Project Report: Android APK Security Analysis

Pradeep Kumar D S (M.S., Ph.D)

1. Scope of the Project

This project aims to perform in-depth security analysis of Android applications using a combination of static and dynamic analysis techniques. The project covers rooting an Android emulator, deploying Frida server, decompiling APKs, extracting security-relevant information, and monitoring network behavior. The goal is to create an automated framework that can be reused by researchers and security analysts for analyzing suspicious or vulnerable Android apps.

2. Limitations

1. The current setup requires a rooted emulator (non-rooted device support via Frida Gadget is not yet implemented).
2. Obfuscation detection is basic and limited to string matching.
3. Native code (e.g., ARM .so files) is not deeply analyzed.
4. Network traffic monitoring may not capture encrypted HTTPS data unless certificate pinning is bypassed.
5. Script assumes availability of external tools like adb, apktool, Frida, mitmproxy, etc.

3. Architecture

The architecture involves five major layers of components:

1. Emulator Layer – Android Virtual Device (API 28) with root access.
    a. Install the apk "net.programmierecke.radiodroid2" – which is used for testing
2. Frida Layer – Frida server running inside the emulator for hooking live methods.
3. Static Analysis Layer – Python script uses apktool and Java decompiler to extract code.
4. Dynamic Analysis Layer – Frida and mitmproxy for runtime and network behavior analysis.
5. Reporting Layer – Script generates logs and a final report on security findings.

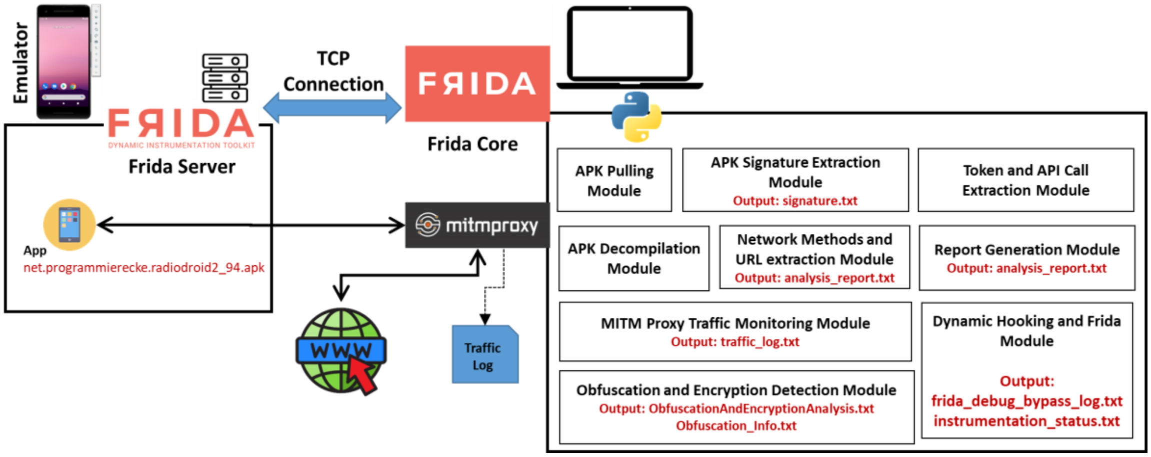

Emulator
FRIDA
DYNAMIC INSTRUMENTATION TOOLKIT
Frida Server
TCP Connection
FRIDA
Frida Core
App
net.programmierecke.radiodroid2_94.apk
mitmproxy
WWW
Traffic Log
APK Pulling Module
APK Signature Extraction Module
Output: signature.txt
Token and API Call Extraction Module
APK Decompilation Module
Network Methods and URL extraction Module
Output: analysis_report.txt
Report Generation Module
Output: analysis_report.txt
MITM Proxy Traffic Monitoring Module
Output: traffic_log.txt
Dynamic Hooking and Frida Module
Output:
frida_debug_bypass_log.txt
instrumentation_status.txt
Obfuscation and Encryption Detection Module
Output: ObfuscationAndEncryptionAnalysis.txt
Obfuscation_Info.txt

Figure 1 - Architecture of Security Analysis Tool

4. Explanation of Architecture

The architecture of the Android APK Security Analysis tool (shown in Figure 1) is divided into multiple well-defined modules, each responsible for a specific task in the analysis process. These modules work together to perform both static and dynamic analysis of APK files using tools like apktool, jadx, Frida, and mitmproxy.

1. **APK Pulling Module:** The pull_apk function is used to extract the APK file from a connected Android emulator or device. It uses ADB (Android Debug Bridge) commands to fetch the APK path of a given package and then pulls it to the local system. This is a crucial step before static and dynamic analysis.
2. **APK Signature Extraction Module:** The extract_apk_signature function checks the digital signature of the APK using the apksigner tool from the Android SDK. It saves the certificate and signature details to a text file for verification and documentation.
3. **APK Decompilation Module:** There are two functions involved here:
   - decompile_apk uses apktool to extract smali code and resources.
   - decompile_java uses jadx to convert APK files into Java source code. These outputs are stored in a specified output folder and are later scanned for keywords and vulnerabilities.
4. **Keyword and URL Extraction Module:** The grep_keywords function scans the decompiled .java and .smali files for sensitive keywords like token, login, and http. Similarly, extract_urls uses regular expressions to extract URLs from the source files, which could point to endpoints or external servers used in the app.
5. **Obfuscation and Encryption Detection Module:** The obfuscation_and_encryption_analysis function uses apkid to detect signs of code obfuscation and analyzes code files for encryption-related keywords (like aes, rsa, base64). These are indicators of potential hidden logic or sensitive operations.
6. **Dynamic Hooking and Frida Module:** The script uses Frida to perform dynamic analysis by spawning and hooking into the app. The hook_antidebug.js script is injected using Frida to bypass anti-debugging mechanisms. Additionally, frida-trace is used to trace key runtime functions like android.os.Debug. Outputs are saved in dedicated log files for further inspection.
7. **MITM Proxy Traffic Monitoring Module:** The start_mitmproxy function launches mitmdump with a custom script to intercept and log HTTP/S traffic from the app. It monitors potential data leaks and API interactions during app execution. The captured logs are saved for further token extraction and network analysis.
8. **Token and API Call Extraction Module:** extract_tokens_from_log parses network logs to identify leaked tokens (Bearer, JWT, etc.), and extract_api_calls scans source code for API calls and parameters. This helps understand how data is transmitted and if it's vulnerable.
9. **Report Generation Module:** The write_report function consolidates findings from static analysis, including URLs, parameters, encryption evidence, and keyword matches. It writes these insights into a report file. The report helps to understand the APK.

5. Future Works

The work can be improved by analyzing and incorporating the following:

1. **Detection of Exposed Components** - Analyze exported components such as Activities, Services, Broadcast Receivers, and Content Providers that may be unintentionally exposed to external applications, leading to privilege escalation or unauthorized access.
2. **Secure Intent Communication** - Investigate whether the application uses encryption when passing sensitive data via Intents, and assess the presence of Intent sniffing vulnerabilities.
3. **PendingIntent Security Analysis** - Examine whether PendingIntent objects are securely created, particularly in relation to implicit Intents and permission leakage. Assess if they expose sensitive data or grant unauthorized access to other applications.

4. **Custom Permission Model Evaluation** - Analyze whether the application defines and enforces custom permissions properly, and if those permissions are susceptible to misuse or privilege misassignment.
5. **Code Injection and Reflection Usage** - Detect the use of dynamic code execution mechanisms such as reflection, DexClassLoader, or native libraries, which may be used to hide malicious logic or bypass static analysis.
6. **Cryptographic Misuse Detection** - Extend the analysis to detect improper usage of cryptographic APIs (e.g., hardcoded keys, insecure encryption modes like ECB, or missing IVs).
7. **Third-Party SDK and Library Risk Analysis** - Evaluate the security implications of embedded third-party SDKs, which may introduce vulnerabilities through tracking, data exfiltration, or insufficient sandboxing.
8. **Dynamic Behavior Monitoring** - Incorporate runtime analysis using tools like Frida or Objection to monitor sensitive API calls, dynamic permissions, and behavioral anomalies that cannot be detected statically.

6. Conclusion

This approach provides a comprehensive framework to analyze Android applications using both static and dynamic methods. The use of an emulator, Frida, and mitmproxy makes it possible to uncover hidden behavior, detect insecure APIs, and identify potential security threats. With minor enhancements, this toolchain can be extended to real devices and obfuscated or native-heavy apps.

7. Manual Testing Procedure

Below is the list of tools and libraries used for manual testing of the project

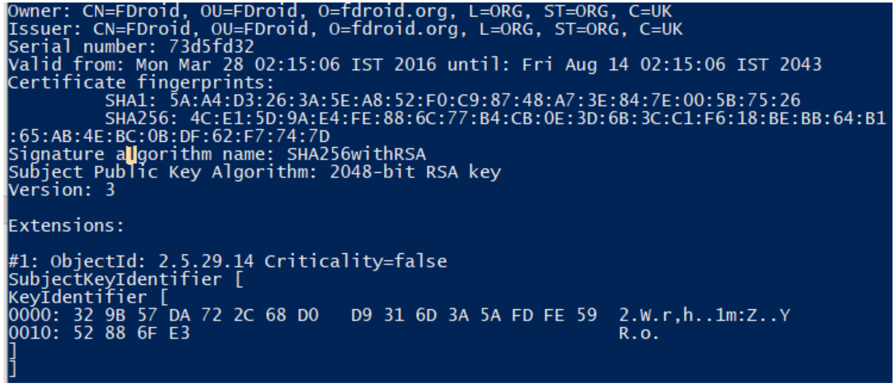
- Install 7z
- Install OpenSSL (choco install openssl -y)

Tool	Purpose	Platform	Command
apktool	Decompiles APK to Smali and resources (layout, manifest)	CLI	apktool d testapp.apk
jadx.bat	Decompiles APK to Java-like source code	CLI	jadx.bat –d output-dir testapp.apk
grep	Regex pattern extraction e.g. base64, decrypt etc	CLI	grep –l base64 –r output-dir
frida	Dynamic analysis and runtime instrumentation of adding bypassing debugging point hooks	CLI	frida –U –f testapp_package_name -l script.js
keytool	Inspect apk certificate and keystore	CLI	keytool -printcert -jarfile testapp.apk
mitmproxy	Intercept and inspect apps network traffic	GUI	mitmproxy --listen-host 0.0.0.0 --listen-port 8080

Steps to extract certificate:

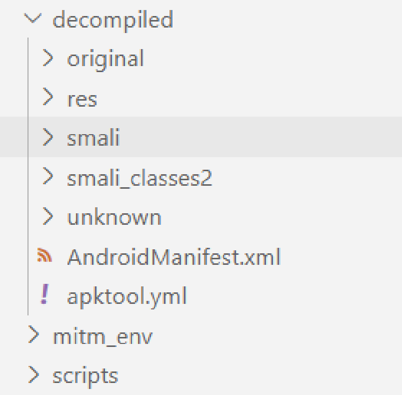
- Unzip the APK file to extract its contents (used 7z to extract manually):
  - unzip net.programmierecke.radiodroid2_94.apk -d net.programmierecke.radiodroid2_94
- Locate the .RSA certificate file inside the META-INF directory:
  - ls net.programmierecke.radiodroid2_94/META-INF/*.RSA

- Use keytool to print certificate information:
  - keytool -printcert -file net.programmierecke.radiodroid2_94/META-INF/A03005BB.RSA
- Convert the .RSA certificate to PEM format using OpenSSL:
  - openssl pkcs7 -in net.programmierecke.radiodroid2_94/META-INF/A03005BB.RSA -inform DER -print_certs -out A03005BB.pem
- Print full certificate details to a text file:
  - openssl x509 -in A03005BB.pem -text -noout >> certificate.txt
- Save the signature fingerprint and issuer details to a separate file (The signature fingerprint refers to a hash (digital digest) of the signing certificate used to sign an APK):
  - keytool -printcert -file net.programmierecke.radiodroid2_94/META-INF/A03005BB.RSA >> signature_fingerprint.txt



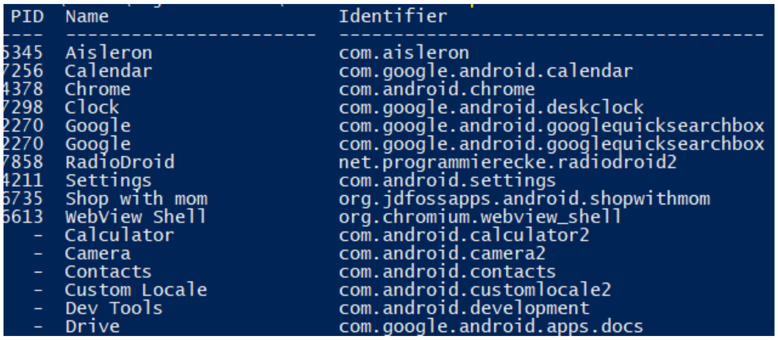
```
Owner: CN=FDroid, OU=FDroid, O=fdroid.org, L=ORG, ST=ORG, C=UK
Issuer: CN=FDroid, OU=FDroid, O=fdroid.org, L=ORG, ST=ORG, C=UK
Serial number: 73d5fd32
Valid from: Mon Mar 28 02:15:06 IST 2016 until: Fri Aug 14 02:15:06 IST 2043
Certificate fingerprints:
         SHA1: 5A:A4:D3:26:3A:5E:A8:52:F0:C9:87:48:A7:3E:84:7E:00:5B:75:26
         SHA256: 4C:E1:5D:9A:E4:FE:88:6C:77:B4:CB:0E:3D:6B:3C:C1:F6:18:BE:BB:64:B1
:65:AB:4E:BC:0B:DF:62:F7:74:7D
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 32 9B 57 DA 72 2C 68 D0   D9 31 6D 3A 5A FD FE 59  2.W.r,h..1m:Z..Y
0010: 52 88 6F E3                                        R.o.
]
]
```

- Using apktool and JADX to decompile the APK – the decompiled output will look like below:



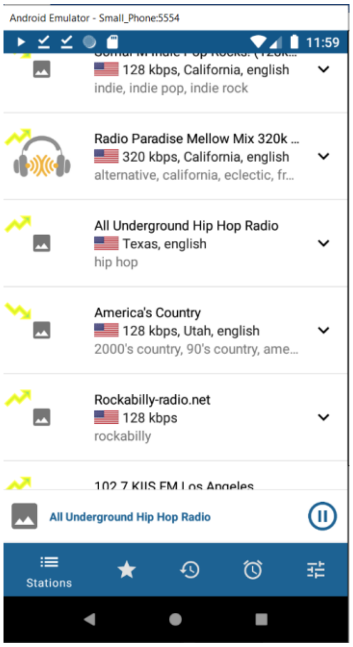
```
decompiled
  original
  res
  smali
  smali_classes2
  unknown
  AndroidManifest.xml
  apktool.yml
mitm_env
scripts
```

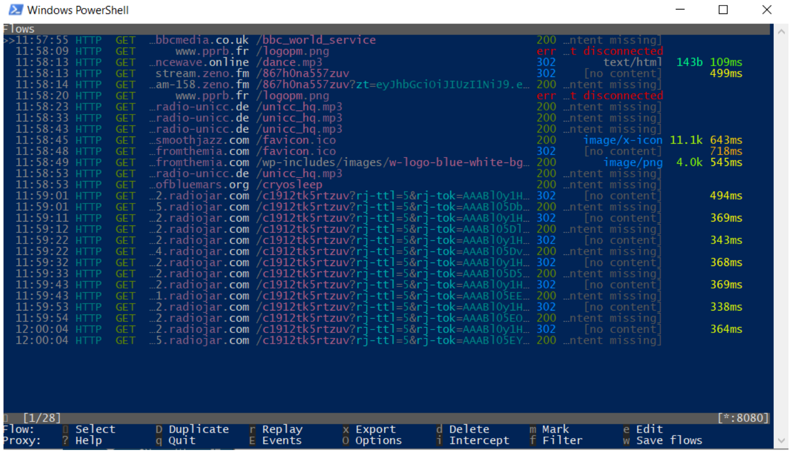
- Frida to check if our target app (RadioDroid) is started:
  - frida-ps –Uai



```
 PID  Name                      Identifier
----  ------------------------  ----------------------------------------
5345  Aisleron                  com.aisleron
7256  Calendar                  com.google.android.calendar
4378  Chrome                    com.android.chrome
7298  Clock                     com.google.android.deskclock
2270  Google                    com.google.android.googlequicksearchbox
2270  Google                    com.google.android.googlequicksearchbox
7858  RadioDroid                net.programmierecke.radiodroid2
4211  Settings                  com.android.settings
6735  Shop with mom             org.jdfossapps.android.shopwithmom
5613  WebView Shell             org.chromium.webview_shell
   -  Calculator                com.android.calculator2
   -  Camera                    com.android.camera2
   -  Contacts                  com.android.contacts
   -  Custom Locale             com.android.customlocale2
   -  Dev Tools                 com.android.development
   -  Drive                     com.google.android.apps.docs
```

- Using mitm gui tool to monitor the app traffic:
  - mitmproxy --listen-host 0.0.0.0 --listen-port 8080


Android Emulator - Small_Phone:5554
11:59
128 kbps, California, english
indie, indie pop, indie rock
Radio Paradise Mellow Mix 320k ...
320 kbps, California, english
alternative, california, eclectic, fr...
All Underground Hip Hop Radio
Texas, english
hip hop
America's Country
128 kbps, Utah, english
2000's country, 90's country, ame...
Rockabilly-radio.net
128 kbps
rockabilly
102.7 KIIS FM Los Angeles
All Underground Hip Hop Radio
Stations



```
Flows
>>11:57:55 HTTP  GET  …bbcmedia.co.uk /bbc_world_service                            200 …ntent missing]
  11:58:09 HTTP  GET     www.pprb.fr /logopm.png                                   err …t disconnected
  11:58:13 HTTP  GET  …ncewave.online /dance.mp3                                   302      text/html  143b 109ms
  11:58:13 HTTP  GET   stream.zeno.fm /867h0na557zuv                               302   [no content]       499ms
  11:58:14 HTTP  GET  …am-158.zeno.fm /867h0na557zuv?zt=eyJhbGciOiJIUzI1NiJ9.e…    200 …ntent missing]
  11:58:20 HTTP  GET     www.pprb.fr /logopm.png                                   err …t disconnected
  11:58:23 HTTP  GET  …radio-unicc.de /unicc_hq.mp3                                200 …ntent missing]
  11:58:33 HTTP  GET  …radio-unicc.de /unicc_hq.mp3                                200 …ntent missing]
  11:58:43 HTTP  GET  …radio-unicc.de /unicc_hq.mp3                                200 …ntent missing]
  11:58:45 HTTP  GET  …smoothjazz.com /favicon.ico                                 200   image/x-icon 11.1k 643ms
  11:58:48 HTTP  GET  …fromthemia.com /favicon.ico                                 302   [no content]       718ms
  11:58:49 HTTP  GET  …fromthemia.com /wp-includes/images/w-logo-blue-white-bg…    200      image/png  4.0k 545ms
  11:58:53 HTTP  GET  …radio-unicc.de /unicc_hq.mp3                                200 …ntent missing]
  11:58:53 HTTP  GET  …ofbluemars.org /cryosleep                                   200 …ntent missing]
  11:59:01 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       494ms
  11:59:01 HTTP  GET  …5.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05Db…   200 …ntent missing]
  11:59:11 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       369ms
  11:59:12 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05Dl…   200 …ntent missing]
  11:59:22 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       343ms
  11:59:22 HTTP  GET  …4.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05Dv…   200 …ntent missing]
  11:59:32 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       368ms
  11:59:33 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05D5…   200 …ntent missing]
  11:59:43 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       369ms
  11:59:43 HTTP  GET  …1.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05EE…   200 …ntent missing]
  11:59:53 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       338ms
  11:59:54 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05EO…   200 …ntent missing]
  12:00:04 HTTP  GET  …2.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl0y1H…   302   [no content]       364ms
  12:00:04 HTTP  GET  …5.radiojar.com /c1912tk5rtzuv?rj-ttl=5&rj-tok=AAABl05EY…   200 …ntent missing]

   [1/28]                                                                                          [*:8080]
Flow:    Select     D Duplicate   r Replay   x Export    d Delete     m Mark    e Edit
Proxy: ? Help       q Quit        E Events   O Options   i Intercept  f Filter  w Save flows
```

Evaluation: Compared the key generated with the Automation output.

8. How To: Automation using Python Script

For this implementation, I have used a rooted Android emulator, run Frida server, and use a Python script for analyzing Android apps.

Step 1: Create an Android Emulator (API 28)

1. Use Android Studio > AVD Manager to create an emulator with API 28.
2. API 28 is easy to root and works well with Frida.

Step 2: Root the Emulator and Start Frida Server

- Push the frida-server to the emulator using the commands below:
  - adb root
    adb remount
    adb push frida-server /data/local/tmp/
    adb shell chmod 755 /data/local/tmp/frida-server
- Start the frida-server:
  - adb shell
    cd /data/local/tmp
    ./frida-server &
- This runs the server in background, and now we can monitor apps.

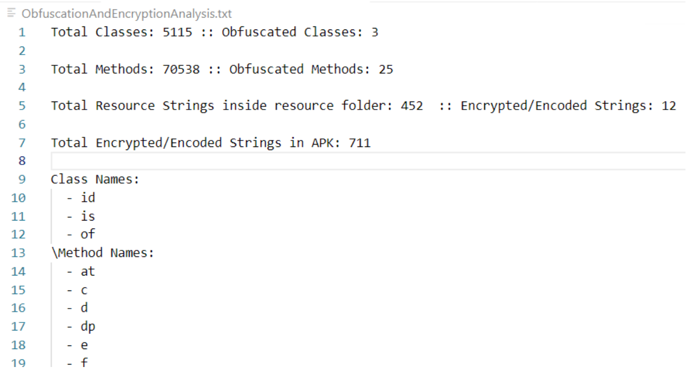
Step 3: Python Script for APK Analysis

This Python script pulls the APK from emulator, decompiles it, and searches for important patterns. Below are the python modules and its objective:

- pull_apk(pkg): Downloads APK from emulator
- decompile_apk(): Converts APK to smali
- decompile_java(): Converts APK to Java
- grep_keywords(): Searches for login tokens, urls, etc.
- extract_urls(): Finds all web URLs used in app

- extract_api_calls(): Finds used Android API methods related to Network communication and authentication, and by using regex on the smali files the method extracts all relevant methods and its details:

  keywords = ["http", "https", "token", "Auth", "Bearer", "login", "OkHttpClient", "Retrofit", "HttpURLConnection", "authenticate", "Session", "JWT"]
- Two obfuscation and encryption methods were implemented as a process of debugging using Java files and using Smali files
  - Using Java Files:
    - **Output File Name: ObfuscationAndEncryptionAnalysis.txt**
    - *obfuscation_and_encryption_analysis_java():*



ObfuscationAndEncryptionAnalysis.txt

```
Total Classes: 5115 :: Obfuscated Classes: 3

Total Methods: 70538 :: Obfuscated Methods: 25

Total Resource Strings inside resource folder: 452  :: Encrypted/Encoded Strings: 12

Total Encrypted/Encoded Strings in APK: 711

Class Names:
  - id
  - is
  - of
\Method Names:
  - at
  - c
  - d
  - dp
  - e
  - f
```

    - From above table we can objserve that out of 5115 classes only 3 were obfuscated (i.e. 0.058%)
    - Out of 70538 methods only 25 methods are obfuscated (i.e. 0.035%)
    - Out of 452 resource strings, only 12 are encrypted/encoded (i.e. 2.65%)
  - Using Smali Files:
    - **Output File Name: Obfuscation_Info.txt**
    - *obfuscation_and_encryption_analysis()*: Checks the apps obfuscation model (Proguard). A sample obfuscation output is given below from analysis of *net.programmierecke.radiodroid2.apk*.

```
[+] APKiD 3.0.0 :: from RedNaga :: rednaga.io
[*] ./net.programmierecke.radiodroid2.apk!classes.dex
 |-> anti_vm : Build.FINGERPRINT check, Build.MANUFACTURER check
 |-> compiler : r8
[*] ./net.programmierecke.radiodroid2.apk!classes2.dex
 |-> compiler : r8 without marker (suspicious)
```

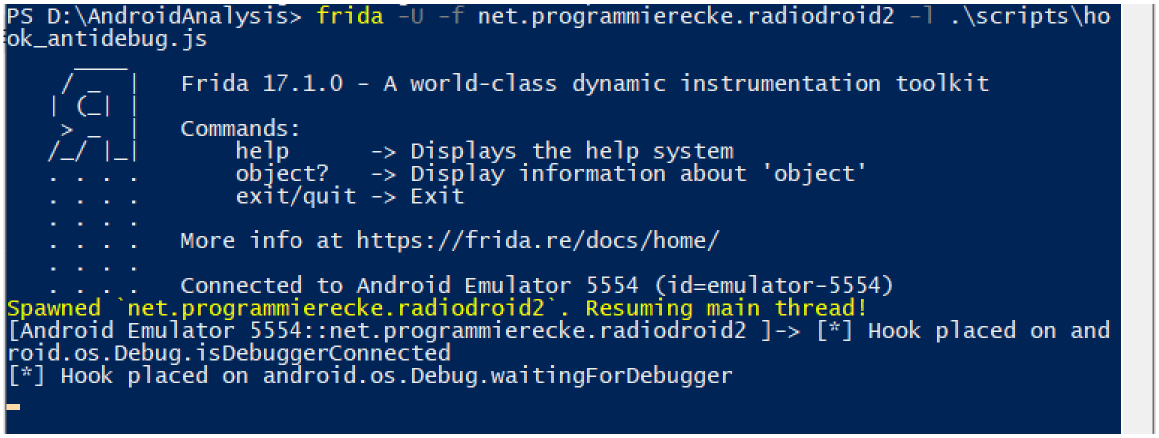
write_report(): Saves everything to a report

Step 4: Start Dynamic Analysis (Using Frida)

Run Frida to monitor app behavior while it's running.

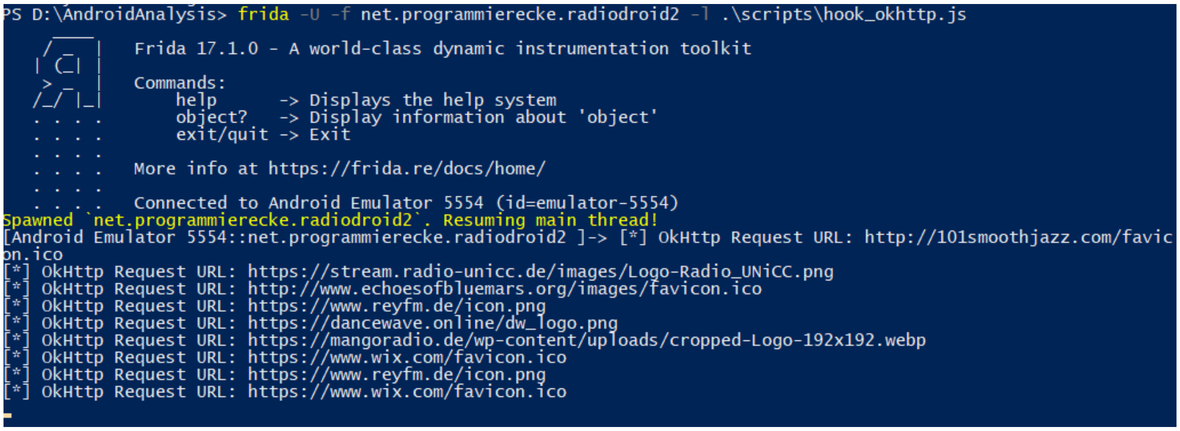
- Hook app using a Frida script:

  ```
  frida -U -f <package_name> -l ./scripts/hook_antidebug.js
  ```

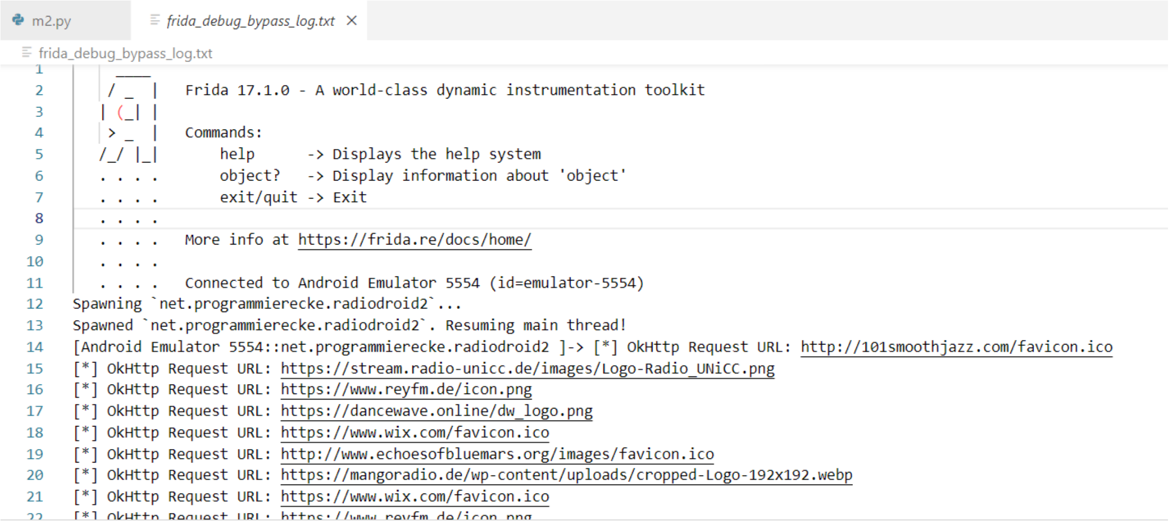


```
PS D:\AndroidAnalysis> frida -U -f net.programmierecke.radiodroid2 -l .\scripts\ho
ok_antidebug.js
     ____
    / _  |   Frida 17.1.0 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `net.programmierecke.radiodroid2`. Resuming main thread!
[Android Emulator 5554::net.programmierecke.radiodroid2 ]-> [*] Hook placed on and
roid.os.Debug.isDebuggerConnected
[*] Hook placed on android.os.Debug.waitingForDebugger
```

Test Result for OkHTTP Hook



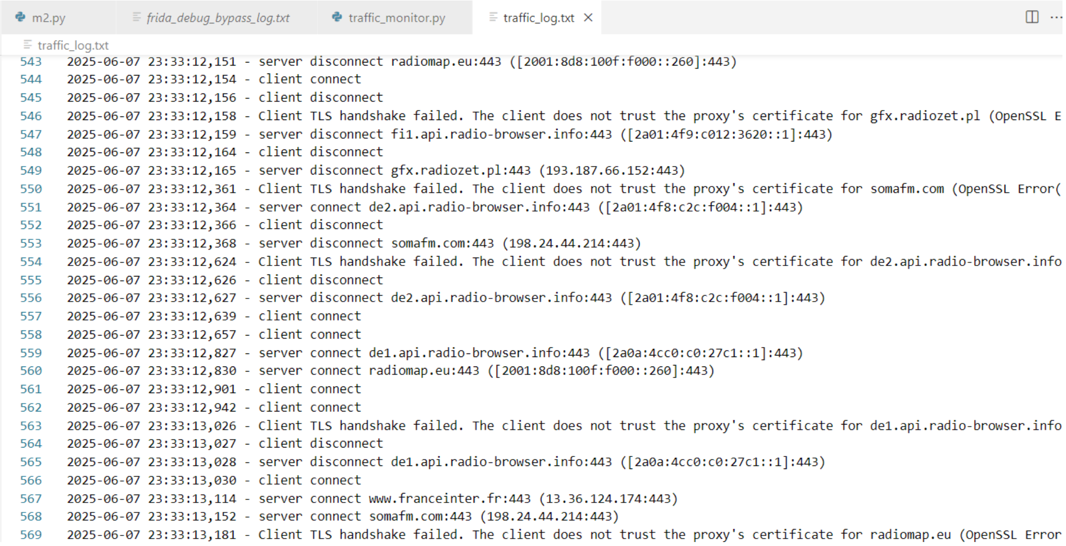
```
PS D:\AndroidAnalysis> frida -U -f net.programmierecke.radiodroid2 -l .\scripts\hook_okhttp.js
     ____
    / _  |   Frida 17.1.0 - A world-class dynamic instrumentation toolkit
   | (_| |
    > _  |   Commands:
   /_/ |_|       help      -> Displays the help system
   . . . .       object?   -> Display information about 'object'
   . . . .       exit/quit -> Exit
   . . . .
   . . . .   More info at https://frida.re/docs/home/
   . . . .
   . . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawned `net.programmierecke.radiodroid2`. Resuming main thread!
[Android Emulator 5554::net.programmierecke.radiodroid2 ]-> [*] OkHttp Request URL: http://101smoothjazz.com/favic
on.ico
[*] OkHttp Request URL: https://stream.radio-unicc.de/images/Logo-Radio_UNiCC.png
[*] OkHttp Request URL: http://www.echoesofbluemars.org/images/favicon.ico
[*] OkHttp Request URL: https://www.reyfm.de/icon.png
[*] OkHttp Request URL: https://dancewave.online/dw_logo.png
[*] OkHttp Request URL: https://mangoradio.de/wp-content/uploads/cropped-Logo-192x192.webp
[*] OkHttp Request URL: https://www.wix.com/favicon.ico
[*] OkHttp Request URL: https://www.reyfm.de/icon.png
[*] OkHttp Request URL: https://www.wix.com/favicon.ico
```



```
  ____
 / _  |   Frida 17.1.0 - A world-class dynamic instrumentation toolkit
| (_| |
 > _  |   Commands:
/_/ |_|       help      -> Displays the help system
. . . .       object?   -> Display information about 'object'
. . . .       exit/quit -> Exit
. . . .
. . . .   More info at https://frida.re/docs/home/
. . . .
. . . .   Connected to Android Emulator 5554 (id=emulator-5554)
Spawning `net.programmierecke.radiodroid2`...
Spawned `net.programmierecke.radiodroid2`. Resuming main thread!
[Android Emulator 5554::net.programmierecke.radiodroid2 ]-> [*] OkHttp Request URL: http://101smoothjazz.com/favicon.ico
[*] OkHttp Request URL: https://stream.radio-unicc.de/images/Logo-Radio_UNiCC.png
[*] OkHttp Request URL: https://www.reyfm.de/icon.png
[*] OkHttp Request URL: https://dancewave.online/dw_logo.png
[*] OkHttp Request URL: https://www.wix.com/favicon.ico
[*] OkHttp Request URL: http://www.echoesofbluemars.org/images/favicon.ico
[*] OkHttp Request URL: https://mangoradio.de/wp-content/uploads/cropped-Logo-192x192.webp
[*] OkHttp Request URL: https://www.wix.com/favicon.ico
[*] OkHttp Request URL: https://www.reyfm.de/icon.png
```

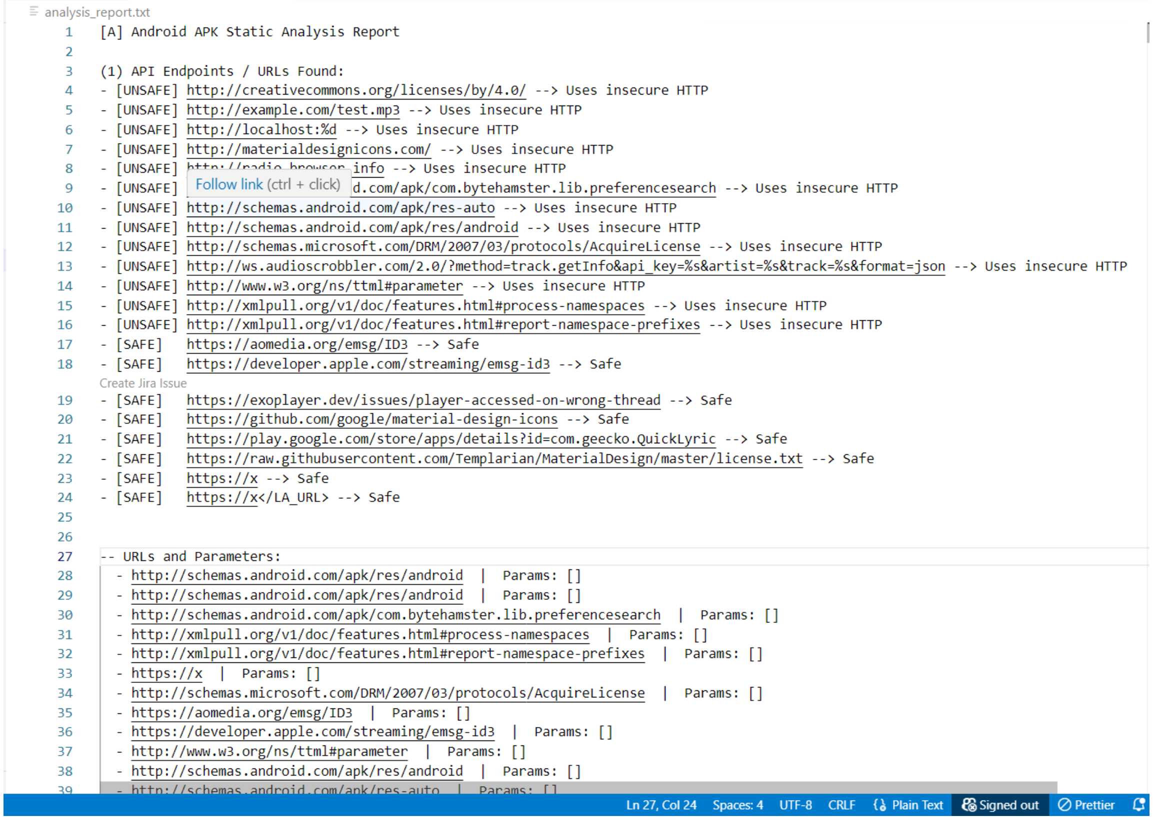
Step 5: Monitor Network using mitmproxy

- The script starts mitmproxy to capture network calls.
- Logs are saved in traffic_log.txt.



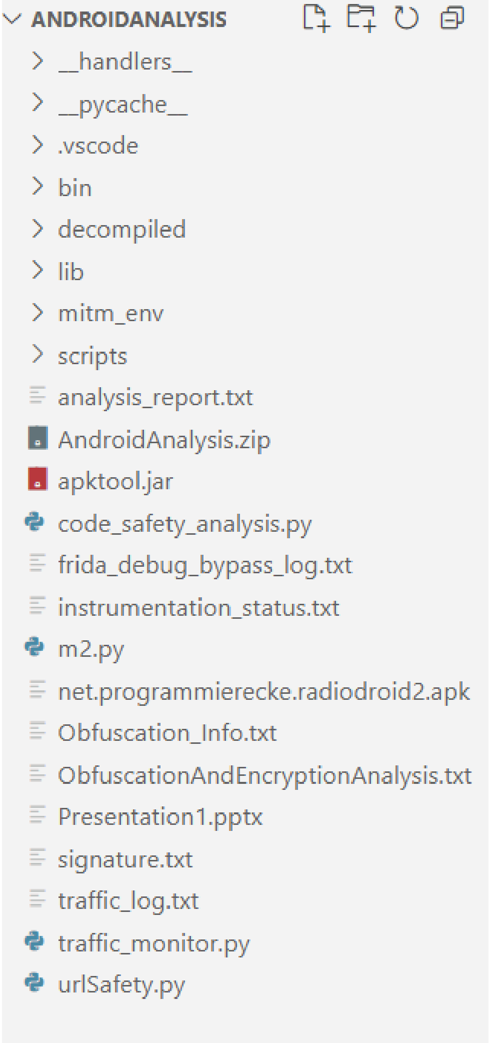
```
2025-06-07 23:33:12,151 - server disconnect radiomap.eu:443 ([2001:8d8:100f:f000::260]:443)
2025-06-07 23:33:12,154 - client connect
2025-06-07 23:33:12,156 - client disconnect
2025-06-07 23:33:12,158 - Client TLS handshake failed. The client does not trust the proxy's certificate for gfx.radiozet.pl (OpenSSL E
2025-06-07 23:33:12,159 - server disconnect fi1.api.radio-browser.info:443 ([2a01:4f9:c012:3620::1]:443)
2025-06-07 23:33:12,164 - client disconnect
2025-06-07 23:33:12,165 - server disconnect gfx.radiozet.pl:443 (193.187.66.152:443)
2025-06-07 23:33:12,361 - Client TLS handshake failed. The client does not trust the proxy's certificate for somafm.com (OpenSSL Error(
2025-06-07 23:33:12,364 - server connect de2.api.radio-browser.info:443 ([2a01:4f8:c2c:f004::1]:443)
2025-06-07 23:33:12,366 - client disconnect
2025-06-07 23:33:12,368 - server disconnect somafm.com:443 (198.24.44.214:443)
2025-06-07 23:33:12,624 - Client TLS handshake failed. The client does not trust the proxy's certificate for de2.api.radio-browser.info
2025-06-07 23:33:12,626 - client disconnect
2025-06-07 23:33:12,627 - server disconnect de2.api.radio-browser.info:443 ([2a01:4f8:c2c:f004::1]:443)
2025-06-07 23:33:12,639 - client connect
2025-06-07 23:33:12,657 - client connect
2025-06-07 23:33:12,827 - server connect de1.api.radio-browser.info:443 ([2a0a:4cc0:c0:27c1::1]:443)
2025-06-07 23:33:12,830 - server connect radiomap.eu:443 ([2001:8d8:100f:f000::260]:443)
2025-06-07 23:33:12,901 - client connect
2025-06-07 23:33:12,942 - client connect
2025-06-07 23:33:13,026 - Client TLS handshake failed. The client does not trust the proxy's certificate for de1.api.radio-browser.info
2025-06-07 23:33:13,027 - client disconnect
2025-06-07 23:33:13,028 - server disconnect de1.api.radio-browser.info:443 ([2a0a:4cc0:c0:27c1::1]:443)
2025-06-07 23:33:13,030 - client connect
2025-06-07 23:33:13,114 - server connect www.franceinter.fr:443 (13.36.124.174:443)
2025-06-07 23:33:13,152 - server connect somafm.com:443 (198.24.44.214:443)
2025-06-07 23:33:13,181 - Client TLS handshake failed. The client does not trust the proxy's certificate for radiomap.eu (OpenSSL Error
```

Overall Report Output (Report Generation Module)



```
[A] Android APK Static Analysis Report

(1) API Endpoints / URLs Found:
- [UNSAFE] http://creativecommons.org/licenses/by/4.0/ --> Uses insecure HTTP
- [UNSAFE] http://example.com/test.mp3 --> Uses insecure HTTP
- [UNSAFE] http://localhost:%d --> Uses insecure HTTP
- [UNSAFE] http://materialdesignicons.com/ --> Uses insecure HTTP
- [UNSAFE] http://radio-browser.info --> Uses insecure HTTP
- [UNSAFE]                    d.com/apk/com.bytehamster.lib.preferencesearch --> Uses insecure HTTP
- [UNSAFE] http://schemas.android.com/apk/res-auto --> Uses insecure HTTP
- [UNSAFE] http://schemas.android.com/apk/res/android --> Uses insecure HTTP
- [UNSAFE] http://schemas.microsoft.com/DRM/2007/03/protocols/AcquireLicense --> Uses insecure HTTP
- [UNSAFE] http://ws.audioscrobbler.com/2.0/?method=track.getInfo&api_key=%s&artist=%s&track=%s&format=json --> Uses insecure HTTP
- [UNSAFE] http://www.w3.org/ns/ttml#parameter --> Uses insecure HTTP
- [UNSAFE] http://xmlpull.org/v1/doc/features.html#process-namespaces --> Uses insecure HTTP
- [UNSAFE] http://xmlpull.org/v1/doc/features.html#report-namespace-prefixes --> Uses insecure HTTP
- [SAFE]   https://aomedia.org/emsg/ID3 --> Safe
- [SAFE]   https://developer.apple.com/streaming/emsg-id3 --> Safe
- [SAFE]   https://exoplayer.dev/issues/player-accessed-on-wrong-thread --> Safe
- [SAFE]   https://github.com/google/material-design-icons --> Safe
- [SAFE]   https://play.google.com/store/apps/details?id=com.geecko.QuickLyric --> Safe
- [SAFE]   https://raw.githubusercontent.com/Templarian/MaterialDesign/master/license.txt --> Safe
- [SAFE]   https://x --> Safe
- [SAFE]   https://x</LA_URL> --> Safe


-- URLs and Parameters:
  - http://schemas.android.com/apk/res/android  |  Params: []
  - http://schemas.android.com/apk/res/android  |  Params: []
  - http://schemas.android.com/apk/com.bytehamster.lib.preferencesearch  |  Params: []
  - http://xmlpull.org/v1/doc/features.html#process-namespaces  |  Params: []
  - http://xmlpull.org/v1/doc/features.html#report-namespace-prefixes  |  Params: []
  - https://x  |  Params: []
  - http://schemas.microsoft.com/DRM/2007/03/protocols/AcquireLicense  |  Params: []
  - https://aomedia.org/emsg/ID3  |  Params: []
  - https://developer.apple.com/streaming/emsg-id3  |  Params: []
  - http://www.w3.org/ns/ttml#parameter  |  Params: []
  - http://schemas.android.com/apk/res/android  |  Params: []
  - http://schemas.android.com/apk/res-auto  |  Params: []
```

9. Project Structure

 ANDROIDANALYSIS __handlers__ __pycache__ .vscode bin decompiled lib mitm_env scripts analysis_report.txt AndroidAnalysis.zip apktool.jar code_safety_analysis.py frida_debug_bypass_log.txt instrumentation_status.txt m2.py net.programmierecke.radiodroid2.apk Obfuscation_Info.txt ObfuscationAndEncryptionAnalysis.txt Presentation1.pptx signature.txt traffic_log.txt traffic_monitor.py urlSafety.py	1. m2.py – Main Program 2. traffic_monitor.py – used by mitmproxy to moniteor the traffic 3. urlSafety.py – identifies whether the url is safe or not 4. code_safety_analysis.py – analyze manifest and smali files for security patterns like [+] Debuggable is enabled [+] Components are Exported or Not [+] Native Library declared [+] Obfuscated class [+] Dynamic Code Loading [+] Native Library Loaded -

10. Project Documents

1. **AndroidAnalysis.zip** – it contains all the above mentioned python files, along with the jar files and hook files (.js)
2. **Android_APK_Security_Analysis_Report.pdf** –the document explaining the tool and its implementation details.

References

1. https://frida.re/docs/home/
2. https://mitmproxy.org/
3. https://developer.android.com/studio
4. https://github.com/iBotPeaches/Apktool
5. https://github.com/skylot/jadx