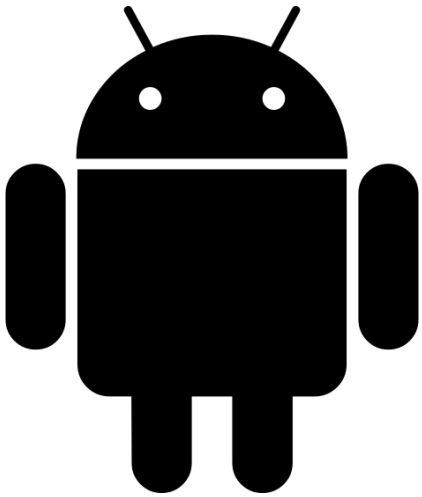
2018-19 FYP Detailed Project Plan

Automated Android Malware Detection



Supervisor: Dr K.P. Chow
Student: Ho Man Hon (3035244356)

Table of content

I. Project Background

According to the statistics from International Data Corporation [1], Android is the most popular mobile operation system, which have chosen by 84.8% of mobile device users. To make higher the chance of success, cybercriminals tend to attack the larger user group. Hence, we can see the daily amount malware of Android is still very large, for example 8,400 new malware instances every day in 2016 [2]. And this trend keeps going faster and faster in the future.

However, security vulnerabilities in Android are very hard to fix. Even Google might release a security update, the other android OS developer in third parties might fail to follow. Also, in the Android malware dataset [3], we can see more and more malware developed with anti-analysis techniques to prevent the malware detection from dynamic checking. Google Play Store has also failed to prevent all malware from uploading into their store, as cybercriminals would also create malware with machine learning behavior [4]. As the result, the failure of automated malware analysis would make the situations worse.

To deal with the dramatically increasing trend of Android malware, we need an automated detection which can keep track on those anti-analysis techniques by different analysis.

II. Objective

Lots of anti-analysis techniques are currently used by the cybercriminals. Due to the limit of the Android OS and the ineffective security update, it is not a good idea to trust your OS provider can keep track on the zero-day malware and response to them by releasing a fix in very short time.
Not only the periodical vulnerability check of the applications, we would need a more accurate automated malware analysis tool be a reasonable solution to the evolving malware around the world.

The Project Goal – To provide the following features in the malware analysis tool:

- Web platform that can receive the Android application with Dynamic malware analysis.
- System logger that can record all activities after running the test application.
- Emulator anti-detection which is able to capture new anti-sandbox techniques.

III. Scope

In this project, we would focus on:

- Android application (APK) analysis
- Implementation of automated malware analysis tools
- Patterns of anti-sandbox techniques

IV. Methodology

This project will be completed after 3 phases:

First Phase – Environment Setup

We will setup the Cuckoo sandbox (with CuckooDroid [5]) environment in order to analyze the Android malware. To achieve this stage we need to install the cuckoo framework, android emulator from Android Studio and then run an application with an analysis report. Sample applications would be obtained from Android Malware Dataset, which is a well-studied malware dataset.

Second Phase – Anti-sandbox Techniques analysis

Here we will start to work on the main theme. In the current malware analysis tool, Droidmon is the logger responsible for recording operation system behavior's changes after running the application [7]. Since current detection only works with predefined anti-sandbox techniques, we would plan to design a more flexible Anti-sandbox Techniques analysis, with software tools in Python to encounter them, using the information given by the Droidmon logger, and then study the pattern on those Anti-sandbox Techniques.

The follow diagram has shown almost all the components that we need to setup for android malware analysis, and what features we are going to add in this phase:

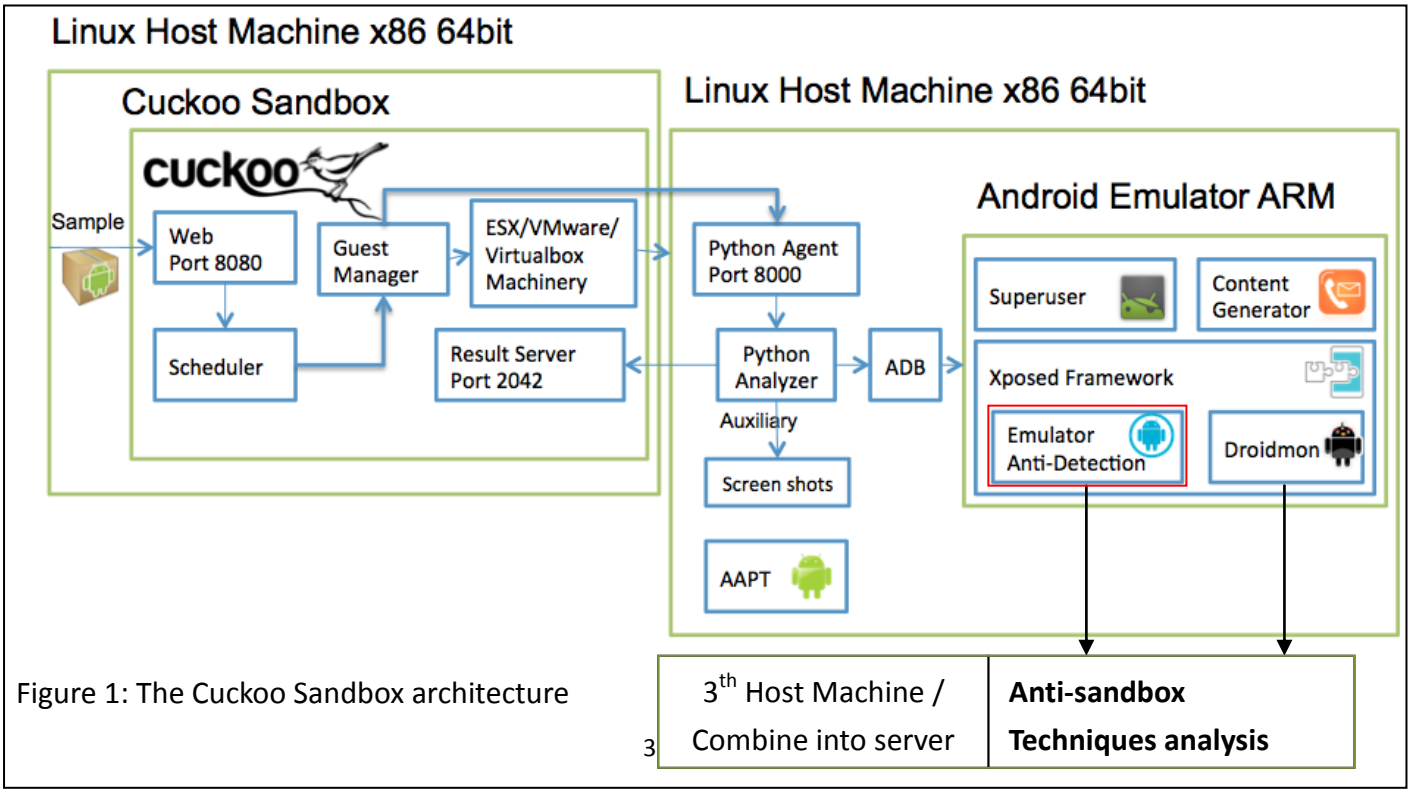


Figure 1: The Cuckoo Sandbox architecture

Final Phase – Handling new suspicious anti-sandbox malware

We will implement the program to report if there are some suspicious anti-analysis activities of the Android application. (For example, malware checks for some system information which is abnormal for 'normal' application) If the detection is malware then nothing we have to do. If the analysis report tells that it is not likely a malware but it contains anti-analysis activities, we might need to have further action. Hence we should be notified and performing static code analysis to see if it has malware behavior in the skipped code.

V. Risks & Challenges

The first risk would be the project difficulty. Since the whole cuckoo framework is very large and Android operation system is another new concept that we have not learnt in deep before, we might need to invest more than estimated time to investigate how to work on the implementation and what is the mechanism inside the framework. To tackle this risk, we will use most of the time in October to get familiar with the malware analysis tool.

The second risk would be the hardware. For each VM, the minimum requirement would be 2GB RAM and 40GB. Since we do not have a powerful machine to run 3 VMs at the same time, we might need to purchase cloud VMs in order to run the whole system. However, malware analysis in public cloud might not be available since some of the cloud provider would not enable malware to run in their public cloud. Hence, we might need to consider if we can use less VMs after we have started to work on the project.

The final risk would occur in the second phase. Anti-sandbox techniques would have many possible ways to achieve the goal, such as checking system information or system environment variable, performing test call to somewhere, etc. To classify how an application performs its anti-sandbox techniques is extremely difficult as sometimes they will also combine with other anti-analysis techniques such as time delay, dynamic loading, renaming, string encryption, etc. We might need to study other anti-analysis techniques in order to figure out the pattern of anti-sandbox techniques.

Overall the project difficulty is very high and the Anti-sandbox patterns observed might have some uncertainty, but we think it would be a useful malware analysis research. We will be patient and use more time to solve those programming challenges and hardware limitations by applying some latest technical solutions.

VI. Project Plan

Date	Schedule
30/09/18	Deliverables of Phase 1: Inception - Complete the project plan - Build a project website
01/10/18 to 22/10/18	Study the theory & related knowledge: - Android emulator (VM setup) - Malware detection tool (Cuckoo with CuckooDroid) - Anti-sandbox techniques (Xposed Framework)
23/10/18 to 07/01/19	- Working on First Phase implementation
07/01/19 to 11/01/19	- First Presentation
20/01/19	Deliverables of Phase 2: Elaboration - Conduct preliminary implementation - Complete detailed interim report
10/01/19 to 25/02/19	- Working on Second Phase implementation
26/02/19 to 12/04/19	- Working on Final Phase implementation - Prepare for final presentation and final report
14/04/19	Deliverables of Phase 3: Construction - Conduct finalized tested implementation - Complete the final report
15/04/19 to 19/04/19	- Final presentation
29/04/19	- Project exhibition

VII. References

[1]: IDC (2018), https://www.idc.com/promo/smartphone-market-share/os

[2]: GD Data (2017), https://www.gdatasoftware.com/blog/2017/04/29712-8-400-new-android-malware-samples-every-day

[3]: DIMVA 2017, http://amd.arguslab.org/evolution

[4]: WIRED (2017), https://www.wired.com/story/google-play-store-malware/

[5]: CuckooDroid, https://github.com/idanr1986/cuckoo-droid

[6]: Sophos, https://news.sophos.com/en-us/2017/04/13/android-malware-anti-emulation-techniques/

[7] Droidmon, https://github.com/idanr1986/droidmon

[8] CopperDroid: Automatic Reconstruction of Android Malware Behaviors (2015), http://dev.www.isocdev.org/sites/default/files/02_4_1.pdf

[9] ReDroid, https://github.com/yzygitzh/ReDroid