CARLOS III UNIVERSITY OF MADRID

Universidad Carlos III de Madrid

MASTER IN CYBERSECURITY

MASTER THESIS

TARGETED EXERCISER FOR ANDROID MALWARE AND GRAYWARE

Author: Mario Herreros Díaz

Mentors: Juan E. Tapiador, Guillermo Suarez-Tangil

September 9th, 2016

TARGETED EXERCISER FOR ANDROID MALWARE AND GRAYWARE

Mario Herreros Díaz [a], Juan E. Tapiador [b], Guillermo Suárez-Tangil [b]

[a] *Student of the Master in Cybersecurity, Universidad Carlos III de Madrid, Leganés, Spain*

[b] *COSEC Lab, Universidad Carlos III de Madrid, Leganés, Spain*

Abstract

Nowadays each person has one or more mobile device (smartphones, tablets, wearables, etc.) with characteristics and performance similar to those of a personal computer. Many people use these devices to check their email or make bank transfers. For this, these devices store a lot of sensitive information that could be very attractive to an attacker.

Mobile devices share same security issues can be found in a conventional computer as malware hidden inside apps or users that don't use nor protect them correctly. Malware has evolved and is able to bypass protection systems and works only under certain circumstances to avoid detection. Some kind of malware is activated depending of the network connection, the device location, the apps installed, the calls received or other types of events. The purpose of this Master Thesis is to study the behavior of malware on mobile devices depending of the device context, focusing on Android operating system. It has developed a system that allows dynamic and automatic generation of many different contexts to study the behavior of Android malware in each of the different scenarios.

It has been designed a technique based on developing of a new language which defines each of the scenarios and events to execute over the device to analyze the malware, in order to study their behavior depending on the characteristics of each context. In this way, it can detect the context features that make triggering the malicious malware or grayware actions. Specifically, this Thesis focuses on analyzing all possible Android events to define a language and developing a system available to understand it and generate automatic executions to launch these events. With this system it is possible detect malware and grayware that a conventional static and dynamic analysis could not detect.

Keywords: Malware, Grayware, Android, Context, Malware detection, Dynamic analysis, Mobile device, Malware behavior

---

Email addresses: 10027558@alumnos.uc3m.es (Mario Herreros Díaz), jestevez@inf.uc3m.es (Juan E. Tapiador), guillermo.suarez.tangil@uc3m.es (Guillermo Suárez-Tangil)

Code Repository: https://bitbucket.org/gtangil/targetdroid [1]

I. Introduction

Nowadays mobile devices are very present within society and every people uses them in their personal and working life. The most popular smartphone OS is Android, with a market share around 82% – 84%. For this reason Android is the platform most targeted by malware and grayware [2]. Figures 1 and 2 show the evolution of the market share in the last years:

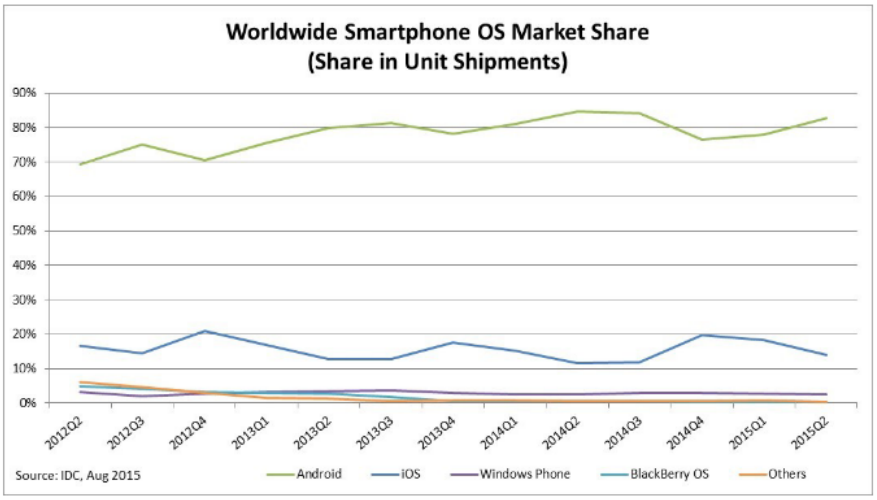

Figure 1: Worldwide Smartphobe OS Market Share Graph

Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Source: IDC, Aug 2015

Figure 2: Worldwide Smartphobe OS Market Share Table

The amount of Android malware families has increased very significantly in recent years. Android malware currently includes 295 different families [3]. Figure 3 shows this increase in the last years:

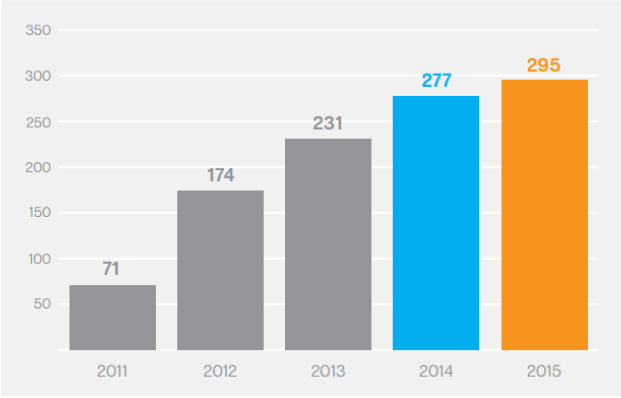

Figure 3: Android Malware Families Evolution

An important fact is the variability of Android malware. There are thousands variants of malware whose target functionality is similar. During the last years, the variability of malware has rose considerably, reaching in 2015 the number of 13,783 types [3]. Figure 4 represents the increment of Android malware types in the last years:

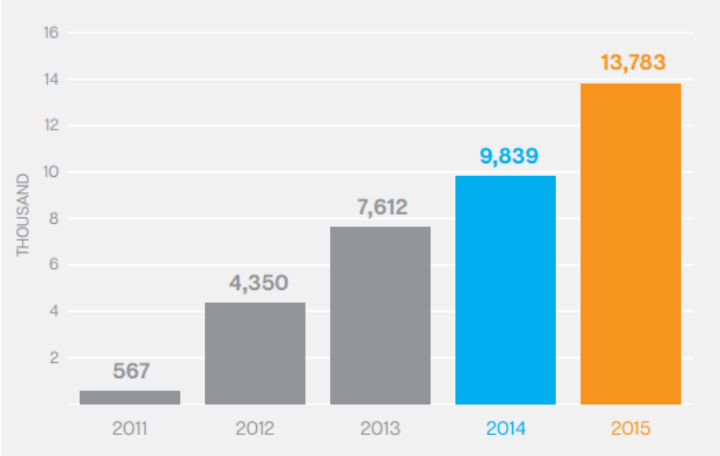

Figure 4: Android Malware Types Evolution

The powerful capabilities of mobile devices in networking, computing and sensing has increased the complexity, power and variability of malware. This allows the malware to evolve and adopt very specific functionality. There is a type of malware that focuses on a specific victim, known as targeted malware. This type of malware focuses on user behavior using his smartphone and other factors relating to him, such as his location, the information handled or applications installed in the device.

The main advantage of targeted malware against security systems is their difficult detection. This is due to the fact that only they execute his malicious code if a concrete context is fulfilled. This situation causes the need to use a lot of resources to detect this type of malware.

In the application stores a large number of applications that must be analyzed before publication are received each day. This makes malware analysis insufficiently comprehensive to detect targeted malware. This problem can be alleviated by the use of dynamic analysis techniques that allow the generation of different contexts and behaviors in an agile and automatic way.

A. Motivation

Deficiencies in malware detection systems and the strong growth of the power and complexity of malware have caused the existence of a major security threat in mobile devices. The analysis of targeted malware is a costly task, so the development of a system that will facilitate and automate the analysis can alleviate this problem existing in mobile security nowadays.

Focusing on the Android system, which is one of the most affected OS by mobile malware, it has developed a system able to define and execute different behaviors and contexts, in an effective manner using a virtual sandbox.

B. Main contributions

This work provides the following contributions:

1) An analysis and study of the Android system and architecture to inject different kind of events and contexts.
2) An analysis and study of targeted malware, focusing on key behaviors and contexts used to trigger the malware.
3) Definition of a new language to easily define a set of behaviors based on different scenarios and contexts.
4) An analysis and development of a new system, based on Targetdroid, capable of computing the new defined language and generate a dynamic analysis based on the contexts and behaviors.

C. Structure of the document

The remaining of this document is structured as follows. First, Section II presents the related work regarding Android targeted malware and Targetdroid. Section III explains the system developed, its architecture and the analysis carried out focusing on the user behavioral events. Section IV presents two case studies using the language defined and the system implemented with a configurable and targeted malware. Finally, Section V presents a series of recommendations and discusses about the importance of security measures in Android malware and the capabilities and deficiencies of the system, and Section VI concludes this work.

Additionally, three appendices are provided. Appendix A shows the User Behavior Language Schema, Appendix B lists all the events defined in the language and their possible values, and Appendix C shows the research planning and estimated Budget of this work.

II. Background and related work

This Thesis is directly motivated by work described in the paper *Detecting Targeted Smartphone Malware with Behavior-Triggering Stochastic Models* [4]. This paper focus on the need to detect targeted malware in smartphones. For this, it developed a system called Targetdroid that transforms different inputs in a dynamic analysis using multiple behavioral models. Targetdroid uses models that perform different emulations of user behaviors in order to detect which context activates the targeted malware or grayware. This system was based on stochastic models (Markov chains).

The main problem of Targetdroid that we attempt to solve in this Thesis is how to reproduce an appropriate set of conditions that will trigger the malicious behavior. In Targetdroid it is necessary to execute all possible states and combinations in the worst case, although mining the behavior of apps in different user-specific contexts and usage scenarios alleviates the problem.

The goal of this Thesis focuses on evolving this system to accept flexible inputs so that it achieves a better definition of the execution context and the user behavior.

Other authors have also studied the problem of detecting targeted malware, such as PyTrigger [5], focusing on Personal Computers (PC), or CopperDroid that generates different behavioral profiles automatically to analyze malware [6].

PyTrigger is a dynamic malware analysis system that automatically exercises a malware binary extracting its behavioral profile even when specific user activity or input is required. To accomplish this, it developed a novel user activity record and playback framework and a new behavior extraction approach. This system uses user activity records to generate different scenarios and events.

CopperDroid reconstructs OS and high level Android specific behaviors observing and dissecting system calls and, therefore, is resistant to the multitude of alterations the Android runtime is subjected to over its life-cycle. This system automatically and accurately reconstructs events of interest that describe, not only well-known process-OS interactions (e.g., file and process creation), but also complex intra- and inter-process communications (e.g., SMS reception), whose semantics are typically contextualized through complex Android objects.

All these systems are used as basis for the thesis explained in this work and the system developed uses concepts found in these related works.

III. Extending TargetDroid

This section describes the system developed that extends the functionality of Targetdroid accepting the new model defined based on the Behavioural User Language.

A. Behavioural User Language

We have designed a language that allows the definition of user behavior using a hierarchy of scenarios, contexts and events. The language is based on JSON text format and it is defined by the JSON scheme defined in Appendix B. This language will be used for generating user behavior files, used as input and interpreted by the system developed to generate different dynamic analysis on a cloud (for this case an Android emulator).

The language is structured in the following artifacts:

- **Scenario**: it is a set of contexts.
- **Context**: it is a set of events.
- **Event**: atomic action performed on the system.

Scenario

This artifact represents all possible wanted scenarios to simulate a specific user behavior. The definition of scenarios allows performing analysis in which you want to maintain context configurations.

Context

This artifact describes all possible events that can be performed during the user behavior simulation determined by a moment in time.

There are three types of different contexts:

- *Emulator configuration context (t = -1)*
  It defines all environmental properties that will be used by subsequent contexts, that is, the characteristics of the emulator to be used as a sandbox for the dynamic analysis. These events will be executed during the preparation of the virtual machine (emulator) in which the app will be analyzed. We refer to this preparation time as t = -1.
- *OS configuration context (t = 0)*
  This context assigns the initial state of the initialized system. In other words, it configures the emulated Android system at the OS level. They are the events that will be injected at time t = 0, establishing an initial state in the Android OS.
- *Execution context (t > 0)*
  It represents each of the events that correspond to user manipulation with the device, such as sending a message, receive a call, run an app or determinate a geolocation. They are all the events that will be injected in times t > 0.

Event

This artifact defines each of the possible actions that can be performed on the system to simulate the user behavior. An event is an atomic unit in this system and includes a very different set of artifacts, such as hardware configurations or Telnet calls to emulator that allow simulate a phone call. A detailed list can be found in Appendix B.

B. Architecture

This section describes the architecture of the system developed. It will explain the components and their functionality inside the system. Figure 5 describes an architecture diagram of the system developed:

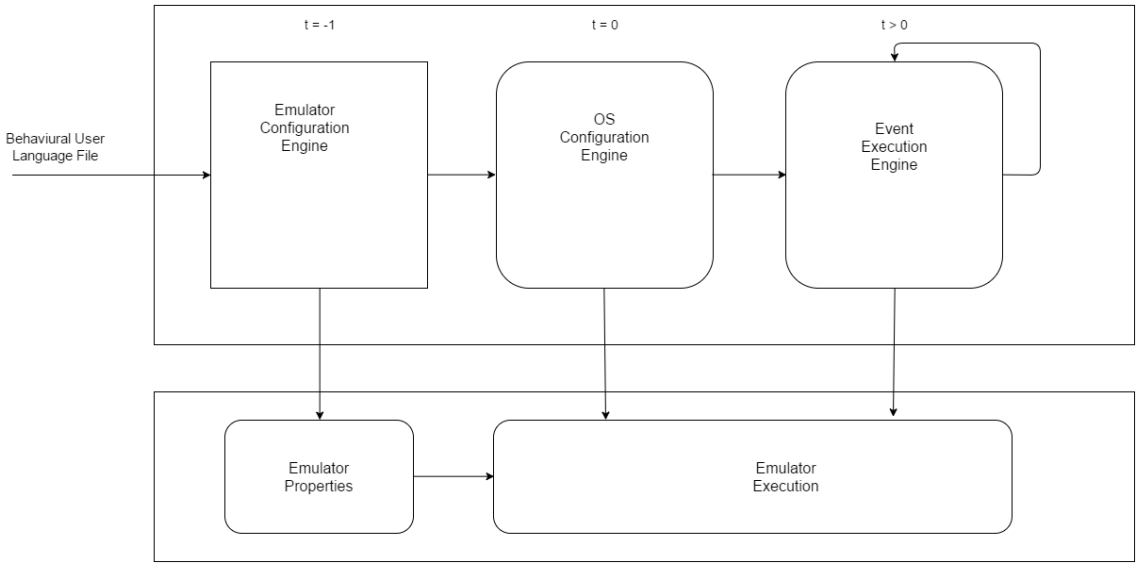

Figure 5: System Architecture

The system is split into two principal components, each of which contains several subcomponents:

- **TagetDroid**
  - *Emulator Configuration Engine:*
    This component generates the emulator configuration before running.
  - *OS Configuration Engine:*
    This component generates the emulator configuration after running.
  - *Event Execution Engine:*
    This component generates all active user behavior events.

- **Emulator**
  - *Emulator Properties:*
    This component represents all the configuration properties to define the emulator used such as cloud before running.
  - This component represents the emulator in running state, ready to listen commands from the Targetdroid system to execute events.

Targetdroid

The main components of Targetdroid are composed of different functional modules:

- **Emulator Configuration Engine (t = -1):**
  There are two principal modules in this component:
  - *Android emulator properties file:*
    This module parses the emulator properties defined in [7].
  - *Command line options*
    There is a set of additional options to create and run the emulator by command line, such as defining the sdk version, networks options, etc.

- **OS Configuration Engine (t = 0):**
  It contains two modules:
  - *Install app module:*
    This module allows install different Android apps using adb [8].
  - *OS configuration event injection:*
    This module can send events to Android emulator to configure the initial state of the emulator. It can allow configure the next options:
    - Timezone
    - Power properties

- **Event Execution Engine (t > 0)**
  It is composed by the following modules:
  - *Intents:*
    This module parses a set of adb commands related to execute intents to generate specific events in the emulator [8]:
    - Start intent
    - Start service intent
    - Broadcast intent
  - *ADB command to event:*
    This module parses a set of adb commands to generate a common user event in the emulator [6]:
    - Connect wifi:
    - Lock screen:
    - Unlock screen:
    - Volume up
    - Go home:
    - Take screenshot:
    - Start clock app
    - Stop clock app
    - Start wifi
    - Check wifi status
    - Enable wifi
    - Disable wifi
    - Enable mobile data
    - Disable mobile data
  - *Telnet command to event:*
    This module uses the Telnet protocol to send commands to emulator and generate a set of events in the cloud. The events related to telnet commands are the following [10]:
    - GSM events
    - Power events
    - Call events
    - SMS events
    - GEO events
  - *Monkey tester:*
    This module allows run monkey scripts, defined previously, such as event to generate UI events during the analysis. This module uses the android tool adb to generate these kind of events.

Emulator

This system has been developed on top of the Android SDK mobile device emulator. It allows us to generate different virtual mobile devices with many configurations and interact with them using different tools, such as adb, telnet, avd manager or logcat.

The main modules used in this system are:

- **Properties.ini file:**
  Every AVD (Android Virtual Device) created has an unique properties file. This file describes the properties of a given virtual device configuration file. It allows configure a set of characteristics for the virtual mobile device created.

- **Command line options**
  When it runs an AVD by command line, there are different options to configure the emulator, such as network or system options.

- **AVD Manager**
  It is a tool to create and configure Android Virtual Device. This tool is used to generate every cloud with the properties defined in each scenario.

- **ADB**
  Android Debug Bridge is a versatile command line tool that lets you communicate with an emulator instance or connected Android-powered device. It is a client-server program that includes three components:

  - A client, which sends commands. The client runs on your development machine. You can invoke a client from a shell by issuing an adb command. Other Android tools such as DDMS also create adb clients.
  - A daemon, which runs commands on a device. The daemon runs as a background process on each emulator or device instance.
  - A server, which manages communication between the client and the daemon. The server runs as a background process on your development machine.

- **Telnet**
  Telnet is an application layer protocol used on the Internet or local area networks to provide a bidirectional interactive text-oriented communication facility using a virtual terminal connection.
  It is possible connect to android emulator using this protocol and send commands to control the emulator.

IV. Case Studies

This section presents two case studies illustrating how the User Behavioral Language and the system developed can contribute to revealing malware with targeted activation mechanisms. The malware used in these case studies are the same from *Detecting Targeted Smartphone Malware with Behavior-Triggering Stochastic Models* [4] but using the new system defined in this work.

It has used an open source malware called *Androrat* (Android Remote Access Tool or RAT) incorporating different specific triggering conditions [11].

Case 1: Dormant Malware/Grayware

Targeted malware is programmed to remain dormant until a specific behavior or system conditions happens. In this case, it has instrumented *Androrat* to activate the RAT component when the device is in a certain location with a battery status upper than 65%, the Wi-Fi status must be connected and the device must have back camera.

For testing purposes, we defined a new User Behavior Language input file describing these conditions. For example, in this case we are not sure of the location values, and we doubt among three locations. It is necessary define three geolocation events in t > 0 choosing the possible values.

An example of file for this case showed next:

```
[
    {
        "name": "case_studie_1",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_1",
                    "sdk_version": "android-10",
                    "port": 5554
                },
                "ini_properties": {
                    "camera_back": true
                }
            },
            {
                "id": 0,
                "install_app": "my.app.client",
                "telnet": {
                    "power": {
                        "capacity": "70",
                        "present": true,
                        "status": "discharging"
                    }
                }
            },
            {
                "id": 1,
                "adb": [
                    {
                        "action": "start_wifi"
                    }
                ],
                "telnet": {
                    "geo": {
```

```
                        "fix": {
                            "latitude": "28.82781",
                            "longitude": "50.89114"
                        }
                    }
                }
            },
            {
                "id": 2,
                "telnet": {
                    "geo": {
                        "fix": {
                            "latitude": "22.82781",
                            "longitude": "52.89114"
                        }
                    }
                }
            },
            {
                "id": 3,
                "telnet": {
                    "geo": {
                        "fix": {
                            "latitude": "25.82781",
                            "longitude": "55.89114"
                        }
                    }
                }
            },
        ]
    }
]
```

For this case, it has defined a unique scenario composed by the next contexts:

- Emulator Configuration Context (time = -1):
  In this context it has defined the emulator properties to use as cloud and the ini property **camera_back: true**, because one of the conditions is the presence of the back camera in the device:

```
                {
                    "id": -1,
                    "emulator": {
                        "name": "case_studie_1",
                        "sdk_version": "android-10",
                        "port": 5554
                    },
                    "ini_properties": {
                        "camera_back": true
                    }
                },
```

- OS Configuration Context (time = 0):
  In this context it has defined the app with the targeted malware in the property **install_app: my.app.client** and the battery status upper than 65%:

```
                {
                    "id": 0,
                    "install_app": "my.app.client",
                    "telnet": {
                        "power": {
                            "capacity": "70",
                            "present": true,
                            "status": "discharging"
                        }
                    }
                },
```

- Execution context (time > 0):
  In this context it has defined the wifi conection in the time 1 and the geolocations events in the times 1, 2 and 3 to check the moment when the targeted malware is activated:

```
                {
                    "id": 1,
                    "adb": [
                        {
                            "action": "start_wifi"
                        }
                    ],
                    "telnet": {
                        "geo": {
                            "fix": {
                                "latitude": "28.82781",
                                "longitude": "50.89114"
                            }
                        }
                    }
                },
                {
                    "id": 2,
                    "telnet": {
                        "geo": {
                            "fix": {
                                "latitude": "22.82781",
                                "longitude": "52.89114"
                            }
                        }
                    }
                },
                {
                    "id": 3,
                    "telnet": {
                        "geo": {
                            "fix": {
                                "latitude": "25.82781",
                                "longitude": "55.89114"
```

```
                                }
                            }
                        }
                    },
```

This example reinforces the versatility of the language developed to generate different conditions and combine them.

Case 2: Anti-analysis Malware

Malware usually uses different techniques to bypass system analysis to detect malicious code. These systems use virtual sandboxes to perform the analysis rather than physical devices due to economic and efficiency factors. These sandboxes have a particular hardware configurations easy to detect and avoid for malware, such as the battery status are always the same or the network configurations are not real.

In the Android emulator, there are default hardware configurations. Such configurations are checked by the malware before executing to bypass the antimalware systems. Some default hardware values are represented in Table 1 [4]:

HW feature	Default value
IMEI	000000000000000
IMSI	012345678912345
SIM	012345678912345
Phone Number	1-555-521-PORT (5554)
Model Number	Sdk
Network	Android
Battery Status	AC on Charging 50%
IP Address	10.0.2.X

Table 1: Default Hardware Configuration for Android Emulator

For this case, we modified *Androrat* to actívate de RAT component in the next cases:

- Battery status different from 50% and AC status is off.
- IMEI has the value 123456789.
- IP Addres is 192.168.111.224

We defined a new Behavioral User Language focusing in this case in the scenario level. The scenario level allows generate different sandboxes with distinct hardware configurations.

An example of file for this case is the next:

```
[
    {
        "name": "case_studie_2_battery",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
```

```
                    "sdk_version": "android-10",
                    "port": 5554
                },
            },
            {
                "id": 0,
                "install_app": "my.app.client",
                "telnet": {
                    "power": {
                        "capacity": "70",
                        "ac": "off"
                    }
                }
            }
        ]
    },
    {
        "name": "case_studie_2_IMEI",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
                    "sdk_version": "android-10",
                    "port": 5554
                },
                "imei": "123456789"
            },
            {
                "id": 0,
                "install_app": "my.app.client",
            }
        ]
    },
    {
        "name": "case_studie_2_IP_ADDRESS",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
                    "sdk_version": "android-10",
                    "port": 5554
                },
                "network": {
                    "IP": "192.168.111.224"
                },
            },
            {
                "id": 0,
                "install_app": "my.app.client",
            }
        ]
    }
]
```

- The first scenario named as *case_studie_2_battery* defines in the time -1 the emulator properties and the time 0 the battery status with capacity from 70% and AC is off and install the app with the malware:

```
{
        "name": "case_studie_2_battery",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
                    "sdk_version": "android-10",
                    "port": 5554
                      },
                  },
                  {
                      "id": 0,
                      "install_app": "my.app.client",
                      "telnet": {
                          "power": {
                              "capacity": "70",
                              "ac": "off"
                          }
                      }
                  }
              ]
         },
```

- The second scenario named as *case_studie_2_IMEI* defines in the time -1 the emulator properties and the IMEI value to 123456789 and in the time 0 the app installation.

```
    {
        "name": "case_studie_2_IMEI",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
                    "sdk_version": "android-10",
                    "port": 5554
                      },
                      "imei": "123456789"
                  },
                  {
                      "id": 0,
                      "install_app": "my.app.client",
                  }
              ]
         },
```

- The third scenario named as *case_studie_2_IP_ADDRESS* defines in the time -1 the emulator properties and the IP address to 192.168.111.224 and in the time 0 the app installation:

```
{
        "name": "case_studie_2_IP_ADDRESS",
        "time": [
            {
                "id": -1,
                "emulator": {
                    "name": "case_studie_2",
                    "sdk_version": "android-10",
                    "port": 5554
                      },
                      "network": {
                          "IP": "192.168.111.224"
                      },
                  },
                  {
                      "id": 0,
                      "install_app": "my.app.client",
                  }
              ]
          }
```

With the scenarios, it is possible configurate different sandboxes in a same execution. Combining the scenarios with the different context events it is possible generate all types of user behaviors.

V. Discussion

There are previous systems that rely on user behavior to detect malware, for example the first version of Targetdroid [4] or pBMDS [12]. These systems are based on a probabilistic approach by correlating user inputs with system calls to detect anomalous activities in mobile devices. The problem in these systems is the lack of specification to generate user behaviors, since it could be necessary reproduce all the possibilities to reach some behavior.

In this thesis, we have analyzed many artifacts that can be used as malware triggers, such as callings, geolocations, hardware, and network properties. We have defined a new scenario definition method based on a language, so that the level of specification has incremented. The analyst can choose the sequence of events between the emulated user and the emulated device on a timeline. The language allows sharing contexts between different sequences to inject variable events over static contexts; that is, the analyst can change the hardware configuration in every scenario and inject the same user context events to focus if the malware is related to the hardware or vice versa.

This system offers a more specific method to analyze malware than the previous systems and major configuration level. But there is a problem yet: the UI/UX events. In this thesis, we have developed a monkey module that generates user interface events based on an input script. The problem is that the number of event combinations could be huge, since every app has a lot of buttons, options, links and functionalities. Combining these all elements with other type of events can produce an extensive analysis only for a single application. The target malware that

use user interface elements to execute their malicious functionality have more advantage than the antimalware systems.

The principal problem is the number of apps that appears every day in the different stores. It is necessary use a high performance antimalware systems that do not use too much time to detect malware, since a comprehensive analysis may require too many resources.

Based on the analysis and findings of this work, there are a number of recommendations for the Android antimalware community. The conventional antimalware systems need to integrate user behavioral methods to detect target malware. In this work, we have proposed a new technique to detect target malware and there are other previous systems that make the same but using other kind of methods. The focus to resolve this problem would consist of developing a complete and good performance system to detect this kind of malware.

VI. Conclusion and future work

Nowadays, everyone have a mobile device with a lot of sensitive data inside. For the attackers, these devices are an important target. For this reason, the malware for mobile devices are an important security problem.

Attackers have improved their techniques and have developed different kind of malware. One of this type of malicious software is the targeted malware. This only performs his malicious functionality under certain circumstances, usually related with the user behavioral. For this reasons, targeted malware is very difficult to detect using conventional antimalware methods.

With the work developed in this thesis, we have defined the basis of a system capable to detect this kind of malware defining user behavioral scenarios for Android systems. It has analyzed the principal user events that trigger the targeted malware. We have defined a language capable to transform scenario definitions to dynamic analysis in the system developed. The creation of this system and the new language defined allows the automatization of a difficult part of the malware analysis process. The key problem with this type of malware is the way of detecting it because it depends on the execution context and user behavior. The usual analyzers do not consider the user behavioral as a factor to trigger malware and there could be a million of different behaviors.

Finally, mobile device malware is growing every year. Every few months new and more sophisticated types of malware appear. The malware analyzers must evolve to counter this important threat. This work means the appearance of a new technique to detect targeted malware in Android applications based on user behavioral manipulating mobile devices. Merging this system with traditional antimalware detectors means palliate the expansion of malware in mobile devices and forces to attackers to create new methods to bypass these security measures.

Future works include:

- Integrating the system with the cloud to establish a large automatic analysis system. This system could analyze a several number of applications at same time to find targeted malware inside. This system could be used by the main app stores.
- Integrating the system with Big Data techniques. It could be generated user behavioral patterns to create behavioral file inputs based on the results in previous analysis.

- Interpolating the user behavioral language to other systems, such as a personal computer or other mobile operative systems, to detect targeted malware.
- Incorporating new language elements to generate new user behavioral events used for new targeted malware to trigger his functionality.

References

[1] Bitbucket, «Targetdroid Repository,» [En línea]. Available: https://bitbucket.org/gtangil/targetdroid.

[2] IDC, «IDC: Smartphone OS Market Share, 2015 Q2,» [En línea]. Available: http://www.idc.com/prodserv/smartphone-os-market-share.jsp.

[3] Symantec, *Internet Security Threat Report,* 2016.

[4] G. Suarez-Tangil, M. Conti, J. E. Tapiador y P. Peris-Lopez, *Detecting Targeted Smartphone Malware with Behavior-Triggering Stochastic Models,* Madrid, 2014.

[5] D. Fleck, A. Tokhtabayev, A. Alarif, A. Stavrou y T. Nykodym, *PyTrigger: A System to Trigger & Extract User-Activated Malware Behavior.*

[6] K. Tam, S. J. Khan, A. Fattori y L. Cavallaro, *CopperDroid: Automatic Reconstruction of Android Malware Behaviors,* San Diego, 2015.

[7] Google, «Android: hardware-properties.ini,» [En línea]. Available: https://android.googlesource.com/platform/external/qemu/+/04b20f411ba9398fc2bfac3d93f0997cd93a86f9/android/avd/hardware-properties.ini.

[8] Google, «Android Debug Bridge,» [En línea]. Available: https://developer.android.com/studio/command-line/adb.html.

[9] X. Gouchet, «Launch intents using ADB,» [En línea]. Available: http://xgouchet.fr/android/index.php?article42/launch-intents-using-adb.

[10] Google, «Control the Emulator from the Command Line,» [En línea]. Available: https://developer.android.com/studio/run/emulator-commandline.html.

[11] A. D. R. A. A. J. P. Bertrand, «Remote administration tool for android devices,» [En línea]. Available: https://github.com/DesignativeDave/androrat.

[12] L. Xie, X. Zhang, S. Zhu y J.-P. Seifert, *pBMDS: A Behavior-based Malware Detection System for Cellphone Devices,* Hoboken, 2010.

[13] M. Herreros Díaz y G. Suarez-Tangil, *Análisis de ofuscación en Apps mediante casos de prueba,* Madrid, 2013.

[14] J. Gajrani, J. Sarswat, M. Tripathi, V. Laxmi, M. Gaur y M. Conti, *A Robust Dynamic Analysis System Preventing SandBox Detection by Android Malware,* Sochi, 2015.

[15] I. Burguera, U. Zurutuza y S. Nadjm-Tehrani, *Crowdroid: Behavior-Based Malware Detection System,* Chicago, 2011.

[16] T. Isohara, K. Takemori y A. Kubota, *Kernel-based Behavior Analysis for Android Malware Detection,* Saitama, 2011.

[17] A. Reina, A. Fattori y L. Cavallaro, *A System Call-Centric Analysis and Stimulation Technique to Automatically Reconstruct Android Malware Behaviors,* 2013.

[18] C. Painters, «Android emulator patch for configurable IMEI, IMSI and SIM card serial number,» [En línea]. Available: http://blog.codepainters.com/2010/11/20/android-emulator-patch-for-configurable-imei-imsi-and-sim-card-serial-number/.

[19] A. Kumar, «Useful Android adb commands over USB/Wi-Fi,» [En línea]. Available: https://thangamaniarun.wordpress.com/2013/04/19/useful-android-adb-commands-over-usbwi-fi/.

Appendix A: User Behavioral Language Schema

This appendix presents the User Behavioral Language schema defined in this thesis. It indicates the structure of the input files and the rules to define them. It uses a JSON format.

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "id": "input JSON to configure a scenario and launch android events",
  "type": "object",
  "required": ["name", "time"],
  "properties": {
    "name":{
      "id": "scenario name",
      "type": "string"
    },
    "time": {
      "id": "definition of context in every execution time",
      "type": "object",
      "oneOf": [
        {
          "type": "object",
          "required": ["id"],
          "properties": {
            "id": {
              "id": "execution time",
              "type": "integer",
              "enum": [-1]
            },
            "emulator": {
              "id": "emulator configuration params",
              "properties": {
                "name": {
                  "id": "emulator name",
                  "type": "string"
                },
                "port": {
                  "id": "emulator port",
                  "type": "integer"
                },
                "sdk_version": {
                  "id": "emulator android version",
                  "type": "string",
                  "enum":["android-8", "android-10", "android-15", "android-16", "android-17", "android-18", "android-19", "android-20", "android-21", "android-22", "android-23"]
                }
              }
            },
            "phone": {
              "id": "phone properties",
              "type": "object",
              "properties": {
                "brand": {
                  "id": "phone brand name",
                  "type": "string"
                },
                "device": {
                  "id": "phone device name",
                  "type": "string"
                },
                "hardware": {
                  "id": "phone hardware name",
                  "type": "string"
                },
```

```
                "imei": {
                  "id": "phone imei",
                  "type": "string"
                },
                "imsi": {
                  "id": "phone imsi",
                  "type": "string"
                },
                "network_provider": {
                  "id": "phone network provider name",
                  "type": "string"
                }
              }
            },
            "network": {
              "id": "network properties",
              "type": "object",
              "properties": {
                "IP": {
                  "id": "phone IP",
                  "type": "string"
                },
                "DNS": {
                  "id": "phone DNS server",
                  "type": "string"
                }
              }
            },
            "ini_properties":{
              "id": "properties to configurate the emulatore",
              "type": "object",
              "properties": {
                "cpu_arch": {
                  "id": "The CPU Architecture to emulator",
                  "type": "string"
                },
                "cpu_model": {
                  "id": "The CPU model (QEMU-specific string)",
                  "type": "string"
                },
                "ram_size": {
                  "id": "Device ram size",
                         "type": "integer"
                      },
                      "screen_type": {
                        "id": "Defines type of the screen",
                        "type": "string"
                      },
                      "main_keys": {
                        "id": "Hardware Back/Home keys",
                        "type": "boolean"
                      },
                      "track_ball": {
                        "id": "Track-ball support",
                        "type": "boolean"
                      },
                      "keyboard": {
                        "id": "Whether the device has a QWERTY keyboard",
                        "type": "boolean"
                      },
                      "keyboard_lid": {
                        "id": "Whether the QWERTY keyboard can be opened/clos
ed",
                        "type": "boolean"
                      },
                      "keyboard_charmap": {
```

```
123.                         "id": "Name of the system keyboard charmap file",
124.                         "type": "string"
125.                       },
126.                       "d_pad": {
127.                         "id": "Whether the device has DPad keys",
128.                         "type": "boolean"
129.                       },
130.                       "gsm_modem": {
131.                         "id": "phone DNS server",
132.                         "type": "string"
133.                       },
134.                       "gps": {
135.                         "id": "Whether there is a GPS in the device",
136.                         "type": "boolean"
137.                       },
138.                       "battery": {
139.                         "id": "Whether the device can run on a battery",
140.                         "type": "boolean"
141.                       },
142.                       "accelerometer": {
143.                         "id": "Whether there is an accelerometer in the devic
    e",
144.                         "type": "boolean"
145.                       },
146.                       "audio_input": {
147.                         "id": "Whether the device can record audio",
148.                         "type": "boolean"
149.                       },
150.                       "audio_output": {
151.                         "id": "Whether the device can play audio",
152.                         "type": "boolean"
153.                       },
154.                       "sd_card": {
155.                         "id": "Whether the device supports insertion/removal
    of virtual SD Cards",
156.                         "type": "boolean"
157.                       },
158.                       "sd_card_path": {
159.                         "id": "SD Card image path",
160.                         "type": "string"
161.                       },
162.                       "cache": {
163.                         "id": "Whether we use a /cache partition on the devic
    e",
164.                         "type": "boolean"
165.                       },
166.                       "cache_path": {
167.                         "id": "Cache partition to use on the device",
168.                         "type": "boolean"
169.                       },
170.                       "cache_size": {
171.                         "id": "Cache partition size",
172.                         "type": "string"
173.                       },
174.                       "lcd_width": {
175.                         "id": "LCD pixel width",
176.                         "type": "integer"
177.                       },
178.                       "lcd_height": {
179.                         "id": "LCD pixel height",
180.                         "type": "integer"
181.                       },
182.                       "lcd_depth": {
183.                         "id": "LCD color depth",
184.                         "type": "integer"
185.                       },
```

```
                        "lcd_density": {
                          "id": "Abstracted LCD density",
                          "type": "integer"
                        },
                        "lcd_backlight": {
                          "id": "LCD backlight",
                          "type": "boolean"
                        },
                        "gpu": {
                          "id": "Enable/Disable emulated OpenGLES GPU",
                          "type": "boolean"
                        },
                        "camera_back": {
                          "id": "Configures camera facing back",
                          "type": "string"
                        },
                        "camera_front": {
                          "id": "Whether the device can run on a battery",
                          "type": "string"
                        },
                        "heap_size": {
                          "id": "Max VM application heap size",
                          "type": "integer"
                        },
                        "sensor_proximity": {
                          "id": "Whether there is an proximity in the device",
                          "type": "boolean"
                        },
                        "sensor_magnetic_field": {
                          "id": "Provides magnetic field sensor values",
                          "type": "boolean"
                        },
                        "sensor_orientation": {
                          "id": "Provides orientation sensor values",
                          "type": "boolean"
                        },
                        "sensor_temperature": {
                          "id": "Provides temperature sensor values",
                          "type": "boolean"
                        },
                        "use_ext4": {
                          "id": "Specifies which file system to use: ext4 of yaffs2",
                          "type": "boolean"
                        },
                        "kernel_path": {
                          "id": "Path to the kernel image",
                          "type": "string"
                        },
                        "kernel_parameters": {
                          "id": "kernel boot parameters string",
                          "type": "string"
                        },
                        "ram_disk_path": {
                          "id": "Path to the ramdisk image",
                          "type": "string"
                        },
                        "system_partition_path": {
                          "id": "Path to runtime system partition image",
                          "type": "string"
                        },
                        "system_partition_init_path": {
                          "id": "Initial system partition image",
                          "type": "string"
                        },
```

```
                        "system_partition_size": {
                          "id": "Ideal size of system partition",
                          "type": "integer"
                        },
                        "data_partition_path": {
                          "id": "Path to data partition file",
                          "type": "string"
                        },
                        "data_partition_init_path": {
                          "id": "Initial data partition",
                          "type": "boolean"
                        },
                        "data_partition_size": {
                          "id": "Ideal size of data partition",
                          "type": "integer"
                        },
                        "snap_storage_path": {
                          "id": "Path to snapshot storage",
                          "type": "string"
                        }
                      }
                    }
                  }
                },
                {
                  "type": "object",
                  "required": ["id"],
                  "properties": {
                    "id": {
                      "id": "execution time",
                      "type": "integer",
                      "enum": [0]
                    },
                    "install_app": {
                      "id": "apks to install",
                      "type": "string"
                    },
                    "timezone": {
                      "id": "timezone for the emulator",
                      "type": "string"
                    },
                    "telnet": {
                      "id": "telnet commands",
                      "type": "object",
                      "properties": {
                        "power": {
                          "id": "power properties",
                          "type": "object",
                          "properties": {
                            "ac": {
                              "id": "ac properties",
                              "type": { "enum": ["on", "off"] }
                            },
                            "capacity": {
                              "id": "capacity properties",
                              "type": "integer"
                            },
                            "health": {
                              "id": "power health properties",
                              "type": {"enum": ["unknown", "good", "overheat",
    "dead", "overvoltage", "failure"] }
                            },
                            "present": {
                              "id": "battery presence",
                              "type": "boolean"
                            },
```

```
                            "status": {
                              "id": "battery status",
                              "type": { "enum": ["unknown", "charging", "discha
    rging", "not-charging", "full"] }
                            }
                          }
                        }
                      }
                    }
                  },
                  {
                    "type": "object",
                    "required": ["id"],
                    "properties": {
                      "id": {
                        "id": "execution time",
                        "type": "integer",
                        "minimum": 1
                      },
                      "telnet": {
                        "id": "telnet commands",
                        "type": "object",
                        "properties": {
                          "power": {
                            "id": "power properties",
                            "type": "object",
                            "properties": {
                              "ac": {
                                "id": "ac properties",
                                "type": { "enum": ["on", "off"] }
                              },
                              "capacity": {
                                "id": "capacity properties",
                                "type": "integer"
                              },
                              "health": {
                                "id": "power health properties",
                                "type": {"enum": ["unknown", "good", "overheat",
    "dead", "overvoltage", "failure"] }
                              },
                              "present": {
                                "id": "battery presence",
                                "type": "boolean"
                              },
                              "status": {
                                "id": "battery status",
                                "type": { "enum": ["unknown", "charging", "discha
    rging", "not-charging", "full"] }
                              }
                            }
                          },
                          "gsm": {
                            "id": "calling events",
                            "type": "object",
                            "properties": {
                              "call": {
                                "id": "call to a number",
                                "type": "string"
                              },
                              "accept": {
                                "id": "accept a call",
                                "type": "string"
                              },
                              "busy": {
                                "id": "busy a call",
```

```
                                "type": "string"
                              },
                              "cancel": {
                                "id": "cancell a call",
                                "type": "string"
                              },
                              "data": {
                                "id": "data connection status",
                                "type": { "enum": ["unregistered", "home", "roaming", "searching", "denied", "off", "on"] }
                              },
                              "voice": {
                                "id": "voice connection status",
                                "type": { "enum": ["unregistered", "home", "roaming", "searching", "denied", "off", "on"] }
                              }
                            }
                          },
                          "sms": {
                            "id": "send a sms message",
                            "type": "object",
                            "properties": {
                              "number": {
                                "id": "number phone to send a sms",
                                "type": "string"
                              },
                              "text": {
                                "id": "message",
                                "type": "string"
                              },
                              "required": [ "number", "sms" ]
                            }
                          },
                          "geo": {
                            "id": "location events",
                            "type": "object",
                            "properties": {
                              "fix": {
                                "id": "http://jsonschema.net/t1/geo/fix",
                                "type": "object",
                                "properties": {
                                  "longitude": {
                                    "id": "longitude",
                                    "type": "string"
                                  },
                                  "latitude": {
                                    "id": "latitude",
                                    "type": "string"
                                  },
                                  "altitude": {
                                    "id": "altitude",
                                    "type": "string"
                                  },
                                  "required": [ "longitude", "latitude" ]
                                }
                              },
                              "nmea": {
                                "id": "nmea",
                                "type": "string"
                              }
                            }
                          }
                        }
                      },
                      "adb": {
                        "id": "adb popular commands",
```

```
                    "type": "array",
                    "properties": {
                      "action": {
                        "id": "adb action name",
                        "type": "string",
                        "enum": ["connect_wifi", "lock_screen", "unlock_scree
n", "volume_up", "go_home", "take_screenshot", "start_clock_app", "stop_clock_
app", "start_wifi", "wifi_status", "enable_wifi", "disable_wifi", "enable_mobi
le_data", "disable_mobile_data"]
                      }
                    }
                  },
                  "intents": {
                    "id": "intents commands",
                    "type": "array",
                    "properties": {
                      "type": {
                        "id": "intent type",
                        "type": "string",
                        "enum": ["start", "start_service", "broadcast"]
                      },
                      "params": {
                        "id": "intent params",
                        "type": "string"
                      }
                    }
                  },
                  "monkey": {
                    "id": "monkey script events",
                    "type": "object",
                    "properties": {
                      "path":{
                        "id": "path to monkey script",
                        "type": "string"
                      },
                      "package": {
                        "id": "apk package where run monkey script",
                        "type": "string"
                      }
                    }
                  }
                }
              }
            ]
          }
        }
      }
```

Appendix B: Events list

This appendix presents all the events that the system accepts and their values.

Emulator Configuration Events

Attribute	Type/Values	Description
emulator		
emulator.name	string	AVD name
emulator.port	integer	AVD port
emulator.sdk_version	string values: android-8 android-10 android-15 android-16 android-17 android-18 android-19 android-20 android-21 android-22 android-23	Android sdk version for AVD
phone		
phone.brand	string	Mobile device brand
phone.device	string	Mobile device model
phone.imei	string	Mobile device IMEI
phone.imsi	string	Mobile device IMSI
phone.provider	string	Mobile device provider
network		
network.IP	string	Mobile device IP
network.DNS	string	The value of <servers> must be a comma-separated list of up to 4 DNS server names or IP addresses.
ini_properties [5]		
ini_properties. cpu_arch	string	The CPU Architecture to emulator
ini_properties. cpu_model	string	The CPU model (QEMU-specific string)
ini_properties. ram_size	integer	The amount of physical RAM on the device, in megabytes.
ini_properties. screen_type	string values: touch multi-touch no-touch	Defines type of the screen.
Ini_properties. track_ball	boolean	Whether there is a trackball on the device.

ini_properties. main_keys	boolean	Whether there are hardware back/home keys on the device.tr
ini_properties keyboard	boolean	Whether the device has a QWERTY keyboard.
ini_properties.keyboard_lid	boolean	Whether the QWERTY keyboard can be opened/closed.
ini_properties.keyboard_charmap	string	Name of the system keyboard charmap file.
ini_properties.d_pad	boolean	Whether the device has DPad keys
ini_properties.gsm_modem	string	Whether there is a GSM modem in the device.
ini_properties.gps	boolean	Whether there is a GPS in the device.
ini_properties.battery	boolean	Whether the device can run on a battery.
ini_properties.accelerometer	boolean	Whether there is an accelerometer in the device.
ini_properties.audio_input	boolean	Whether the device can record audio.
ini_properties.audio_output	boolean	Whether the device can play audio.
ini_properties.sd_card	boolean	Whether the device supports insertion/removal of virtual SD Cards.
ini_properties.sd_card_path	string	SD Card image path
ini_properties.cache	boolean	Whether we use a /cache partition on the device.
ini_properties.cache_path	string	Cache partition to use on the device. Ignored if disk.cachePartition is not 'yes'.
ini_properties.cache_size	string	Cache partition size
ini_properties.lcd_width	integer	LCD pixel width.
ini_properties.lcd_height	integer	LCD pixel height.
ini_properties.lcd_depth	integer	Color bit depth of emulated framebuffer.
ini_properties.lcd_density	integer values: 120 160 240 213 320	A value used to roughly describe the density of the LCD screen for automatic resource/asset selection.
ini_properties.lcd_backlight	boolean	Enable/Disable LCD backlight simulation,yes-enabled,no-disabled.
ini_properties.gpu	boolean	Enable/Disable emulated OpenGLES GPU.
ini_properties.camera_back	string	Configures camera facing back. Must be 'emulated' for a fake camera, 'webcam<N>' for a web camera, or 'none' if back camera is disabled.

ini_properties.camera_front	string	Configures camera facing front. Must be 'emulated' for a fake camera, 'webcam<N>' for a web camera, or 'none' if front camera is disabled.
ini_properties.heap_size	integer	The maximum heap size a Dalvik application might allocate before being killed by the system. Value is in megabytes.
ini_properties.sensor_proximity	boolean	Whether there is an proximity in the device.
ini_properties. sensor_magnetic_field	boolean	Provides magnetic field sensor values.
ini_properties. sensor_orientation	boolean	Provides orientation sensor values.
ini_properties. sensor_temperature	boolean	Provides temperature sensor values.
ini_properties. use_ext4	boolean	Specifies which file system to use: ext4 of yaffs2
ini_properties. kernel_path	string	Path to the kernel image.
ini_properties. kernel_parameters	string	Kernel boot parameters string.
ini_properties. system_partition_path	string	Path to runtime system partition image.
ini_properties. system_partition_init_path	string	Initial system partition image.
ini_properties. system_partition_size	integer	Ideal size of system partition.
ini_properties. data_partition_path	string	Path to data partition file. Cannot be empty. Special value <temp> means using a temporary file. If disk.dataPartition.initPath is not empty, its content will be copied to the disk.dataPartition.path file at boot-time.
ini_properties. data_partition_init_path	string	Initial data partition. If not empty, its content will be copied to the disk.dataPartition.path file at boot-time.
ini_properties. data_partition_size	integer	Ideal size of data partition.
ini_properties. snap_storage_path	string	Path to a 'snapshot storage' file, where all snapshots are stored.

Table 2: Emulator Configuration Events

OS Configuration Events

Attribute	Type/Values	Description
install_app	string	Path to apk to install
timezone	string	Timezone used by emulated mobile device.
power		
power.ac	string values: on off	Set AC charging state to on or off.
power.capacity	integer	Set remaining battery capacity state (0-100).
power.health	string values: unknown good overheat dead overvoltage failure	Set battery health state.
power.present	boolean	Set battery presence state.
power.status	string values: unknown charging discharging not-charging full	Change battery status as specified.

Table 3: OS Configuration Events

Execution Context Events

Attribute	Type/Values	Description
power		
power.ac	string values: on off	Set AC charging state to on or off.
power.capacity	integer	Set remaining battery capacity state (0-100).
power.health	string values: unknown good overheat dead overvoltage failure	Set battery health state.
power.present	boolean	Set battery presence state.
power.status	string values: unknown charging discharging not-charging full	Change battery status as specified.
gsm		
gsm.call	string	Simulate an inbound phone call from <phonenumber>.
gsm.accept	string	Accept an inbound call from <phonenumber> and change the call state "active".
gsm.busy	string	Close an outbound call to <phonenumber> and change the call state to "busy".
gsm.cancel	string	Terminate an inbound or outbound phone call to/from <phonenumber>.
gsm.data	string values: unregistered home roaming searching denied off on	Change the state of the GPRS data connection to <state>.
gsm.voice	string values: unregistered home	Change the state of the GPRS voice connection to <state>.

	roaming searching denied off on	
sms		
sms.number	string	Sender phone number
sms.message	string	SMS message
geo		
geo.fix.longitude	string	Send a simple GPS longitude to the emulator instance. It is mandatory.
geo.fix.latitude	string	Send a simple GPS latitude to the emulator instance. It is mandatory.
geo.fix.altitude	string	Send a simple GPS alititude to the emulator instance. It is optional.
geo.nmea	string	Send an NMEA 0183 sentence to the emulated device, as if it were sent from an emulated GPS modem.
intents		
intents.type	string	Intent type
intents.params	string	Intent parameters
adb		
adb.action	string values: connect_wifi lock_screen unlock_screen volumen_up go_home take_screenshot start_clock_app stop_clock_app start_wifi enable_wifi wifi_status enable_mobile_data disable_mobile_data	Action to execute
monkey		
monkey.path	string	Monkey script path
monkey.package	string	Package where the monkey script will be execute

Table 4: Execution Context Events

Appendix C: Planning and budget

In order to comply with the regulations of MsC Thesis by Universidad Carlos III de Madrid, this appendix presents the Planning and Budget of the Thesis.

First, it is going to be presented a planning that defines this project in different tasks. It has been generated a typical Gantt chart presenting in a graphical form the duration of each phase.

The resume of the phases is shown in Table 4 with its start and end date. Gantt chart with detailed information is shown in Figure 6.

	Duration (days)	Start Date	End Date
Definition of goals	27	16/01/2016	23/02/2016
Analysis	34	01/03/2016	15/04/2016
Design	14	24/05/2016	10/06/2016
Development	16	13/06/2016	04/07/2016
Testing	4	5/07/2016	10/07/2016
Report	34	18/07/2016	1/09/2016

Table 5: Work Planning

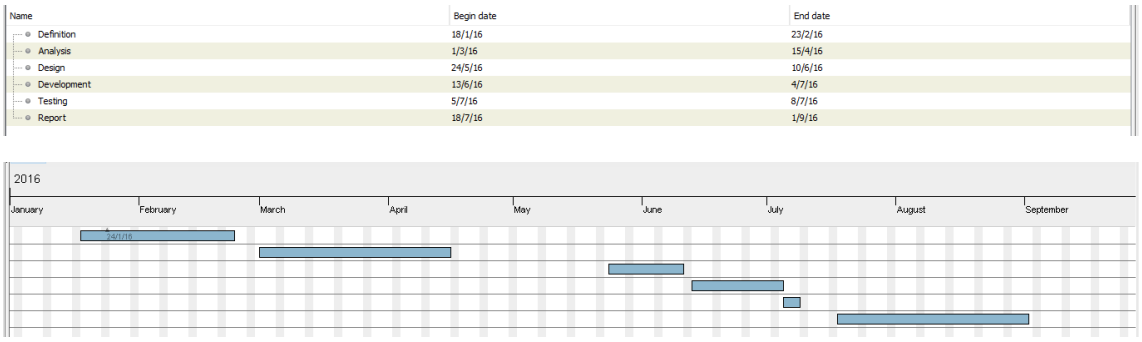

Figure 6: Work Planing Gantt Chart

Last, it is going to be presented the budget. It is broken down into various estimations by type cost:

- Personal cost: 31.600 €
- Hardware cost: 30 €
- Software cost: 0 €
- Indirect cost: 301 €
- Total cost: 31.600 €

Note: VATs are included.

Concept	Time (hours)	Fees	RRHH cost
Security Engineer	500	50 €/hour	25.000 €
PhD	120	55 €/hour	6.600 €
TOTAL			31.600 €

Table 6: Estimated Project Personal Cost

Concept	Units	Unit price	Estimated life time	Airtime (months)	Cost
Acer Aspire 5740	1	600 €	60 months	4,3	30 €
TOTAL					30 €

Table7: Estimated Project Hardware Cost

Concept	Units	Unit price	Estimated life time	Airtime (months)	Cost
Ubuntu Linux	1	0 €	-	4,3	0 €
Sublime Text 3	1	0 €	-	4,3	0 €
SmartGit Hg 7.1	1	0 €	-	4,3	0 €
Github	1	0 €	-	4,3	0 €
Bitbucker	1	0 €	-	4,3	0 €
MiKTeX	1	0 €	-	4,3	0 €
TeXnic Center	1	0 €	-	4,3	0 €
Adobe Reader	1	0 €	-	4,3	0 €
Google Chrome	1	0 €	-	4,3	0 €
TOTAL					0 €

Table8: Estimated Project Software Cost

Concept	Monthly price	Airtime (months)	Cost
Electricity	40 €	4,3	172 €
Internet	30 €	4,3	129 €
TOTAL			301 €

Table9: Estimated Project Indirect Costs

Concept	Amount
Personal cost	31.600 €
Hardware cost	30 €
Software cost	0 €
Indirect cost	301 €
TOTAL	31.931 €

Table10: Estimated Project Total Cost