Tools

1- APKLeaks

Apkleaks ↗ is an open-source utility designed for static analysis of Android APK files, with a primary focus on identifying sensitive data such as API keys, URLs, AWS S3 buckets, and Firebase URLs. This tool automates the process of string analysis, facilitating the detection of hardcoded secrets and potential security vulnerabilities.

installation :

```
$ git clone https://github.com/dwisiswant0/apkleaks
$ cd apkleaks/
$ pip3 install -r requirements.txt
```

Usage

```
$ apkleaks -f ~/path/to/file.apk
# from Source
$ python3 apkleaks.py -f ~/path/to/file.apk
```

To run the tool using these custom rules, use the following command:

```
$ apkleaks -f /path/to/file.apk -p rules.json -o ~/Documents/apkleaks-results.txt
```

Arguments (disassembler)

We give user complete discretion to pass the disassembler arguments. For example, if you want to activate threads in `jadx` decompilation process, you can add it with `-a/--args` argument, example: `--args="--threads-count 5"`.

```
$ apkleaks -f /path/to/file.apk -a "--deobf --log-level DEBUG"
```

2-APKID

APKiD ↗ gives you information about how an APK was made. It identifies many compilers, packers, obfuscators, and other weird stuff.

For more information on what this tool can be used for, check out:

- **Android Compiler Fingerprinting ↗**
- Detecting Pirated and Malicious Android Apps with APKiD ↗

- APKiD: PEiD for Android Apps ↗
- APKiD: Fast Identification of AppShielding Products ↗

installation

```
git clone https://github.com/rednaga/APKiD
cd APKiD/
docker build . -t rednaga:apkid
docker/apkid.sh ~/reverse/targets/android/example/example.apk
[+] APKiD 2.1.0 :: from RedNaga :: rednaga.io
[*] example.apk!classes.dex
 |-> compiler : dx
```

Usage

```
usage: apkid [-h] [-v] [-t TIMEOUT] [-r] [--scan-depth SCAN_DEPTH]
             [--entry-max-scan-size ENTRY_MAX_SCAN_SIZE] [--typing
{magic,filename,none}] [-j]
             [-o DIR]
             [FILE [FILE ...]]

APKiD - Android Application Identifier v2.1.2

positional arguments:
  FILE                                  apk, dex, or directory

optional arguments:
  -h, --help                            show this help message and exit
  -v, --verbose                         log debug messages

scanning:
  -t TIMEOUT, --timeout TIMEOUT         Yara scan timeout (in seconds)
  -r, --recursive                       recurse into subdirectories
  --scan-depth SCAN_DEPTH               how deep to go when scanning
nested zips
  --entry-max-scan-size ENTRY_MAX_SCAN_SIZE  max zip entry size to scan in
bytes, 0 = no limit
  --typing {magic,filename,none}        method to decide which files to
scan

output:
  -j, --json                            output scan results in JSON
format
```

```
  -o DIR, --output-dir DIR                    write individual results here
(implies --json)
```

3-SSLUnpinning

An Xposed Module to bypass SSL certificate pinning

Description

If you need to intercept the traffic from an app which uses certificate pinning, with a tool like Burp Proxy, the SSLUnpinning will help you with this hard work! The SSLUnpinning through Xposed Framework, makes several hooks in SSL classes to bypass the certificate verifications for one specific app, then you can intercept all your traffic.

Usage

- install Xposed in your device (root access on Android 5.1 or later); http://repo.xposed.info/module/de.robv.android.xposed.installer
- Download the APK available here https://github.com/ac-pm/SSLUnpinning_Xposed or clone the project and compile;
- Install mobi.acpm.sslunpinning_latest.apk on a device with Xposed:

```
adb install mobi.acpm.sslunpinning_latest.apk
```

- SSLUnpinning will list the applications to choose from which will be unpinned;
- Ok! Now you can intercept all traffic from the chosen app.

Download

Get it from Xposed repo: http://repo.xposed.info/module/mobi.acpm.sslunpinning

```
تأكد من أن جهازك يعمل بصلاحيات الروت:

أداة SSLUnpinning تحتاج إلى صلاحيات Root لتثبيت Xposed Framework.

تنصيب Xposed Framework:

قم بتنزيل تطبيق Xposed Installer من هذا الرابط: Xposed Installer.
```

اتبع التعليمات الخاصة بتثبيته على جهاز Android.

تنزيل أداة SSLUnpinning:

قم بتنزيل ملف الـ APK الخاص بـ SSLUnpinning من الرابط: GitHub Link.

أو قم باستنساخ الكود وتجميعه باستخدام Android Studio.

تثبيت SSLUnpinning:

قم باستخدام الأمر التالي لتثبيت الأداة عبر ADB:

```
adb install mobi.acpm.sslunpinning_latest.apk
```

تفعيل الأداة داخل Xposed Framework:

بعد تثبيت SSLUnpinning، افتح تطبيق Xposed Installer.

قم بتفعيل المود الخاص بالأداة SSLUnpinning.

أعد تشغيل الجهاز لتفعيل المود بشكل كامل.

اختيار التطبيق المستهدف:

افتح تطبيق SSLUnpinning.

ستظهر قائمة بالتطبيقات المثبتة على جهازك.

اختر التطبيق الذي تريد تجاوز شهادة التحقق الخاصة به.

استخدام Burp Suite لاعتراض حركة المرور:

قم بضبط إعدادات جهازك لتوجيه حركة المرور إلى Burp Suite باستخدام البروكسي.

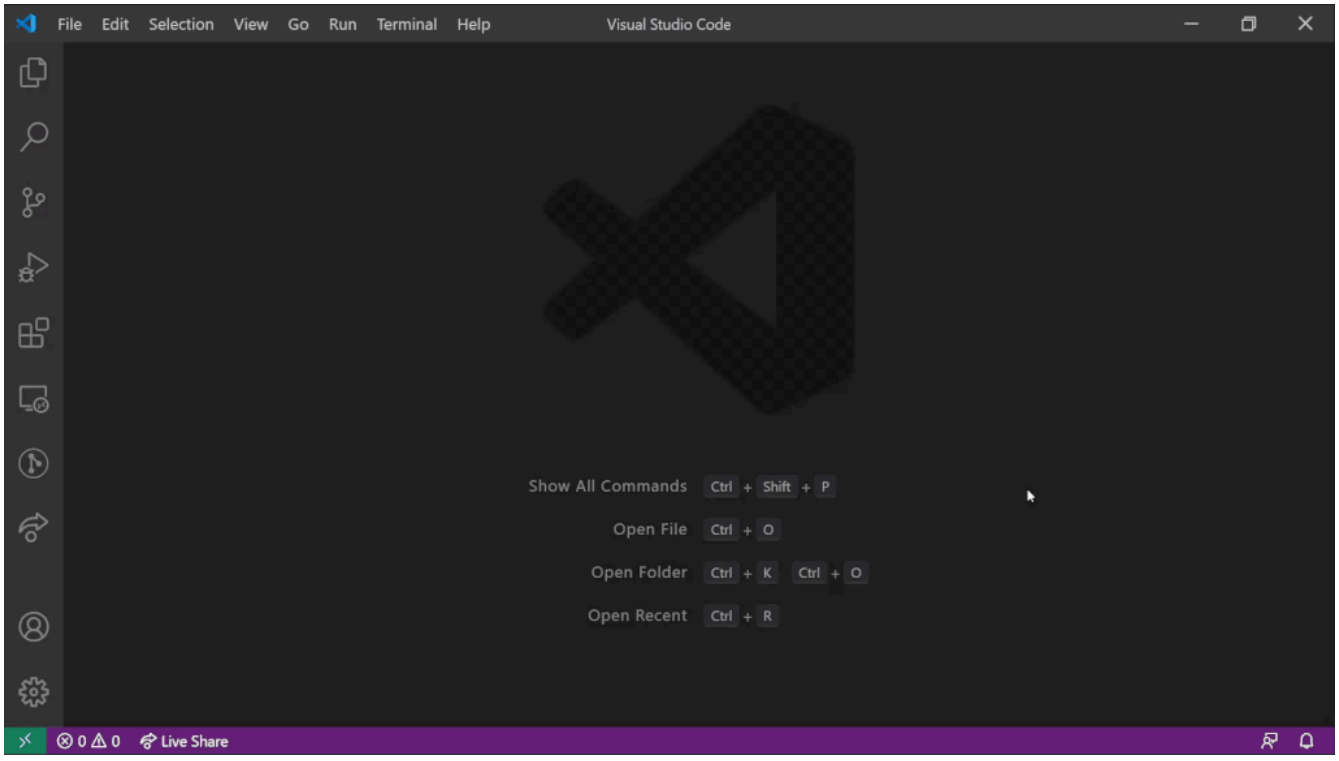
تأكد من أنك قمت بتثبيت شهادة CA الخاصة بـ Burp Suite على جهازك.

4- APKlab

APKLab ↗ is a **convenient Visual Studio Code extension leveraging tools such as *Apktool* and *jadx* to enable features including app unpacking, decompilation, code patching (e.g. for MITM), and repackaging straight from the IDE.**

Open APK or Apktool project

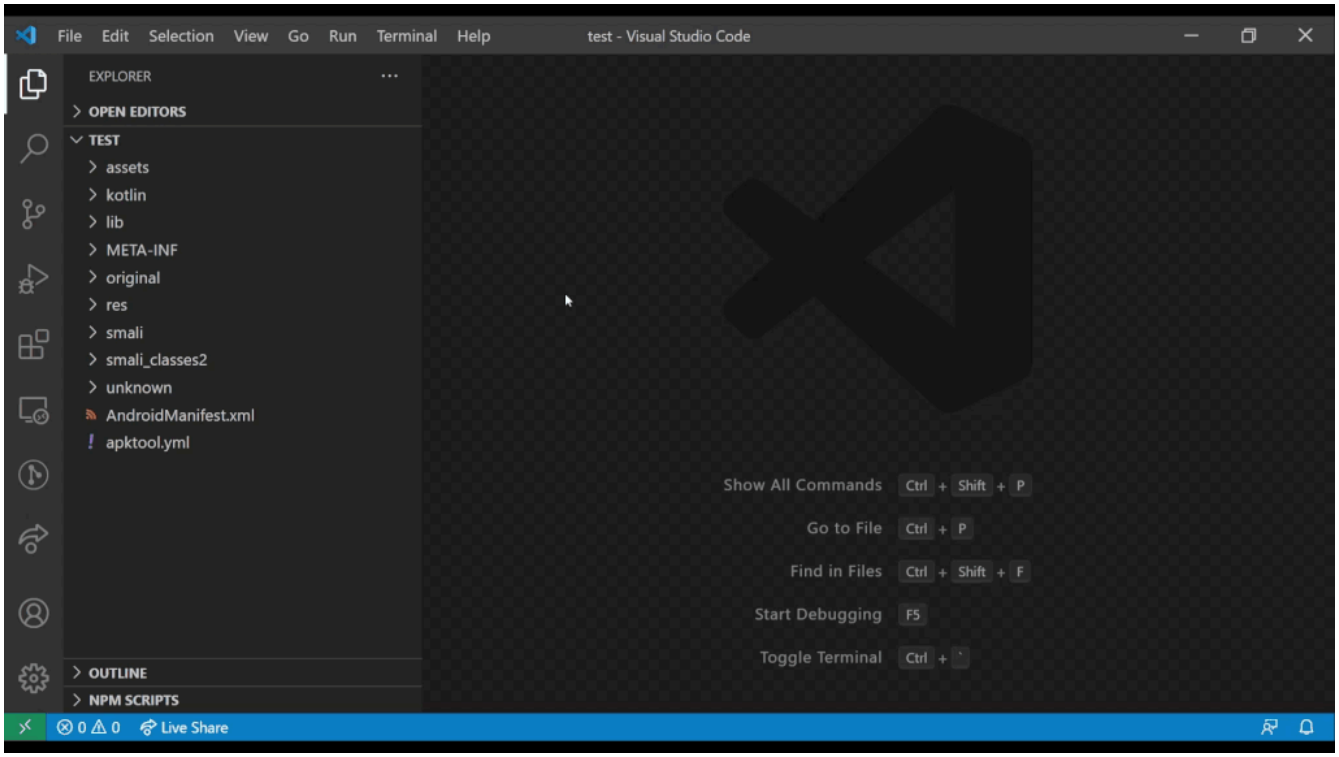
- Open the Command Palette (Ctrl+Shift+P) ➜ APKLab: Open an APK



- Or Just open an existing Apktool project folder

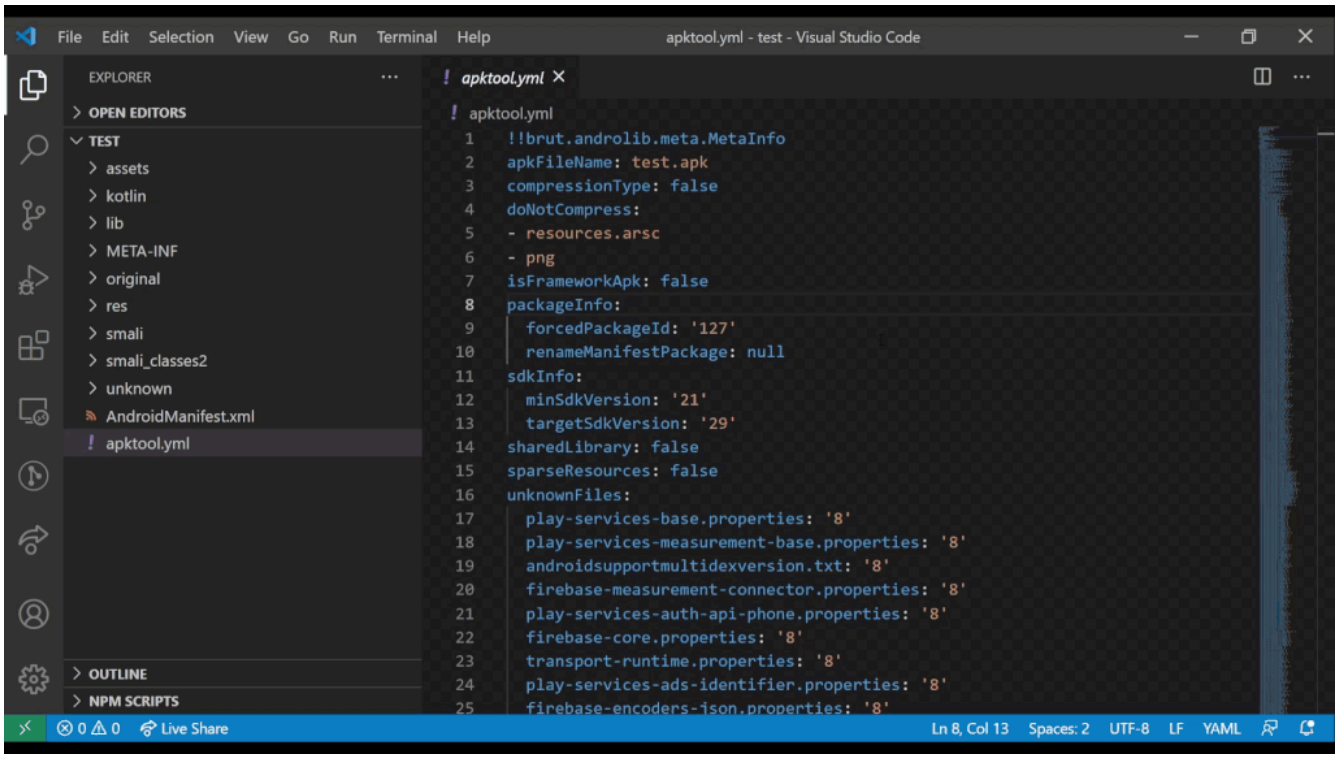
Apply MITM patch

- Right-Click on or inside `apktool.yml` file ➜ APKLab: Prepare for HTTPS inspection



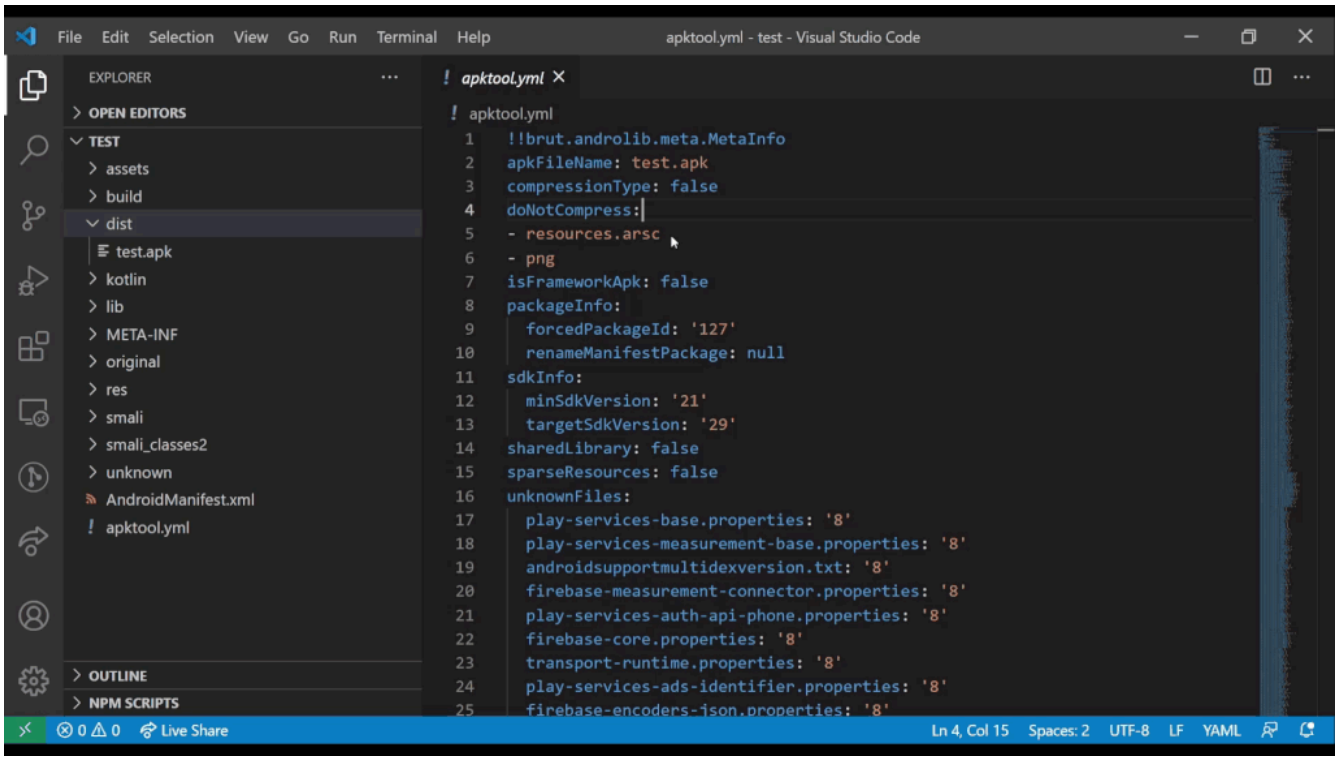
ReBuild and Sign APK

- Right-Click on or inside `apktool.yml` file ➜ APKLab: Rebuild the APK



Install APK to device

- Right-Click on `.apk` file (in `dist` directory) ➜ APKLab: Install the APK



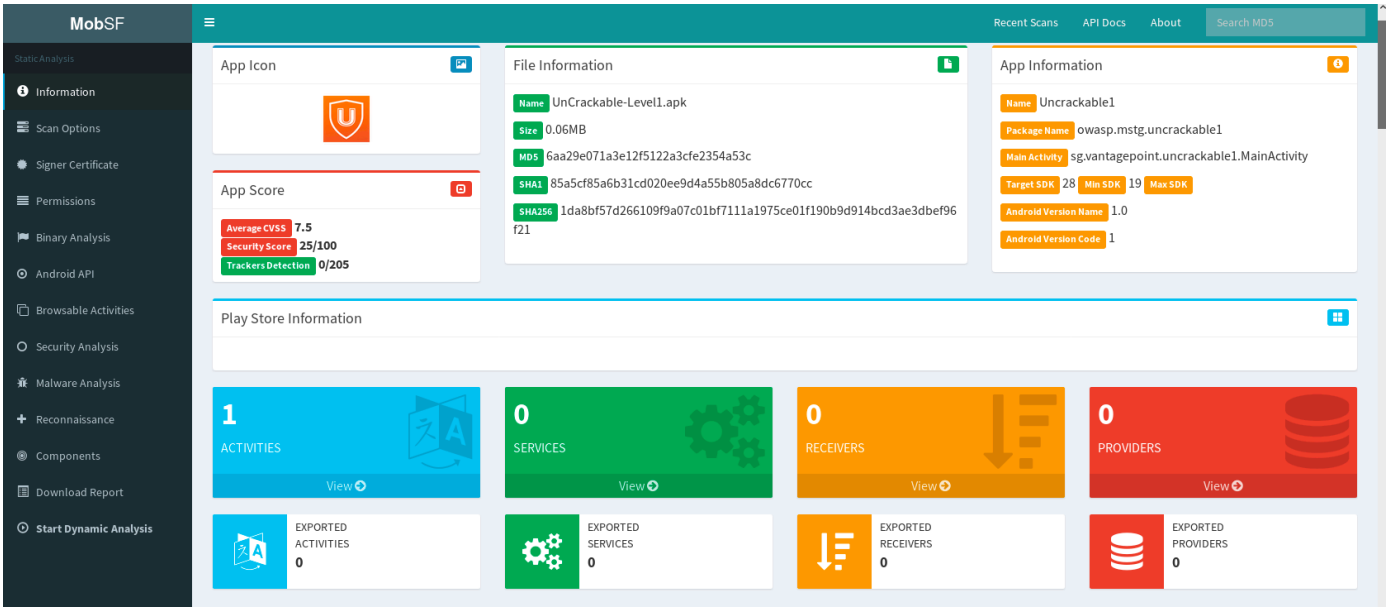
Clean ApkTool frameworks dir

- Open the Command Palette (Ctrl+Shift+P) ➜ APKLab: Empty ApkTool Framework Dir

---

5- MOBSF

After MobSF is done with its analysis, you will receive a one-page overview of all the tests that were executed. The page is split up into multiple sections giving some first hints on the attack surface of the application.



Documentation

Quick setup with docker

```
docker pull opensecurity/mobile-security-framework-mobsf:latest
docker run -it --rm -p 8000:8000 opensecurity/mobile-security-framework-mobsf:latest

# Default username and password: mobsf/mobsf
```