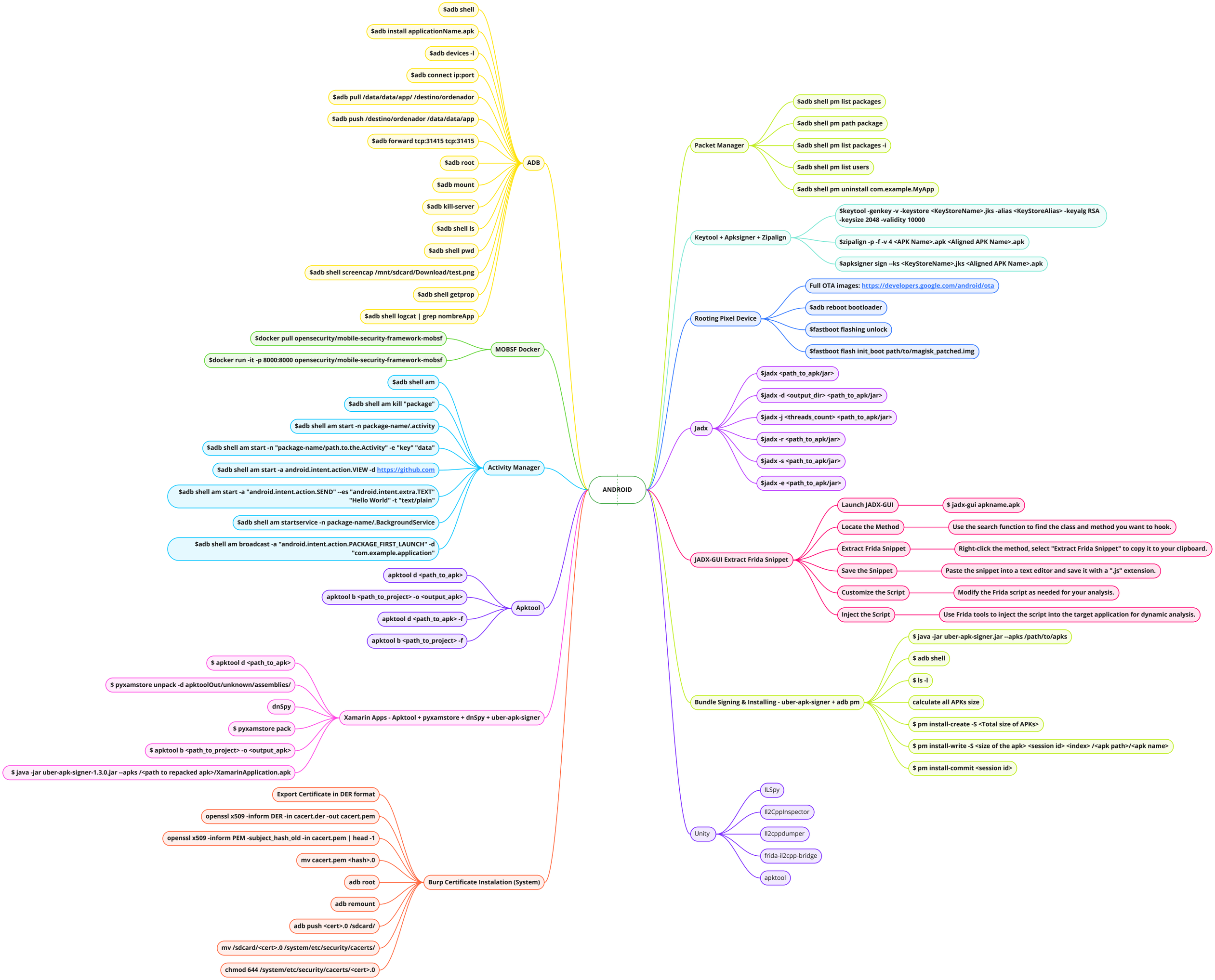
Android Cheatsheet MindMap



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.

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.