Android Security Testing

The following topics will be covered in this lab:

- Secure source code review patterns for Android
- Privacy and sensitive information review
- General process of APK security analysis
- Static secure code scanning with QARK

Privacy scanning with Androwarn

To automate the privacy scanning with APK, we can use the tool Androwarn which is a Python script to do the privacy information scanning.

Step 1 -- scanning of an APK

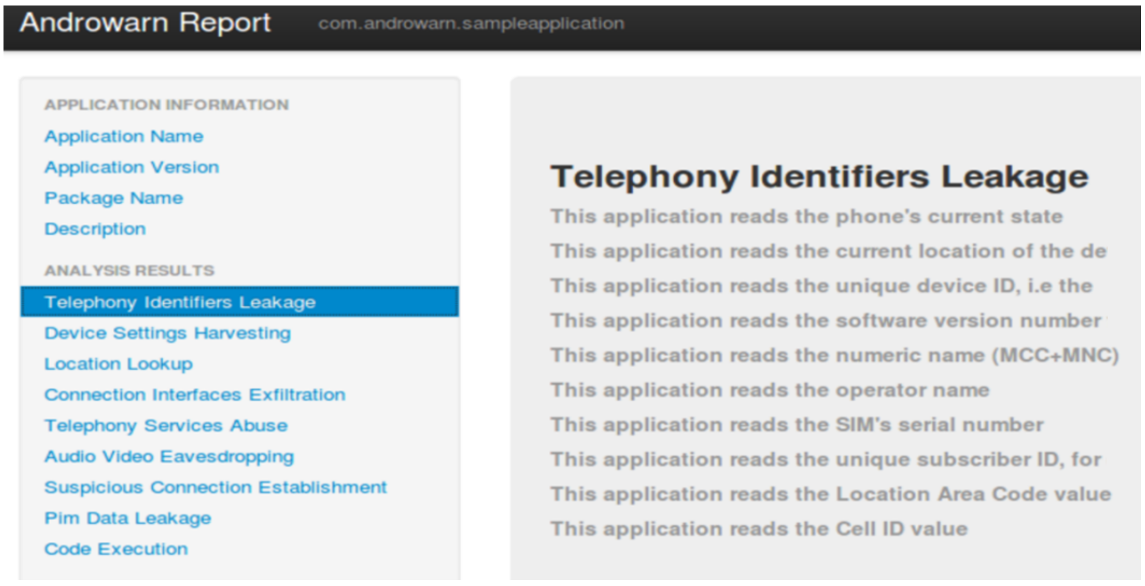
The execution of Androwarn takes some parameters, such as the APK, the report format, the level of verbosity, and the lookup to Google Play. The Google Play lookup is recommended to be disabled if the testing environment can't connect to the internet, as follows:

```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab07\androwarn

python androwarn.py -i ./_SampleApplication/bin/SampleApplication.apk -r html -v 3
```

For detailed usage of Androwarn, refer to python [androwarn.py -h]

Step 2 -- review the report

If you have specified the HTML report output in the previous step, then the report will be generated in the current directory as follows:

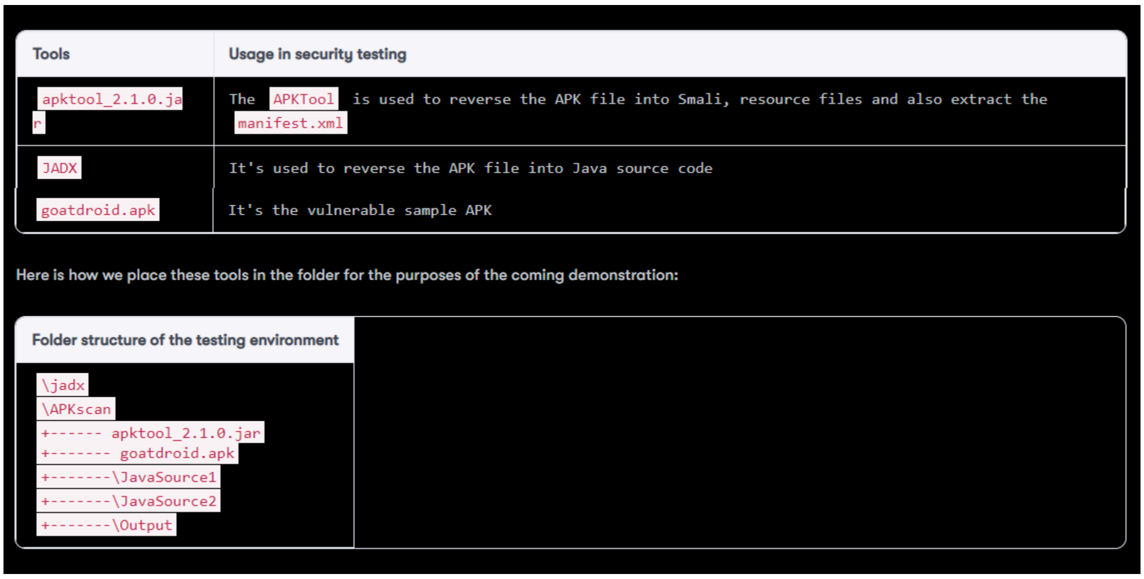


Telephony Identifiers Leakage

This application reads the phone's current state
This application reads the current location of the de
This application reads the unique device ID, i.e the
This application reads the software version number
This application reads the numeric name (MCC+MNC)
This application reads the operator name
This application reads the SIM's serial number
This application reads the unique subscriber ID, for
This application reads the Location Area Code value
This application reads the Cell ID value

General process of APK security analysis

This is a list of the tools for the reverse engineering of APK and security analysis:



Tools	Usage in security testing
`apktool_2.1.0.jar`	The `APKTool` is used to reverse the APK file into Smali, resource files and also extract the `manifest.xml`
`JADX`	It's used to reverse the APK file into Java source code
`goatdroid.apk`	It's the vulnerable sample APK

Here is how we place these tools in the folder for the purposes of the coming demonstration:

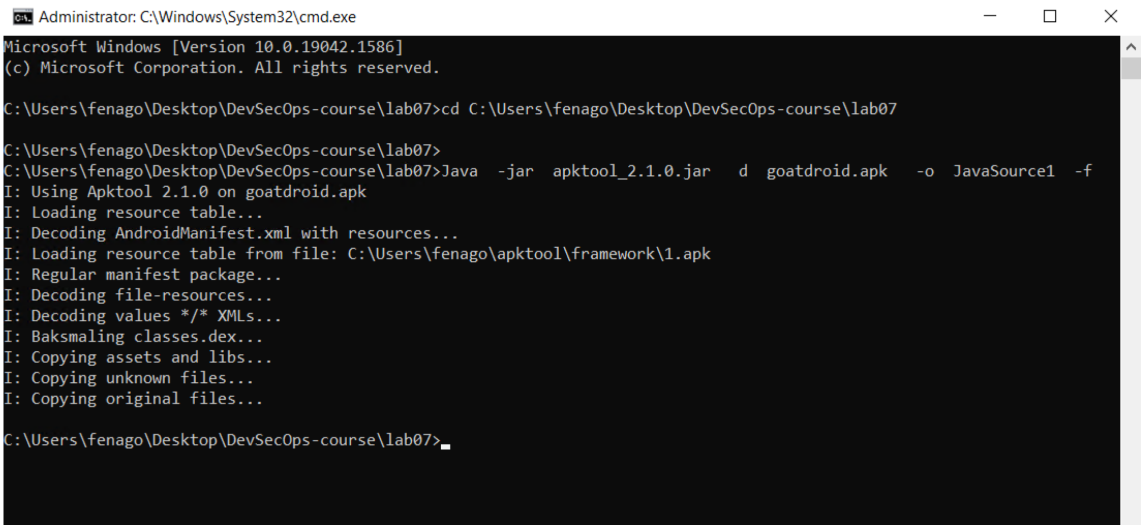
Folder structure of the testing environment

`\jadx`
`\APKscan`
`+------ apktool_2.1.0.jar`
`+------- goatdroid.apk`
`+-------\JavaSource1`
`+-------\JavaSource2`
`+-------\Output`

Step 1 -- use APKTool to reverse the APK to Manifest.xml, Smali and resources

The purpose of this step is to generate the Smali, resource files, and [manifest.xml] for initial security analysis. There are some security issues that can be identified by these file types, such as sensitive information exposure and incorrect permission settings, as follows:

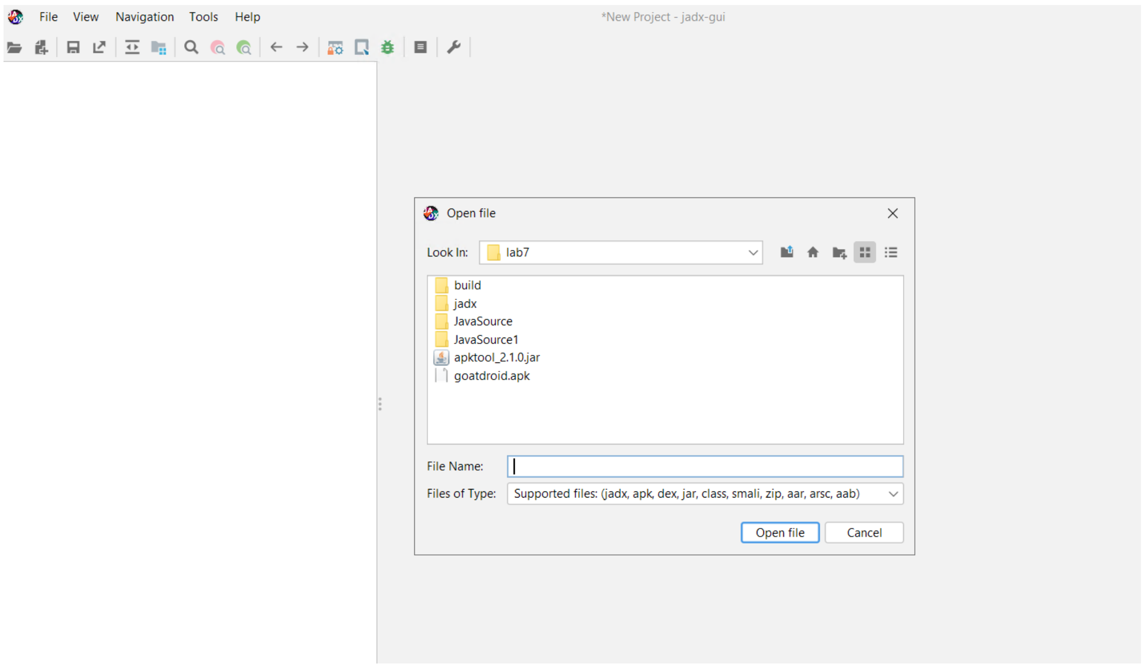
```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab07

Java  -jar  apktool_2.1.0.jar   d  goatdroid.apk   -o  JavaSource1  -f
```

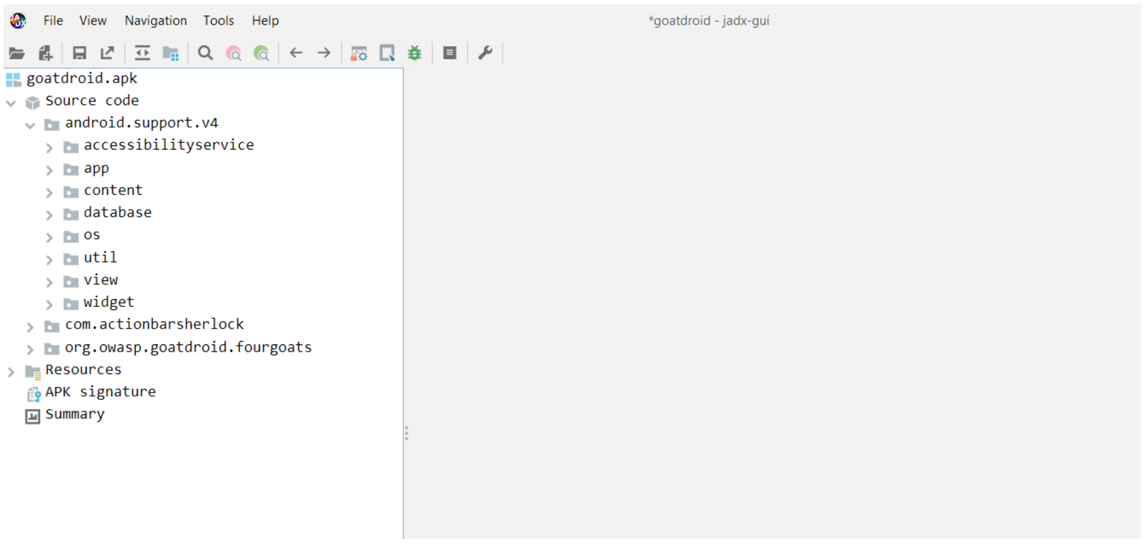


```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\fenago\Desktop\DevSecOps-course\lab07>cd C:\Users\fenago\Desktop\DevSecOps-course\lab07

C:\Users\fenago\Desktop\DevSecOps-course\lab07>
C:\Users\fenago\Desktop\DevSecOps-course\lab07>Java  -jar  apktool_2.1.0.jar   d  goatdroid.apk   -o  JavaSource1  -f
I: Using Apktool 2.1.0 on goatdroid.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\fenago\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values */* XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

C:\Users\fenago\Desktop\DevSecOps-course\lab07>_
```

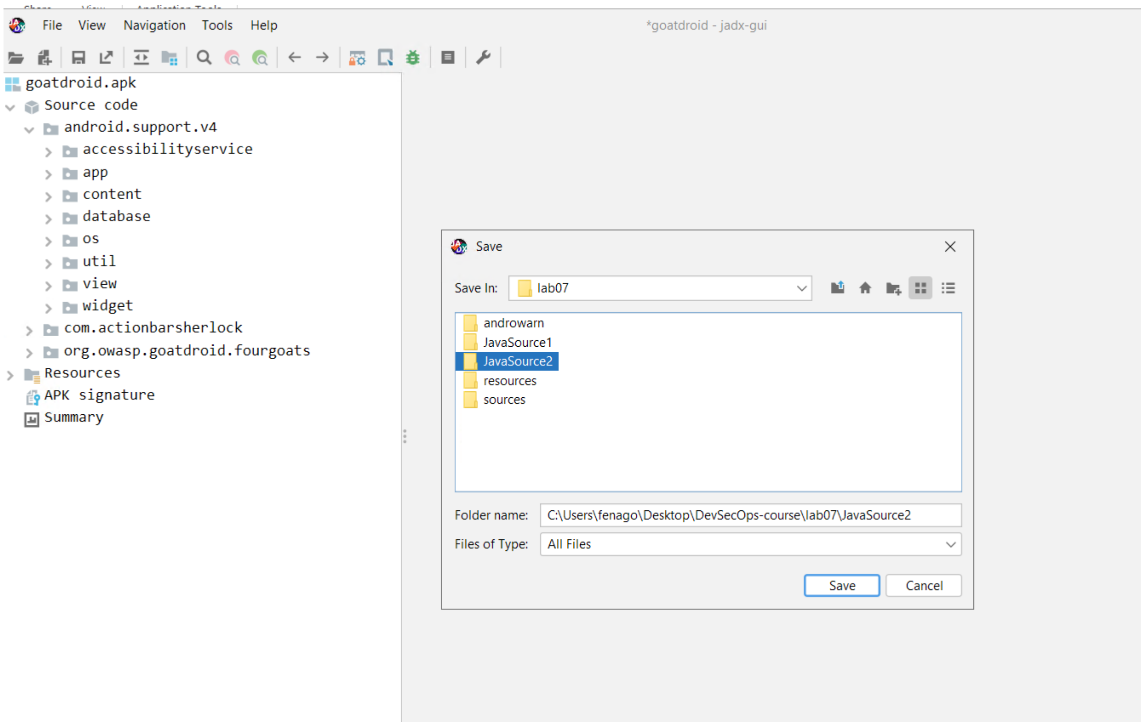
Step 2 -- use JADX to reverse the APK into Java source code

This will reverse the APK into Java source code. Then, we can do static secure code scanning in the next setup. Launch `jadx-gui-1.3.4.exe` located in **Desktop\DevSecOps-course\lab07** folder to start Jadx GUI:





```
goatdroid.apk
  Source code
    android.support.v4
      accessibilityservice
      app
      content
      database
      os
      util
      view
      widget
    com.actionbarsherlock
    org.owasp.goatdroid.fourgoats
  Resources
  APK signature
  Summary
```



Folder name: C:\Users\fenago\Desktop\DevSecOps-course\lab07\JavaSource2

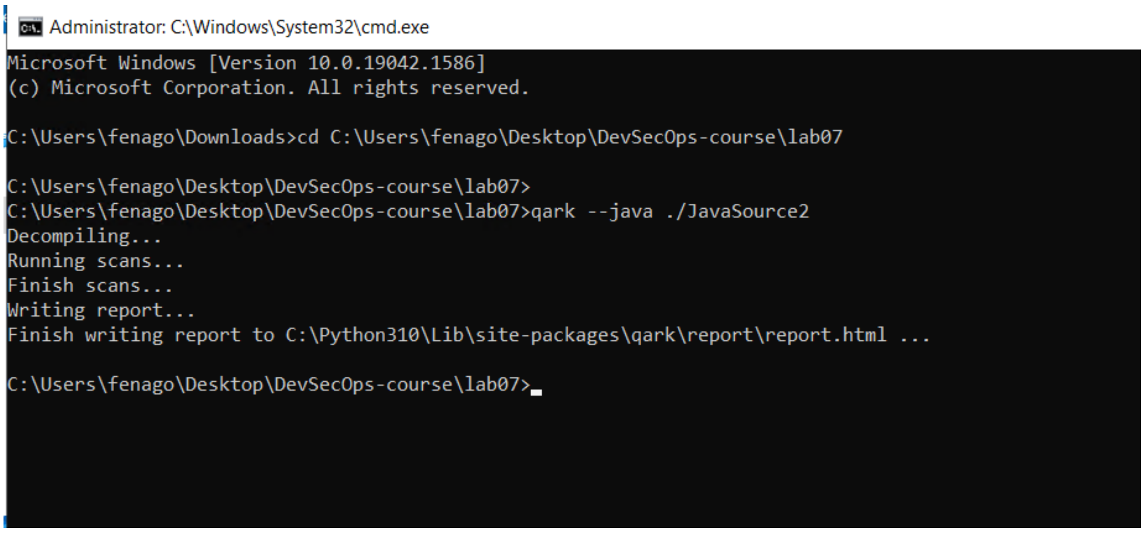
Files of Type: All Files

To save Java source code, Click **File** and select **Save All** and save in `JavaSource2` folder.

Step 3 -- APK scanning with QARK

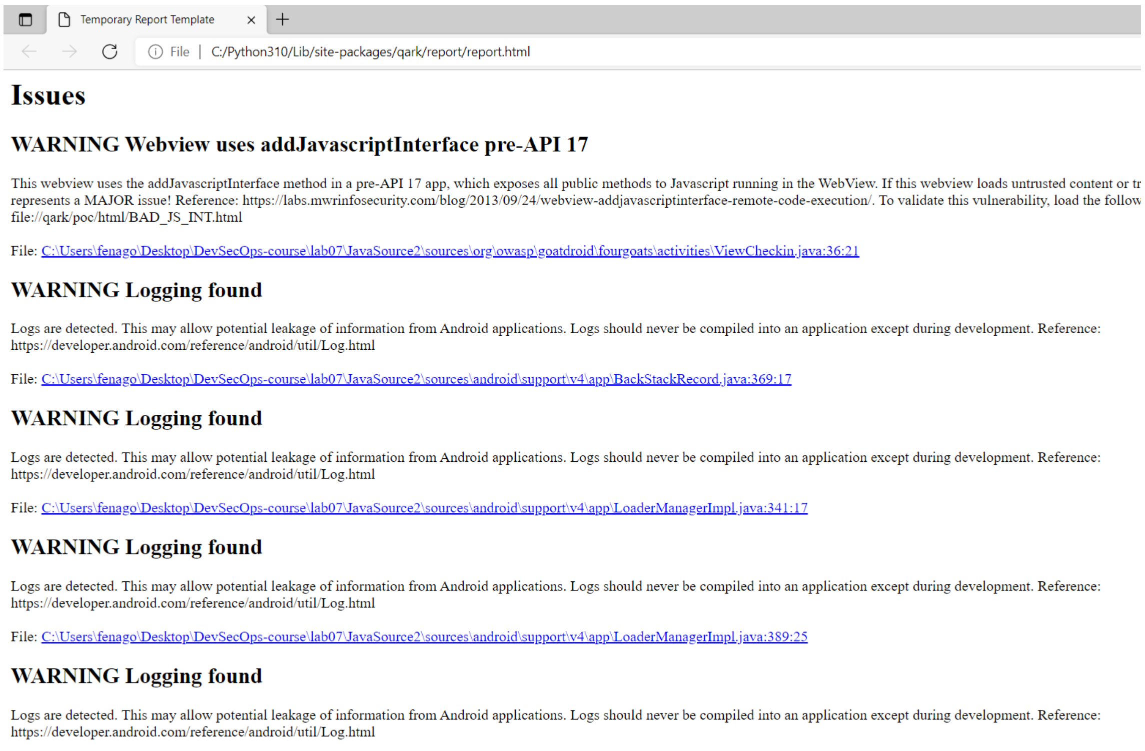
To scan the APK, execute the python script [qarkMain.py] with parameters, as follows:

```
cd C:\Users\fenago\Desktop\DevSecOps-course\lab07

qark --java ./JavaSource2
```


```
Administrator: C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19042.1586]
(c) Microsoft Corporation. All rights reserved.

C:\Users\fenago\Downloads>cd C:\Users\fenago\Desktop\DevSecOps-course\lab07

C:\Users\fenago\Desktop\DevSecOps-course\lab07>
C:\Users\fenago\Desktop\DevSecOps-course\lab07>qark --java ./JavaSource2
Decompiling...
Running scans...
Finish scans...
Writing report...
Finish writing report to C:\Python310\Lib\site-packages\qark\report\report.html ...

C:\Users\fenago\Desktop\DevSecOps-course\lab07>_
```

Step 3 -- review the results

The report will be generated under the [C:\Python310\Lib\site-packages\qark\report\report.html]. The following screenshot shows the scanning report of the `goatdroid.apk` source code:


Temporary Report Template
File | C:/Python310/Lib/site-packages/qark/report/report.html

Issues

WARNING Webview uses addJavascriptInterface pre-API 17

This webview uses the addJavascriptInterface method in a pre-API 17 app, which exposes all public methods to Javascript running in the WebView. If this webview loads untrusted content or tr
represents a MAJOR issue! Reference: https://labs.mwrinfosecurity.com/blog/2013/09/24/webview-addjavascriptinterface-remote-code-execution/. To validate this vulnerability, load the follow
file://qark/poc/html/BAD_JS_INT.html

File: C:\Users\fenago\Desktop\DevSecOps-course\lab07\JavaSource2\sources\org\owasp\goatdroid\fourgoats\activities\ViewCheckin.java:36:21

WARNING Logging found

Logs are detected. This may allow potential leakage of information from Android applications. Logs should never be compiled into an application except during development. Reference: https://developer.android.com/reference/android/util/Log.html

File: C:\Users\fenago\Desktop\DevSecOps-course\lab07\JavaSource2\sources\android\support\v4\app\BackStackRecord.java:369:17

WARNING Logging found

Logs are detected. This may allow potential leakage of information from Android applications. Logs should never be compiled into an application except during development. Reference: https://developer.android.com/reference/android/util/Log.html

File: C:\Users\fenago\Desktop\DevSecOps-course\lab07\JavaSource2\sources\android\support\v4\app\LoaderManagerImpl.java:341:17

WARNING Logging found

Logs are detected. This may allow potential leakage of information from Android applications. Logs should never be compiled into an application except during development. Reference: https://developer.android.com/reference/android/util/Log.html

File: C:\Users\fenago\Desktop\DevSecOps-course\lab07\JavaSource2\sources\android\support\v4\app\LoaderManagerImpl.java:389:25

WARNING Logging found

Logs are detected. This may allow potential leakage of information from Android applications. Logs should never be compiled into an application except during development. Reference: https://developer.android.com/reference/android/util/Log.html

Summary

To automate these security and privacy security reviews, we applied different tools based on the scenario. Androwarn is used to do privacy scanning for any APK files. We used QARK to do the automated APK static security scanning.

In the next lab, we will discuss the infrastructure security for system hardening, secure communication and configurations.