Scope Methodology

& J. I. Weverink

16 november 2021

Inhoudsopgave

1 Inleiding

2 Doelstelling

3 Scope

For the research it was important to write a scope. The scope described what data had to be collected, what data was not collected and the reasoning behind these decisions.

For the research only Android applications were tested, the reason behind this was that the necessary tools that were provided only work on Android applications. This means that no IOS applications were tested. Another reason that iOS applications were not tested was that they could not be installed on an emulator on Windows operating systems, while android applications could be installed on Unix systems (also includes MAC OS) and Windows systems.

It was decided that doing a behavior analysis would help to understand how the application works. During this analysis only visual data of the application was collected. The research does not require the collection of any form of background activities performed by the application during the behavior analysis. These activities did not hold value for the behavior analysis.

It was decided that a network analysis was necessary to raise the probability of retrieving the information that the research required. During this analysis all sources that the applications connected to were collected. These sources could consist of IP-addresses and domain names. The research allowed for the gathering of information of any potential servers with the use of tools such as shodan.io. Connecting to these services however fell outside the scope of the research. This had the potential of raising unwanted attention of the malicious parties.

For properly analyzing an Android application it was decided that doing a code analysis was very important, the code had potential to had revealed a lot of important information about what the application was doing that otherwise had been very hard to find. It had been important for the code analysis to get ahold of the source code of the software. This was achieved by using a decompiling program if the source of the application was not available.

For further understanding of the application, it had been decided that a process analysis was required. During this analysis, the memory and the processes that had been activated in the background were monitored. The different dependencies that the application used, had also been monitored.

During the research there was a possibility that a connection between a malicious application and a hidden database or server were found. This created a chance to potentially gain access to the server or database. It was decided that the actions within the research were within the confounds of the law. For this reason, it was crucial for the research that the international laws were being upheld. One of these laws had prohibited the research from interfering or tampering with the server or database without the express consent of the other party. Because of this reason, it was decided that there would be made notes about the connection and not interact with it in any way.

4 Methodology

In order to start the project every team member had set up their testing environment. Every member had chosen to either work in a virtual machine or use their local machine to test the application. Using a local machine, every member must consider their own safety when working with malicious software. Publicly available tools were also properly setup on the local or virtual machine such as Android Studio, mitmproxy and Wireshark.

4.1 Finding the malicious applications

For the research it was decided that all the selected applications had come from the site Koodous and were confirmed to be malicious with the site VirusTotal. This had been done individually by all team members to find applicable APK files for the research. Some checks were performed manually to see if the applications contained any pointers that indicated that they contained malware. The application was checked on its availability for download on the Google Play store. The potentially malicious application was compared with the variant from the Google Play store. A few examples of these pointers were differing sizes or if both had the exact same package name as the Google Play store variant while differing in permissions or certificates. If it had one of those pointers, then it likely was an indicator that there was something extra hidden in the code.

The application was retrieved from both Koodous and the Google Play Store. The APK from Google Play was downloaded using the apkcombo.com tool. Both APK files were uploaded to VirusTotal to check for any differences in permissions between the two versions of the application. Anything VirusTotal had marked anything as malicious, was noted. This was up to the discretion of every individual team member to decide if they decided to work on the chosen application for the duration of the project.

4.2 Behavior analysis

The research required that five applications were examined. This had been done on a virtual computer with android studio installed or locally. During the installation of the application the network traffic had been captured. This had been done using Wireshark and mitmproxy. The network traffic was not relevant for behavior analysis but was captured as part of the network analysis.

The behavior research had been predominantly about using the application and noting anything out of the ordinary. This had also been compared step by step to the original (non-malicious) application. The differences between the two are noted for further investigative purposes. The ordinary behavior had been investigated further during network and code analysis.

4.3 Network analysis

During the behavior analysis the network traffic from the application was captured using Wireshark and mitmproxy. In the network analysis this captured network traffic had been analyzed using a combination of Wireshark, mitmproxy and shodan.io. During the analysis anything that stood out like any IP addresses and or domain names were written down.

Shodan.io was used for analyzing the IP addresses and domain names found with Wireshark and mitmproxy. Wireshark and mitmproxy were able to reveal all the network traffic sent and received by the application. Shodan.io revealed the other functions of the servers and whether there are similar servers, perhaps from the same hosting provider.

4.4 Code analysis

For the code analysis, JadX and Dex2Jar were used to decompile the application. This created the opportunity to read the code. An attempt was made to analyze the code. Analyzing the code was very difficult due to obfuscation. Where possible it was documented whether the code was malicious or not.

4.5 Process analysis

The Processes were analyzed to gain the insights that the research needs. The application had been started in the emulator of Android Studio. During the startup and use of the application, the Memory profiler had been used to monitor the memory usage of the application. All memory usage was documented and all peculiar behavior had been investigated further, to understand what had caused the suspicious behavior. Optionally, a heap dump was done during execution, which led to interesting findings.

4.6 Countermeasures and detection

To finalize the analysis, a YARA ruleset had been created for the application. All findings were documented into a report which was delivered to the team leader. These steps were performed by all the members. If an application needed additional steps this was declared at the start of the application chapter.

Referenties