Sets a breakpoint at the specified address.

V
Enters the visual mode, providing an interactive interface for exploring and analyzing the binary.
pdd @ <address>
Disassembles the data at the specified address.
iz
Lists all strings found in the binary.
izz
Lists all function names and strings found in the binary.
izzq
Lists all unique function names found in the binary.

dc
Continues the execution after a breakpoint or stops.
dr
Shows all registers and their values.
Vpp
Opens the pseudo-graph view to visualize the control flow graph.
wox <value> @ <address>
Overwrites the value at the specified address with the specified value.
q
Quits Radare2.

iOS Cheatsheet MindMap

- iOS
  - Jailbreak
    - Download palera1n-macos-universal from https://palera.in/ into Downloads folder
    - Enable Full Disk Access for Terminal
    - $ cd ~/Downloads
    - $ sudo mkdir -p /usr/local/bin
    - $ sudo mv ./palera1n-macos-universal /usr/local/bin/palera1n
    - $ sudo xattr -c /usr/local/bin/palera1n
    - $ sudo chmod +x /usr/local/bin/palera1n
    - $ palera1n
    - Follow Palera1n instructions
  - Getting the binary & binary information
    - Install the application from the AppStore.
    - Connect your jailbroken device to your computer.
    - $ iproxy 2222 44
    - $ frida-ps -Uai | grep <app name>
    - $ python3 dump.py <app name>
    - $ frida -U --codeshare dki/ios-app-info -f <packageName>
    - Once Inside Frida -> **dki/ios-app-info.** Execute **appInfo()**
  - Ipa Installation
    - Node-Applesign (MacOS)
      - $ npm install -g applesign
      - Open Xcode and create a new proyect to generate a Mobile Provisioning Profile
      - $ applesign -w -c -m embedded.mobileprovision target.ipa
    - ideviceinstaller
      - $ ideviceinstaller install <ipaFile>
  - SSL Pinning
    - SSL Killswitch 2
      - Download SSLLillSwitch deb file from https://github.com/nabla-c0d3/ssl-kill-switch2/releases
      - $ ssh root@iphone_ip (Password alpine)
      - $ dpkg -i com.nablac0d3.sslkillswitch2_0.14.deb
      - Go to setting & look for SSL kill switch application.
      - Click on Disable SSL Certificate
    - SSL Pinning Bypass via Objection
      - Install Frida Server
        - From iPhone open Cydia => Source => Edit => Add => https://build.frida.re => search =>frida =>Install
      - $ sudo pip3 install objection
      - $ objection — gadget package_name explore
      - $ ios sslpinning disable
    - SSL Pinning bypass via Frida
      - $ frida -U --codeshare snooze6/ios-pinning-disable -f package_name
      - $ frida -U -f package_name -l your_custom_ssl_pinning_bypass.js
  - Installing Frida
    - Install Frida Server in Palera1n:
      From iPhone open Sileo => Source => Edit => Add => https://build.frida.re => search =>frida =>Install
  - USB SSH TUNNEL
    - Linux: $ sudo apt install libusbmuxd-tools
    - MacOS: $ brew install libimobiledevice
    - $ iproxy 2222 44
    - $ ssh -p 2222 root@iphone_ip (Password alpine)
    - sudo ./itnl 2222 22 (A little bit old)
  - File Transfer
    - Filza
      - Open Sileo and search for Filza File Manager
      - Click on GET button
      - Open Filza App
    - SCP
      - $ scp /path_to_file_to_upload root@your_iphon_ip:/path_to_destination_folder
      - $ scp root@your_iphon_ip:/path_to_file /path_to_destination_folder
  - App Provisioning Profile
    - Open Xcode and create a sample project
    - In Certificates, Identifiers & Profiles, click Profiles in the sidebar, then click the add button (+) on the top left.
    - Select an App Store distribution profile for your platform, then click Continue.
    - Choose the App ID you used for development and click Continue.
    - If you use automatic signing during development, choose one of the Xcode given options
    - Select your distribution certificate, then click Continue.
    - Enter a profile name, then click Generate.
    - Click Download.
  - libimobiledevice
    - Installation
      - $ sudo apt-get install build-essential pkg-config checkinstall git autoconf automake libtool-bin libplist-dev libimobiledevice-dev libzip-dev usbmuxd
      - $ git clone https://github.com/libimobiledevice/ideviceinstaller.git
      - $ cd ideviceinstaller
      - $ ./autogen.sh
      - $ make
      - $ sudo make install
    - Useful Commands
      - $ ideviceinfo -l
      - $ idevicesyslog -m <appname> (--match STRING)
      - $ idevicebackup2 list
      - $ ideviceprovision list
      - $ ideviceinstaller install <ipaFile>
      - $ ideviceenterrecovery
      - $ iproxy 2222 44
  - Additional Jailbreaks
    - unc0ver (iOS 11.0 up to 14.8 (excluding iOS 12.5.6/12.5.7)
    - Dopamine (iOS 15.0 to 16.5)
    - Taurine (iOS 14.0 to 14.8.1)
    - Chimera (iOS 12.0 to 12.1.2)
    - Pangu933 (iOS 9.2 up to 9.3.3)

JUST MOBILE SECURITY

Jailbreak

IMPORTANT! never setup the passcode!, if the phone had ever setted up passcode reset it from factory.
1. **Download palera1n-macos-universal** from https://palera.in/ into Downloads folder
2. **Enable Full Disk Access for Terminal** (this only has to be done once) a. macOS Ventura and above: System Settings → Privacy & Security → Full Disk Access b. If Terminal does not show up in the list, click the plus icon and select it from Applications → Utilities. (this only has to be done once):

3. **$ cd ~/Downloads**
4. **$ sudo mkdir -p /usr/local/bin**
5. **$ sudo mv ./palera1n-macos-universal /usr/local/bin/palera1n**
6. **$ sudo xattr -c /usr/local/bin/palera1n**
7. **$ sudo chmod +x /usr/local/bin/palera1n**
8. **$ palera1n**
9. Follow Palera1n instructions

Additional Jailbreaks

unc0ver (iOS 11.0 up to 14.8 (excluding iOS 12.5.6/12.5.7)
Dopamine (iOS 15.0 to 16.5)
Taurine (iOS 14.0 to 14.8.1)
Chimera (iOS 12.0 to 12.1.2)
Pangu933 (iOS 9.2 up to 9.3.3)

Installing Frida

Install Frida Server in Palera1n:
From iPhone open Sileo => Source => Edit => Add => https://build.frida.re => search =>frida =>Install

USB SSH TUNNEL

Installing iproxy:
Linux: **$ sudo apt install libusbmuxd-tools**
MacOS: **$ brew install libimobiledevice**

Connecting via SSH:
1. **$ iproxy 2222 44**
Starting iproxy binding port 44 (Palera1n default SSH port) to 2222
2. **$ ssh -p 2222 root@iphone_ip** (Password alpine)
Connecting via ssh to device

Getting the binary & binary information

1. Install the application from the AppStore.
2. Connect your jailbroken device to your computer.
3. **$ iproxy 2222 44**
Run iProxy from terminal
4. **$ frida-ps -Uai | grep <app name>**
Obtain app Package name
5. **$ python3 dump.py <app name>**
Pull a decrypted IPA from a jailbroken device using frida-ios-dump
6. **$ frida -U --codeshare dki/ios-app-info -f <packageName>**
Get additional information
7. Once Inside Frida -> **dki/ios-app-info.** Execute **appInfo()**

File Transfer

Installing Filza: (also useful to install .ipa files)
1. Open **Sileo** and a new source "http://apt.thebigboss.org/"
2. Search for **Filza File Manager**
3. Click on **GET** button
4. **Open Filza App**

Using scp:
1. **$ scp /file_path_to_upload root@your_iphon_ip:/path_to_destination_folder**
Push file to device
2. **$ scp root@your_iphon_ip:/path_to_file /path_to_destination_folder**
Pull file from device

Ipa Installation

ideviceinstaller:
$ ideviceinstaller install <ipaFile>

Node-Applesign (MacOS):
1. **$ npm install -g applesign**
2. Open Xcode and create a new proyect to **generate a Mobile Provisioning Profile**
3. **$ applesign -w -c -m embedded.mobileprovision target.ipa**

SSL Pinning

SSL Killswitch 2:
1. On the device **download SSLKillSwitch deb file** from https://github.com/nabla-c0d3/ssl-kill-switch2/releases
2. **$ ssh root@iphone_ip** (Password alpine)
Connect via ssh to device.
3. **$ dpkg -i com.nablac0d3.sslkillswitch2_0.14.deb**
Installing the Killswitch 2 package.
4. Go to **setting** & look for **SSL kill switch** application.
5. **Click on Disable SSL Certificate** and SSL pinning of all the applications will be bypassed.

SSL Pinning Bypass via Objection:
1. **Install Frida Server**: From iPhone open Cydia => Source => Edit => Add => https://build.frida.re => search =>frida =>Install
2. **$ sudo pip3 install objection**
Installing objection in MacBook
3. **$ objection — gadget package_name explore**
Running Objection
4. **$ ios sslpinning disable**
Running bypass SSL pinning command

SSL Pinning bypass via Frida:
$ frida -U --codeshare snooze6/ios-pinning-disable -f package_name
or
$ frida -U -f package_name -l your_custom_ssl_pinning_bypass.js

Useful Sileo Repositories

- https://opa334.github.io
- https://ios.jjolano.me
- https://build.frida.re
- https://apt.thebigboss.org
- https://repo.co.kr

App Provisioning Profile

1. Open **Xcode** and create a sample proyect

2. In **Certificates, Identifiers & Profiles**, click **Profiles** in the sidebar, then click the **add button (+)** on the top left.

3. Under **Distribution**, select an App Store distribution profile for your platform, then **click Continue**.

4. Choose the **App ID** you used for development (the App ID that matches your bundle ID) from the App ID pop-up menu, then **click Continue**.

5. If you use **automatic signing** during development, choose one of the **Xcode given options**

6. **Select your distribution** certificate, then **click Continue**.

7. Enter a profile name, then **click Generate.**

8. **Click Download**.

libimobiledevice

Installation:
1. **$ sudo apt-get install build-essential pkg-config checkinstall git autoconf automake libtool-bin libplist-dev libimobiledevice-dev libzip-dev usbmuxd**
2. **$ git clone** https://github.com/libimobiledevice/ideviceinstaller.git
3. **$ cd ideviceinstaller**
4. **$ ./autogen.sh**
5. **$ make**
6. **$ sudo make install**

Useful Commands:
$ ideviceinfo -l
Show information about a connected device
$ ideviceprovision list
Manage provisioning profiles on a device
$ ideviceinstaller install <ipaFile>
Installing Ipa files into the device
$ ideviceenterrecovery
Make a device enter recovery mode
$ iproxy 2222 44
Starting iproxy binding port 44 (Palera1n default SSH port) to 2222
$ idevicesyslog -m <appname> (--match STRING)
Relay syslog of a connected device
$ idevicebackup2 list
Create or restore backups for devices running iOS 4 or later