Unified Mobile Malware Analysis (UMMA)

METHODOLOGY

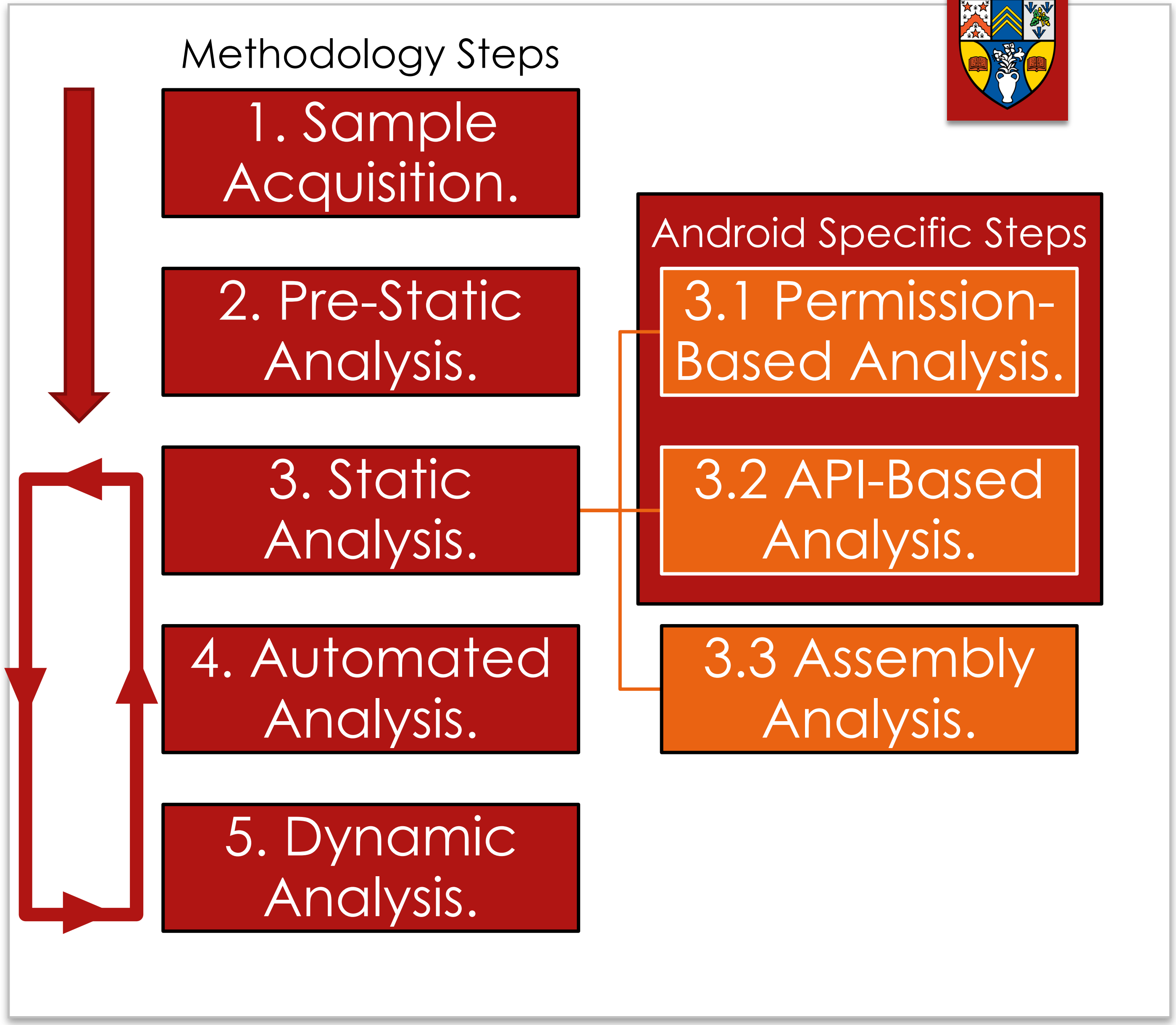
BY: MYKOLA (NICK) NESTERENKO

Disclaimer: Handle Malware at your own risk

Disclaimer: Handle Malware at your own Risk.

The information and content provided in this guide are for educational and informational purposes only. The author and presenter do not assume any responsibility for any misuse or damages resulting from the application or misapplication of the information and software presented. Always exercise caution and follow best practices when handling and analysing malware or other potentially harmful software. By accessing and using this information, you agree to indemnify and hold harmless the author and presenter from any liability, claims, or damages arising from the use or misuse of the provided information and software.

Unified Mobile Malware Analysis (UMMA) v0.3 Beta



Methodology Steps

1. Sample Acquisition.

2. Pre-Static Analysis.

3. Static Analysis.

4. Automated Analysis.

5. Dynamic Analysis.

Android Specific Steps

3.1 Permission-Based Analysis.

3.2 API-Based Analysis.

3.3 Assembly Analysis.

Disclaimer: Handle Malware at your own risk.

1. Sample Acquisition

Setting up a safe environment:

- Create an isolated virtual environment for analysis (for Apple Silicon Parallels users, use the provided malware analysis virtual machine).
- Configure network settings to prevent accidental spreading of malware.
- Ensure that the analysis environment does not contain sensitive data.

Transferring the Malware Sample to a Safe Environment:

- Use secure methods for transferring the malware sample (e.g., password protected .zip archives).
- Limit exposure to other devices during transfer.
- Avoid opening or executing the sample outside the analysis environment.

Sample Verification:

- Confirm the integrity of the malware sample by checking hashes and digital signatures.
- Document the sample's properties, such as file type, size, and origin.

Adopting Backup Routine and Best Practices:

- Maintain backups of the analysis environment to enable quick recovery in case of accidental infection.
- Follow best practices for handling and storing malware samples to prevent accidental release or exposure.

Disclaimer: Handle Malware at your own risk.

2. Pre-Static Analysis

- **Online Malware Analysis Platforms**:
  - In addition to VirusTotal, consider using other platforms like **Hybrid-Analysis**, **Any.Run** and specifically **Joe Sandbox**.
  - Leverage these services for more comprehensive results, as each may use different detection engines and databases and could assist in classification of the malware sample.
  - Obtain a broader understanding of the malware's behaviour, distribution, and potential impact.
- **Threat Intelligence Gathering**:
  - Research similar malware samples, campaigns, or threat actors to gain context on the sample under analysis.
  - Use platforms like **MITRE ATT&CK** and **AlienVault OTX** for threat intelligence information.
  - Correlate findings to inform subsequent stages of the analysis process.
- **String Search**:
  - Dumping strings of the sample may unveil information which could point to the malicious code segments or the lack of such.

Disclaimer: Handle Malware at your own risk.

3. Static Analysis

- **Code and Control Flow Analysis:**
  - Disassemble or decompile the mobile malware sample (APK, IPA, or other formats) to study its code structure, logic, and functionality.
  - Identify potential malicious code patterns, encryption routines, or obfuscation techniques specific to mobile platforms.
  - Use tools like **JADX**, **apktool**, or **Hopper** for in-depth examination and control flow visualisation on mobile malware.
- **Permission-Based Analysis:**
  - Examine the requested permissions within the mobile malware sample to identify potential abuse or privacy concerns.
  - Assess the necessity of each permission based on the sample's functionality and evaluate the risk associated with granting these permissions.
  - Utilise tools like MobSF, Androguard, or Android Studio's Manifest Viewer to investigate permissions within the mobile malware.
- **API Code Analysis:**
  - Inspect the mobile malware sample's use of APIs, focusing on those that access sensitive resources, device features, or services.
  - Analyse the API calls and their implementation to identify potential vulnerabilities, malicious intent, or non-compliant usage.
  - Employ tools like JADX, Frida, or jadx-gui to explore and evaluate API calls within the mobile malware sample.

Disclaimer: Handle Malware at your own risk.

4. Automated Analysis

Static Analysis with Automated Tools:

- Utilise automated static analysis tools like **MobSF**, Androwarn, or Quark-Engine to quickly scan the mobile malware sample and provide a high-level overview of its properties.
- Obtain insights on code structure, permissions, API usage, and potential security risks with minimal manual effort.
- Use the generated reports to inform further in-depth manual analysis, focusing on specific areas of interest or concern.

Automated Behaviour Analysis and Profiling:

- Employ tools like AppMon, DroidBox, or TaintDroid to automatically analyse and profile the mobile malware's behaviour during execution.
- Capture information on data leakage, sensitive API usage, and other potentially malicious activities or patterns.
- Compare the extracted behavioural profiles against known malware samples to identify similarities, shared code, or potential attribution.

Disclaimer: Handle Malware at your own risk.

5. Dynamic Analysis

- Controlled Execution Environment:
  - Set up a controlled execution environment using emulators, simulators, or dedicated test devices to safely execute the mobile. malware sample
  - Isolate the test environment from production networks and ensure proper monitoring and containment to prevent unintended. consequences
  - Employ tools like **Android Emulator**, Genymotion, or iOS Simulator to create a suitable environment for dynamic analysis.
  - For MacOS users, use tools firewall tools like **LuLu** and Little Snitch to ensure restricted malware network communications.
- Real-time Behaviour Monitoring:
  - Observe the mobile malware's behaviour during runtime, including file operations, network activities, system API calls, and interactions with device hardware or sensors
  - Utilise tools like **Frida**, strace, or Logcat to monitor the malware's actions and capture detailed logs for further analysis
  - Identify any deviations from expected behaviour, potential vulnerabilities, or malicious activities that were not evident during static analysis
- Reverse Engineering and Debugging:
  - Leverage reverse engineering and debugging tools, such as IDA Pro, **JADX**, or radare2, to gain a deeper understanding of the malware's functionality and execution flow
  - Set breakpoints, inspect memory, and manipulate execution to uncover hidden functionality, anti-analysis techniques, or encryption keys
  - Use the insights gained through dynamic analysis to inform further static analysis or develop countermeasures and detection signatures

Disclaimer: Handle Malware at your own risk.

android_analysis Script

- Script that combines several tools in one workflow:
  - Pre-Static Analysis: VirusTotal Lookup with **AndroPyTool**.
  - Static Analysis: Dumping information such as list of permission, strings, API-calls, etc. with **androguard** and producing JSON report with **AndroPyTool**.
  - Assembly Analysis: Automatically launching **jadx**, an Android specific decompiler, with Debugging Capabilities and creating **Frida** snippets.
  - Automated Analysis: Launching Mobile Security Framework (MobSF) UI in FireFox (The .apk needs to be uploaded into MobSF manually).
  - Dynamic Analysis: **MobSF** + **Frida** Dynamic Analysis within the launched MobSF window, manual **adb** configuration required.

- Usage:
  - --vtk: VirusTotal API Key. Required.
  - --f: File path to the .apk file you wish to analyse. Required.

Disclaimer: Handle Malware at your own risk.