

Free University of Bozen/Bolzano
Faculty of Computer Science

Thesis

**A Computational Forensic Methodology for Malicious
Application Detection on Android OS**

Svetlana Voronkova

Submitted in partial fulfillment of the requirements for degree of
Master in Computer Science at the Free University of Bolzano

Thesis advisor: Prof. Francesco Di Cerbo

March, 2011

Contents

1	Introduction	8
2	State of Art	10
2.1	The concept of digital forensics	10
2.1.1	The potential digital evidence in computer-related crimes	10
2.1.2	The forensic investigation process	11
2.1.3	The most common digital forensics techniques	12
2.2	The mobile forensics	12
2.2.1	Traditional forensics challenges when dealing with mobile forensics	13
2.2.2	The mobile forensics methodologies	13
2.3	Mobile device threats	14
2.3.1	Smartphone malware review	15
2.4	Short overview of most common mobile forensics tools	17
2.4.1	MOBILedit! Forensic	17
2.4.2	Oxygen Forensic Suite	18
2.4.3	CellDEK	19
3	Research Approach	21
3.1	Android OS	21
3.1.1	Android security model	21
3.1.2	Android's issues with threats	22
3.1.3	Existing Android forensics strategies	22
3.2	The role of data mining	23
3.2.1	Frequent set mining	24
3.2.2	Applications' categorization profiles	25
3.2.3	Data preprocessing	25
3.3	Apriori frequent set mining example	27
4	Suspicious Application Detection Algorithm	30
4.1	The use case	30
4.2	The pseudo-code	32
4.3	The complexity	33

4.4	Evaluation of Detection Algorithm	35
4.4.1	Correctness evaluation	36
4.4.2	Performance evaluation	37
5	Proposed System Prototype	40
5.1	Outline	40
5.2	Proposed system architecture	40
5.3	AForensics for Android	40
5.3.1	User interface	41
5.4	The Web Forensics application	42
5.4.1	User interface and functionality	43
5.4.2	The reporting mechanism	44
6	Related Works	45
6.1	Open Source Android Forensics application	45
6.2	Permission based technology by SMobile	45
7	Conclusions and future work	46
	References	48
A	APPENDIX	51
A.1	Flow chart of the Suspicious Application Detection Algorithm	51

List of Figures

1	Cumulative number of malware signatures [19].	15
2	Smartphone malware effects [22].	16
3	Example of a XML forensic report [20]	18
4	Oxygen Forensic Suite [21]	19
5	Data Matrix	27
6	Main steps of Suspicious Application Detection Algorithm	31
7	The distribution of permissions in the classification model	35
8	Execution time graphs form apps with 1-5 permissions (a), with 6-10 permissions(b), with more than 10 permissions (c)	38
9	Execution time comparison	39
10	System Architecture	41
11	AForensics main menu(a), AForensics final report representation (b)	41
12	User notification	42
13	Web Forensics, Menu	43
14	Web Forensics, report	44
15	The flow chart of the Suspicious Application Detection Algorithm	52

List of Tables

1	Set of classification profiles, defined on the basis of the relevant Android security permissions.	26
2	The example of applications with their permissions	27
3	The set L1	28
4	The set L2	28
5	The set L3	29
6	The set L4	29
7	The correctness test for the applications with 1-5 permissions	36
8	The correctness test for the applications with 6-10 permissions	36
9	The correctness test for the applications with more than 10 permissions	37

Abstract

E-discovery includes any process of searching, storing and securing digital information. Digital forensics is one of the least explored branches of e-discovery that examines how the extraction of digital evidence can aid in crime investigations and identification of potential suspects. Due to the growing popularity of smartphones, the field of mobile forensics - that is a part of digital forensics - gains more and more importance.

However, many existing mobile forensic techniques cannot be (yet) fully applied to the Android OS, as its file system and architecture differ from already existing mobile platforms in many aspects. Moreover, Android provides access to a wide range of libraries and sophisticated functions, that make the operating system very attractive to malicious software creators and increases the probability of its use in various illegal activities.

This study presents a semi-automated mobile forensics methodology aimed at supporting a forensics examiner in detection of suspicious applications such as the ones that are generally considered as malware, i.e. those that might exploit sensible data stored on the mobile devices in a vulnerable way.

The technique is based on the features of Android's security model, namely the set of standard permissions exposed by each application. The methodology relies on a set of more than 13000 safe applications, hosted on the Android Market and collected with AppAware, a specific tool for Android OS.

The proposed system includes a web-based application for forensic professionals that allows to perform the detection of malicious applications on the chosen source and to manage the results of analysis. Another component is an Android application that presents the shorten version of detection report to the mobile device end user.

The paper presenting this approach was submitted and accepted to the 4th International Workshop on Computational Forensics (IWCF), that took place in November 2010 [1]. The work presented has received a positive feedback.

Acknowledgments

It is a pleasure to thank the people who made this thesis possible.

I am first of all grateful to my supervisor, Prof. Francesco Di Cerbo, for the patient guidance, encouragement, advices and support he has provided throughout all the research and thesis writing period.

Further thanks go to Andrea Girardello and Florian Michahelles for providing the data without which this research could not be carried out. In addition I would like to thank them for the contribution to the article that we have presented in 4th International Workshop on Computational Forensics (IWCF).

I am thankful to the staff of Computer Science Faculty of Free University of Bozen/ Bolzano for their competence and help during my Master Degree studies and this university.

I would like to thank also my patient thesis reviewers: Oana Tifrea, Andrea Janes and Sarunas Marciuska. Their constructive comments and critics indeed helped to improve this thesis. Additional thanks to Oana for being such a good flatmate, taking over all my domestic responsibilities and letting me to concentrate on the writing.

The special thanks goes to Lijana Lubyte, who not only always kept me company whenever I needed a break from studying or writing the thesis, but also has been a very good friend during past 3 years.

I wish to thank also my colleagues from “Eurac Research” ICT department for their support, understanding and nice, friendly environment that they have created in the office and maintained during all these months.

Finally, I thank to my parents, my sister, all my friends in Lithuania (particularly to Veslava Filipovic and Giedre Aniulyte), who despite the geographical distance were always nearby, listened to me and supported in every decision I have taken.

1 Introduction

The cellular industry had grown and evolved rapidly in the past few years. As the capabilities and functionality of mobile devices are constantly improving and new innovative technologies are adopted, more and more people rely on mobile technologies. Nearly every small scale device includes features like high-bandwidth Internet connection, location tracking etc. The amount and importance of personal data carried by PDA's, cellular phones and smartphones became similar to those stored in computers and laptops. All these sophisticated capabilities and data in modern mobile devices make them very attractive to malicious software developers, and increase the probability of their use in various illegal activities. This aspect is particularly relevant for smartphones.

The growing interest in Android devices implies that also lawbreakers will start to use those devices for their crimes and to design malicious applications to conduct frauds and other malicious activities. Evidence discovered during the analysis of a mobile device involved in a crime, is usually very beneficial for the overall investigation process, and has a high value in proving guilt or innocence of potential suspects in a trial. Here evidence is interpreted as information of probative value that is stored or transmitted in binary form [3]. However, mobile forensics is somewhat challenging for the investigators, mainly due to the rapid change of technologies and gaps in the law enforcement. Despite the fact that the digital forensics evolves very quickly the forensics technologies and tools are far behind when it comes to mobile device investigation.

The key issue that makes mobile forensics so complex is that the cell phone market offers a variety of hardware, equipped with different operating systems in different versions. Operating systems from different manufacturers usually rely on different file systems, and have different memory management mechanisms. Forensic tools face many problems handling devices that adopt a variety of standards and properties.

In this context, the Android Operating System for smartphones is relatively new, still rapidly evolving. The market share of Android-based smartphones tends to grow constantly, and new Android tablets start to be available as well. What is generally referred to as "Android OS" delivers a complete software suite for mobile devices: an operating system, middleware and key mobile applications [34]. It enables the developers to take advantage of all functionalities and features included in the handset, to create innovative and sophisticated mobile applications. It also provides a convenient mechanism for application distribution. Android Market: software providers can publish their applications in a few steps, reaching very easily (potentially) all Android users. At the same time, through Android market users can search for applications and install them in seconds. As an open source operating system that relies on the Linux kernel, Android is considered to be a secure platform. Nevertheless the Android malware market exists, even if it's still in an infancy stage. Several applications that have been detected on the Android Market only alert the need of forensics tools oriented to this particular mobile OS.

It is more and more important that the forensic tools, that generally assist the examiner in her investigation, are able to support the analysis of these devices. The aim of the following research is to design and develop a semi-automated malware detection system for Android mobile operating system, in order to support the activities of a forensics or security examiner.

In particular, the following study presents a web-based spyware and malware detection system that is going to support forensic examiners in detection of suspicious applications, i.e. the ones that might exploit sensible data stored on mobile devices in a vulnerable way. The developed method is not going to clearly determine the maliciousness of a certain application, it simply signals the forensics analyst a situation that could require additional specific investigations. The final decision as to whether to consider the analyzed application suspicious belongs to the examiner. Therefore the system is semi-automated. The methodology has been trained on a set that consists of more than 13000 applications hosted on the Android Market and collected with AppAware, an Android application that captures and shares installations, updates and removals of Android third party applications. Additionally, the system includes an Android application intended for mobile devices users. Its aims are different, considering general public and forensics analysts. In the former case, the application warns

a user about possible malicious applications present in her mobile phone, leaving to her the decision whether to remove such applications or not. In the latter, the application collects a list of installed applications and their requested security permissions, in order to enable the analyst to conduct a forensics analysis on them.

In the following chapter of this thesis the background in digital and mobile forensics is presented. The further chapters discuss the goals of the work and technologies analyzed. Then, it is presented the description of the model, that is at the base of the methodology. The study proceeds by presenting the work accomplished, conclusions and the future works on the studied field.

2 State of Art

2.1 The concept of digital forensics

Electronic discovery, also called e-discovery, refers to any process in which electronic data is sought, located, secured, and searched with the intent of using it as evidence in a civil or criminal legal case [2]. The digital forensics is one of the least explored branches of e-discovery. Generally this term is defined as follows: “digital forensics also called computer forensics is simply the application of computer investigation and analysis techniques in the interests of determining potential legal evidence. Evidence might be sought in a wide range of computer crime or misuse, including but not limited to theft of trade secrets, theft of or destruction of intellectual property, and fraud” [4].

The core goals of the digital forensics are “the preservation, collection, validation, identification, analysis, interpretation, documentation and presentation of after-the-fact digital information derived from digital sources for the purpose of facilitating or furthering the reconstruction of events as forensic evidence” [4].

In the face of the increasing amount of cyber-crimes and computer abuses encountered today the digital forensic science is rapidly gaining the importance. The evidence gathered using the computer forensics methodologies has a great value in the court and helps to solve computer-related cases. Moreover sometimes such methods also clarify certain details of cases that not necessarily involve a computer crime. Computers are not the only means by which crimes might be conducted. In the typical investigation any kind of digital devices such as PDA or smartphones, routers, modems, external storage/flash drives, etc. can be examined. However, due to the rapid growth of this relatively new discipline many of the existing laws, legal precedents, and practices related to computer forensics have many gaps and do not cover properly all the cases that involve digital devices.

2.1.1 The potential digital evidence in computer-related crimes

Digital evidence is becoming a routine part of criminal cases. Nearly 85% of the current cases in one or another way are dealing with the evidence retrieved from digital devices [5]. The potential computer evidence is considered to be any digital information that might be related to the investigated case. Typical media that may hold such evidence are computer, notebook, CDs, DVD's, flash drives as well as Smartphones, iPods, cell phones, digital cameras, and other kinds of electronic devices. The media containing an evidence related to the case is not necessarily found on the crime scene, it can be retrieved also from different location. According to a guide by the US Department of Justice, any electronic data that is potential evidence is often not obvious. Moreover it has the possibility to transcend borders with ease and speed, is fragile and can be easily altered, damaged, or destroyed. The electronic data is also time sensitive [6]. Therefore, in order to be able to prove the origin of the information, the evidence has to be handled with a great care. When dealing with digital information as a potential evidence every forensic specialist must respect the 3 following guidelines:

- Do not alter or damage the original.
- Proof that your recovered evidence is the same as the original.
- Inspect evidence without altering it.

There are two main types of data that are collected as an evidence during the forensics process – **persistent data and volatile data**. The example of **persistent data** is the data stored on a hard drive or another digital media and which is preserved when the digital device is shut down. The **volatile data** refers to any type of data that is stored in memory (cash, RAM) or exists in transmission and is lost when the device is

turned off. It is essential to an investigator to know in which case what kind of data should be captured, as often the attempt to capture one type of data may lead to the loss of the other type of evidence.

2.1.2 The forensic investigation process

Computer-related crime is a crime that encompasses all of those illegal activities that are committed by or with the aid of a computer or an information technology. Additionally, computer-related crimes include the activities where the computers are the target of the criminal enterprise. [7]. Therefore there are distinguished two types of investigations where the computer forensic methodology can be used:

- when the device was used as an instrument to commit a crime e.g. malicious software, hacking;
- when the computer is a target of a crime e.g. fraud by means of Internet.

In the first case the investigator is not necessary present when the machine used in the crime is switched off and the investigation will mostly focus on the persistent data. In the contrary, the digital device's involvement as a target issues the capturing of the information present in RAM and running process, i.e. volatile data. Regardless of the situation, the techniques, methodologies and procedures used are the same. The typical digital forensic investigation process is based on the following phases:

- preservation;
- data collection;
- examination;
- analysis;
- reporting.

Preserving the information in the original state is the most important phase of the investigation process, as it ensures the integrity of the information collected on the crime scene and provides as much as possible relevant data to the next steps of the overall process. This is analogous to taking photographs, fingerprints, or blood samples from a crime scene. The preservation involves operations such as evaluation of the risk of turning off the computer, preventing people from using computers during the data collection, disconnecting the device from the network.

The data collection phase includes searching for evidence in the sources that had some contribution to the committed crime. The investigator should assure that no data was changed during the evidence gathering and examination phases. Moreover it's very important to maintain properly the custody chain, which means documenting all the evidence found and all of the software and versions used in its extraction[8]. Having such an up to date documentation makes the evidence creditable in court.

The amount of data collected during the data collection phase is usually huge. Therefore in order to extract from this data the information of particular interest, the forensic **examination methods** and techniques need to be applied. There are two main ways in which the collected data are examined: live and dead analysis. **Dead analysis** is a traditional data examination where the investigations are performed on hard drive or another data, after the device used in the crime was shut down. Nowadays the **live analysis** is becoming more and more popular. It is suitable when the attack was performed using just device's memory (RAM), but not touching any data on the hard drive.

Once the relevant information has been extracted, the **analysis phase** has to be performed. The main goal of the analysis is to derive valuable information that addresses the questions that initiated the investigation process.

The overall forensic process is concluded with the **report**. Report can be written or oral. The investigator should pay attention that it would be written in a clear way, using natural language so that individuals without a background in Computer Science would be able to understand it.

2.1.3 The most common digital forensics techniques

The typical computer related crimes that require digital data examination can include hacking, phishing, sexual abuse, various frauds, as for example credit card frauds, hidden steganographic data transfers and so on. Sometimes the advantage of computer forensics techniques might be taken while dealing with the cases not directly related to the digital devices, as homicide or theft. However, the investigators usually apply the same forensic techniques to the both types of crimes. The most common techniques recently used in the digital forensics investigation process are: IP Address and Email tracing, file structure and storage media analysis, and recently also GPS data analysis.

IP Address and Email tracing are the methods that are applied when the examiner has to investigate the cases that involve use of Internet and computer networks. Internet Protocol Address Tracing means to trace an IP address right down to its real address. Email Address Tracing refers to the situations when it is important to know the exact location of the email sender. The tracing operation mainly focuses on email header analysis as it includes the details about source machine IP address, real email server from which the email originated, the date and time.

Other digital forensics techniques are the ones that involve computer system analysis, mainly file system analysis and storage media examination. When performing **file structure analysis** the investigator usually looks for the files that may be encrypted or hashed. Generally, the automated tools are used to process and decrypt such files. However in some special cases also manual interference might be required. Yet different actions are performed during the **physical or removable disk analysis**. These disks might have been erased and it can become almost impossible to recover data from it. Therefore data recovery tools and methods have to be adopted by investigators when dealing with physical media.

Last but not least, in some circumstances, it can be interesting to know where a certain digital device has been in a certain moment, therefore also GPS-enabled mobile phones can provide valuable information to answer the question. Modern smartphones have navigation-support software and in some cases it is possible to find past positions in their caches.

2.2 The mobile forensics

The mobile forensics science, as a part of digital forensics, focuses on recovering digital evidence from small scale digital devices under forensically sound conditions using accepted methods [9]. The phases of digital forensics investigation for mobile devices are the same as for computers: data collection, examination, analysis and reporting. However, the mobile forensics is somewhat challenging for the investigators, mainly due to the rapid change of technologies and gaps in the law enforcement. Despite the fact that the digital forensics science evolves very quickly the forensics technologies and tools are far behind when it comes to the mobile device investigation. Unlike the computers, mobile devices vary in their standards, use of operating system and hardware, filesystem and memory management [10]. One of the key issues that makes mobile device forensic more complex than traditional computer forensic is the constant activity of the mobile device. The information stored on the device tends to change continuously making the reproduction of the exact evidence very complicated [10]. Moreover, the mobile devices also have a better memory access control than the computers. For these reasons the traditional digital forensic techniques can be sometimes ineffective for the data extraction from mobile phones. Another key difference between computer and mobile forensic is the operating system and the filesystem. The cell phone market offers a variety of operating systems and their versions. The OS from different manufacturers use filesystems that differ from the standards used in

computers. The forensics tools face many problems handling all these cases.

The computers usually have a certain standards and requirements for hardware, whereas the hardware architecture of mobile devices differs from model to model. The operating systems are highly customized by the mobile phone producers in the way that they would correspond to the hardware of the certain phone model and additional devices such as digital camera, GPS, wireless network connection [11]. Therefore even the same versions of OS might support different hardware architecture for different manufacturers. Lately becoming very popular smartphones perfectly reflect all above mentioned issues.

2.2.1 Traditional forensics challenges when dealing with mobile forensics

The standard digital forensics techniques usually differ from whether the device of interest if found turned on or turned off. The same rule applies also tho the mobile device investigation process. However many difficulties and challenges are encountered using these techniques when dealing with small scale devices. The off-line analysis of mobile phones gets more complex than computer examination due to the fact that mobile phones contain an internal clock, which continuously changes data stored in the cell phone's flash memory. Therefore it is impossible to reproduce a consistent bit-to-bit image of the entire memory as is it is done during the computer examination.

Considering the live forensics - the technique that applies to the switched on devices, the connectivity of the phone plays a vital role [12]. It is necessary to keep device isolated from any networks during the analysis phase, otherwise it may lead to the loss of some information that could be beneficial for the running investigation. The preservation of this requirement is very complicated because of the expanded connectivity of small scale devices i.e. the possibility to deal with cellular network and through it with Internet services.

2.2.2 The mobile forensics methodologies

The mobile device is a combination of a handset, a SIM card and often also a memory card. The mobile forensics methodologies focus though mainly on two different areas: SIM card forensics and memory forensics.

The Subscriber Identity Module (SIM) card is a smart card that contains personal user information as well as the network data that is necessary to authenticate and identify subscribers on the Network. Each SIM card is secured with the 4 digits personal identification number(PIN) that must be entered once the mobile device is turned on. The typical data that can be found during the SIM card investigation process includes: SIM card serial number, details of previous calls made, the date and time, received and missed calls, received, saved or deleted SMS messages. Additionally, the SIM card stores also the contact list, the phone book, calendar entries, and images [13]. For the SIM card forensics examination it is essential to prevent the connection to the any mobile phone network, as this may result in overwriting data of a great importance to the case. The first step to be taken in such situation is removing the SIM card from the analyzed device and making an image or a copy of it. The action is performed using special device that copies the contents from the SIM card of interest to the blank card while not allowing it to connect to any networks. The analysis is then carried out on the freshly copied card so that the original data could not be damaged or altered in any other way. This phase is usually performed via PC applying standard forensics data extraction methodologies. The following types of the SIM card analysis might be accomplished by a forensic investigator [14]:

- Integrated Circuit Card ID (ICCID) : each SIM card is internationally identified via its ICCID. This 18 or 19 digit number is stored on the SIM. Knowing this number the analyst can identify the international location of the device.
- International Mobile Subscriber Identity (IMSI): the number that identifies the individual operator

network, which is the network the SIM card works on. The network provider communicates with the SIM card via this number and it is used to attach phone calls to the SIM.

- Mobile Country Code (MCC): the three digit number is used to identify the country from which the SIM card is originated.
- Mobile Network Code (MNC): the code that is used in conjunction with the above mentioned MCC to identify the Network provider to which the SIM card belongs.
- Mobile Station International Subscriber Directory Number (MSISDN): the 15 digit number which uniquely identifies the subscription in either a UTMS or GSM network.
- Abbreviated Dialling Numbers (ADN): a list of numbers that the user of the SIM card stores to have an easy access to the numbers to dial, basically this is the user's contact list. From the contact list the analyst can identify the incoming and outgoing calls to the certain phone numbers as well as the time and date of any calls made or received.
- Short Message Services (SMS): commonly known as text messages. Depending on the SIM card the analyst may get the information about sent/received/deleted messages.

The goal of **memory forensics** is to extract data stored in the flash memory of the cellular phone and to find out the meaningful evidence [15]. The memory forensics is divided into the physical data analysis and logical data analysis. The logical data acquisition approach focuses on the extraction on data carried out by flash memory of the mobile device on the filesystem level. The analysis is accomplished via computer with the use of a forensic tool that is communicating with mobile phone system. As in the case of SIM card analysis the copy of entire filesystem is done to guarantee the integrity of evidence and not to irreversibly alter the important information. The data that can be extracted applying the logical analysis varies with the forensic tools used, but typically include SMS, MMS, call registers, videos, pictures, audio files and calendar entries.

When performing the physical data analysis the memory of the cell phone is seen in its raw form. Physical data acquisition approach implies the extraction of full memory image bit-by-bit excluding the expandable memory. The physical level access method can enhance the recovery rate for deleted data in comparison with the logical level access method [15]. The physical data analysis is performed using flasher tools, JTAG tests access, or by removing a flash memory from a mobile device and reading its contents with a memory chip reader [16]. The flasher tools are good for making the memory image as they have a simple hardware connection with the cell phone. The second approach - JTAG emulator dumps a whole physical memory and produces a complete mobile device memory image that is ready for the forensics analysis. If the analyzed device is damaged or by any other reason none of the above mentioned methods can be applied the memory can be read using a memory chip reader.

2.3 Mobile device threats

As the popularity of mobile devices and especially smartphones among customers increases the devices and their OS are more and more often exploited by malicious software developers. **Malicious software** is a software that attempts to spy, use, damage or destroy a digital device or information stored on it without notifying the user. The malicious software or malware is divided into groups with respect to the way it may harm the system. By today there exist the following categories of malicious software: Trojans, spyware, rootkits, adware, worms and viruses. Although the detection, analysis and prevention of malicious software are the mobile phone security tasks, the forensics and security science are strongly related when it comes to the smartphone or other small scale device analysis. The reason for such a relation is a new emerging threat - the malicious applications that steal, monitor or in any other way exploit the user's sensitive data stored on the device. The sensitive data in this case are contact lists, SMS messages, call history, location

tracking, account credentials etc. One of the main groups of such applications are the ones that use a digital steganography in order to obtain user's sensitive data. **Steganography** is the science of hiding data within data. The information might be hidden in many different ways from a message encoded in the second letter of each word of a large body of text to such sophisticated techniques as hiding the sensitive information within image, audio or video files [17].

2.3.1 Smartphone malware review

According to the mobile malware evolution report by Peter Gostev from Kaspersky Lab [18] the appearance of the first malicious software for smartphones is dated for June 2004, which overlaps with increasing popularity of small scale devices. The worm Caribe released by a professional virus writing group 29A was targeting a Symbian OS – the most popular mobile platform at that time. Its main task was to check autonomously for bluetooth devices in range and establish a connection to devices with activated bluetooth in order to transmit an infected Symbian OS Installation System (SIS) file. After the execution of transferred.sis file Caribe was installing and repeating the same behavior on the newly infected mobile device [18].

The latest research of Kaspersky Lab presents the following mobile malware evolution statistics:

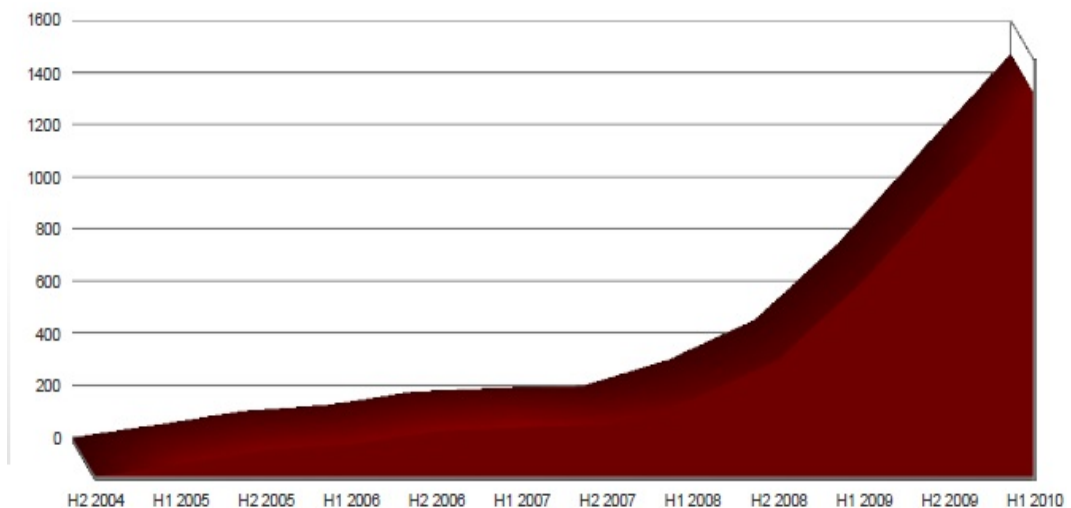


Figure 1: Cumulative number of malware signatures [19].

The chart (figure 1) indicates the first notifiable growth in malicious software evolution for years 2005-2006. It is very likely that such an increase is related to the introduction of a certificate-based signing of applications for Symbian OS. Starting from the Symbian S60 3rd Version, which is currently used on most smartphones the application certification became mandatory thus significantly reducing the number of new malware appearances for next two years. The number of new malicious mobile software starts to increase again in 2008 and grows till now. This fact is mostly correlated with the launch of new mobile platforms: Apple's iPhone and Google's Android OS. Both platforms use an online store for distributing the software. This approach is relatively new in the world of smartphones, therefore the malware propagation on these platforms is still basically not investigated field of research. However, the first malware identification cases on both Android and iPhone alert about the emerging vulnerability of these platforms and rise the need of the tools that would prevent their exploitation by malicious software developers.

Depending on the nature of the mobile threat and the way it infects the device different malicious activities may occur on the phone. The Dai Labor researchers of the Technical University of Berlin suggest the following

categorization of the malicious activities based on their collected data [22]:

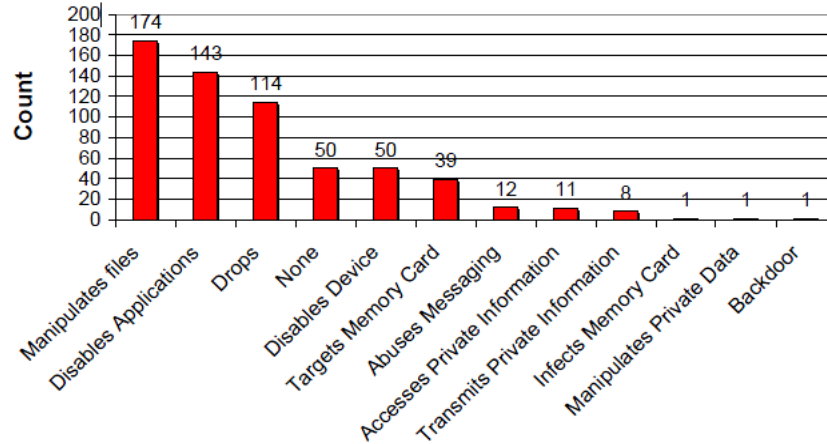


Figure 2: Smartphone malware effects [22].

Most of the malicious applications simply manipulate the files stored on the device or attempt to disable other installed applications. The other part of malicious activities affects the sensitive data stored on the devices, mainly accesses, transmits or in any other way manipulates the private data of the users.

Due to the rapid evolution of mobile devices and their advanced capabilities the distinctions between them and the computers has been significantly minimized. The users more often choose a smartphone over the PC to perform such activities as emailing, data storing, Internet browsing, banking etc. Therefore there is a higher chance that small scale device might be affected by the same type of malware that exists for traditional computers. The United States Computer Emergency Readiness Team proposes the following list of the mobile threats that includes the types of malware typical for computers as well as the ones that particularly hit smartphones [23]:

- social engineering;
- exploitation of social networking;
- mobile botnets;
- exploitation of mobile applications; and
- exploitation of e-commerce.

The malware spreading through social engineering is the most known method widely used to infect the computers and now as well the smartphones. The social engineering attack is defined as “an attack, where an attacker uses human interaction (social skills) to obtain or compromise information about an organization or its computer systems” [24]. The most popular social engineering forms that may affect small scale devices are **phishing**, **vishing** and **smishing**.

The most common **phishing** activity is an email pretending to be from a valid financial or e-commerce provider. The users are typically manipulated to provide their personal information or financial data. The example of such phishing attack on smartphones is an Android application called “DROID09” that was discovered on Android Market in January, 2010. The application “pretended” to be a useful online banking utility but turned out that it was actually stealing online banking credentials of the users. Indeed it is not known how exactly the application was performing the fraud [25].

Vishing stands for “Voice Phishing”. As the phishing it tricks the individuals into revealing their financial or personal information just that the attacks are carried out adopting voice technologies. **Smishing** approach intends to manipulate the users via SMS and text messages.

The growing trend of social networking sites such as Facebook, MySpace or Twitter has brought a new way of mobile device exploitation. The information sharing among user via social networks is usually unwarranted and not completely secure, therefore there always exists a risk that the device might get infected

A **botnet** is a set of compromised computers or bot clients running malicious software that enables a “both-order or “botmaster to control these computers remotely[23]. Due to the good integration with the Internet the bontents are starting to be applied also to the mobile networks. The security researchers Derek Brown and Daniel Tijerina with Tipping Point’s Digital Vaccine Group have built an intentionally malicious application that was able to connect nearly 8,000 iPhones and Android Smartphones into a mobile botnet. The created application simply links to the Weather Underground Website and provides weather forecast information to its users. However in the background the app is actually gathering information on the users who downloaded it, including their GPS coordinates and phone numbers.[26]. The experiment made by the researches only proves how easy it is possible to exploit the mobile devices via botnet.

The innovative approach of application distribution via online stores used by Android and iPhone also has its vulnerabilities. Potentially anyone can place his developed application on the online store and make it available to the thousands of users. Although the applications are accepted only after review process, there is no guarantee that the malware may not simply bypass the criteria set by reviewers and be placed on the market. The good example is the “aka Tap Snake” application identified on Android Market. The application is capable of sending out a users GPS location via HTTP POST to the site gpsdatapoints.appspot.com/addpoint. The information can then be retrieved by a remote user using another “app aka GPS SPY” [27].

Another trend that started quickly to migrate to mobile devices is e-commerce. The mobile devices are used not only to look up the product information but also to make the orders, purchases and complete other financial operations. Such an integrity of mobile devices is very welcomed by the users as it provides them with the possibility to accomplish the purchase or to make a credit card payment from any location and at any time. There exists a set of different applications for smartphones that accept payment or even swipe credit card data. However such application might be as well exposed by the malicious software writer applying “skimming”. **The skimmig** is the illegal copying of information from the magnetic strip of a credit card in order to gain the access to the bank account of the victim.

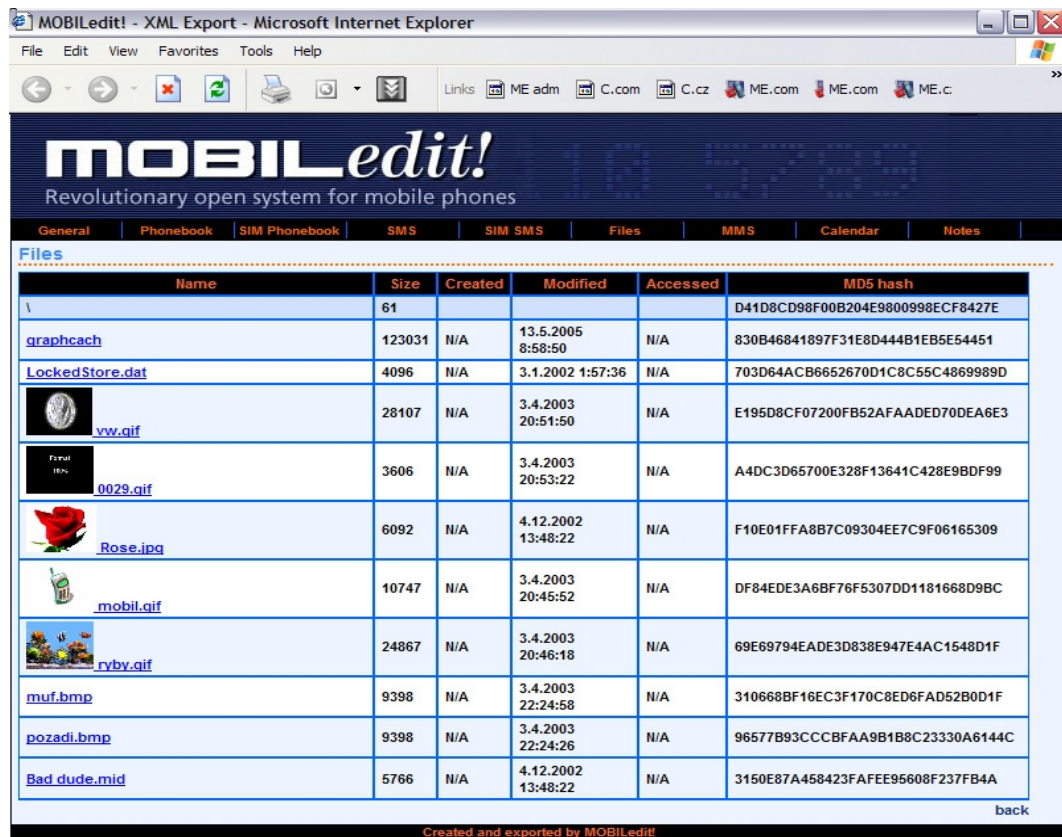
2.4 Short overview of most common mobile forensics tools

The market offers a very wide range of different tools for secure data extraction, evidence collection, data aquisition and analysis from the mobile devices. In this section, we discuss some of the most known and commonly used tools and software solutions of mobile forensics. Some more forensic software solutions are discussed in [28].

2.4.1 MOBILedit! Forensic

The MOBILedit! Forensic is considered to be one of the most trusted and reliable mobile device investigation tools used in 70 different countries. It enables the full control of the mobile device from PC via cable, IrDA or Bluetooth allowing to view the contents of the phone, perform searches and complex synchronizations, make full data copy or send SMS or MMS messages. In addition, the explicit forensic device investigation is offered. MOBILedit! Forensic collects all possible data stored on the small scale device and generates an extensive and completely secure report onto a PC. The software handset also allows the user to customize

the final report adapting it to the certain rules or needs of the judicial system. The reporting system is very flexible as it supports different languages as well as variety of formats such as MS Word, XLS, and XML.








Name	Size	Created	Modified	Accessed	MD5 hash
\	61				D41D8CD98F00B204E9800998ECF8427E
graphcach	123031	N/A	13.5.2005 8:58:50	N/A	830B46841897F31E8D444B1EB5E54451
LockedStore.dat	4096	N/A	3.1.2002 1:57:36	N/A	703D64ACB6652670D1C8C55C4869989D
 vv.gif	28107	N/A	3.4.2003 20:51:50	N/A	E195D8CF07200FB52AFAADED70DEA6E3
 0029.gif	3606	N/A	3.4.2003 20:53:22	N/A	A4DC3D65700E328F13641C428E9BDF99
 Rose.jpg	6092	N/A	4.12.2002 13:48:22	N/A	F10E01FFA8B7C09304EE7C9F06165309
 mobil.gif	10747	N/A	3.4.2003 20:45:52	N/A	DF84EDE3A6BF76F5307DD1181668D9BC
 ryby.gif	24867	N/A	3.4.2003 20:46:18	N/A	69E69794EADE3D838E947E4AC1548D1F
muf.bmp	9398	N/A	3.4.2003 22:24:58	N/A	310668BF16EC3F170C8ED6FAD52B0D1F
pozadi.bmp	9398	N/A	3.4.2003 22:24:26	N/A	96577B93CCCCBFAA9B1B8C23330A6144C
Bad dude.mid	5766	N/A	4.12.2002 13:48:22	N/A	3150E87A458423FAFEE95608F237FB4A

Figure 3: Example of a XML forensic report [20]

The MOBILedit! Forensic supports a broad range of mobile phones and due to the rapid growth of the mobile device market the list of supported mobile phones is constantly updated. As a reliable software solution MOBILedit! Forensic corresponds to the judicial forensic standards. The tool prevents any changes in the device ensuring the correctness and validity of the collected evidence. Moreover all data blocks, such as the SMS are protected by the MD5 hash algorithm.

According to the Cell Phone Forensic Tools analysis report by US National Institute of Standards and Technology Moibledit! Forensics performs best on logical data acquisition level and manages the following type of data: phone and subscriber information, phonebook, SIM phonebook, missed calls, last numbers dialed, received calls, inbox, sent items and drafts. Moreover the tool gives examiners the ability to acquire SIM card data using a PC/SC-compatible reader [29].

2.4.2 Oxygen Forensic Suite

The Oxygen Forensics Suite is one of the leaders in the mobile device investigation tools with the main focus on smartphones. The tool supports more than 1600 different mobile device models including the the newest Symbian OS, Nokia S60, Sony Ericsson UIQ, Windows Mobile 5/6, Blackberry, iPhone and Android Smartphones. The forensic version of Oxygen Suite is available for police departments, law enforcement units, and all government services that wish to use the software for investigative purposes. The Oxygen Forensics

Suite enables the communication with the investigated device via cable, IrDA and Bluetooth in the same way as The MOBILedit! Forensic does.

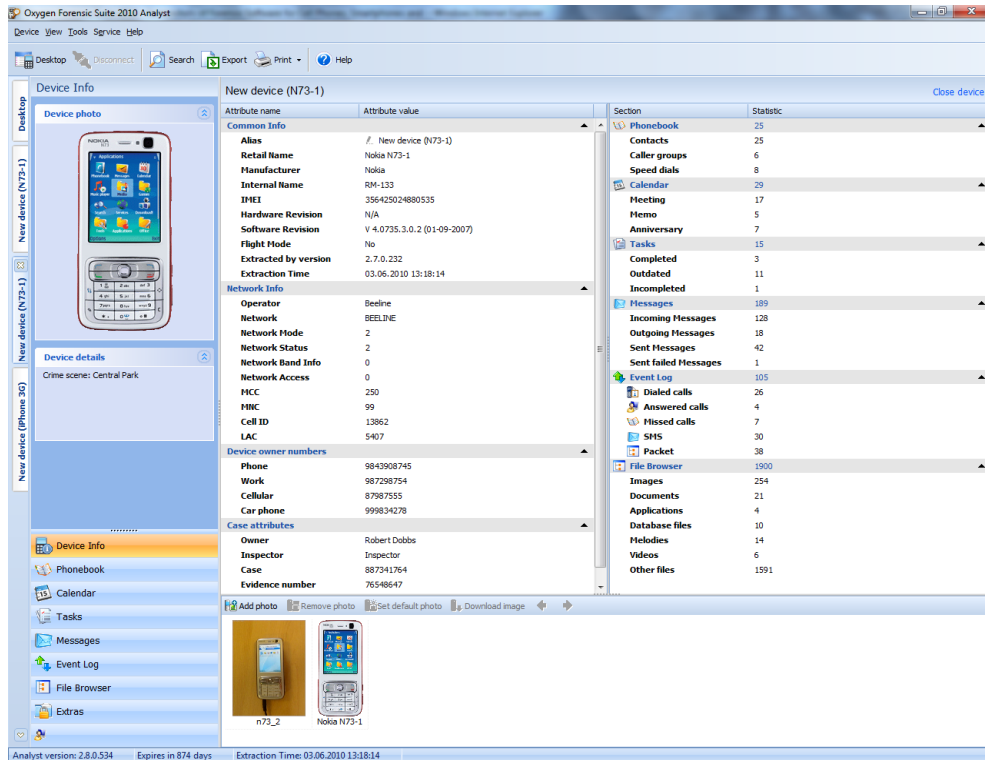


Figure 4: Oxygen Forensic Suite [21]

The Software interface is specially designed for forensic analysis, data search and reporting. Depending on the model of the examined device there can be retrieved the basic information such as SIM-card data, contacts list, outgoing/incoming calls, standard SMS/MMS/E-mails, calendar, text notes, pictures and videos. Besides the standard smartphone data the tool offers the extraction of unique information such as SMS messages deleted from phone memory, extended phonebook information which includes contact photos, caller groups, speed dials and last contact modification date and time, MMS and E-mail attachments, GPS positioning data etc. In addition, the tool offers the reporting with the support of the most popular file formats such as XLS, RTF and PDF.

2.4.3 CellDEK

The CellDEK is the first portable handset data extraction kit designed for use at the scene of a crime and all working environments associated with on-going investigations. It is compatible with 2000 of the most popular cell phones and PDA's, including Blackberrys. CellDEK now is providing the support also for Apple iPhone, iPhone 3G and iPod touch devices and for satellite navigation devices such as TomTom, Garmin. As the above discussed tools The CellDEK unit provides examiners with the ability to connect to the investigated device via a cable, Bluetooth or IrDA connection. The toolkit contains an embedded touch-screen PC, data cables for different phone manufacturers, PC/SC SIM card reader and a write-protected flash memory card reader.

The CellDEK software is intuitive and no specialist skills are required to operate the system. The data extraction customization is also offered by the system so that the examiner can select himself which data

actually to acquire. The following data can be extracted using the toolkit : Handset Time and Date, Serial Numbers (IMEI, IMSI), Dialed Calls, Received Calls, Phonebook, SMS, MMS messages, Deleted SMS from SIM, Calendar, Memos, To Do Lists, Pictures, Video, and Audio.

Cell Phone Forensic Tools analysis report by US National Institute of Standards and Technology indicates CellDEK as a reliable forensic toolkit that can be sufficiently used for extensive information acquisition [29]. The main advantage of the tool is the ability to produce the report that includes detailed log of the acquisition process, all commands sent to the target device and all data received from the target device. Such a functionality allows the examiner to recover the data or the set of previously performed actions in case of the acquisition process failure. However CellDEK does not have a built in search functionality. The search in the generated reports can be performed applying some external third party tools.

3 Research Approach

The aim of this study is to design and develop a mobile forensics oriented, semi-automated malware detection system for Android OS based on the features of Android's security model and the application of known data mining classification technique – **Frequent Itemset Identification**. The methodology relies on the analysis of a set of more than 13000 safe applications hosted on the Android Market.

3.1 Android OS

Android is an operating system for mobile devices that includes middleware and key applications, and relies on the modified version 2.6 of the Linux kernel. It was initially developed by Android Inc., a company later purchased by Google, and lately by the Open Handset Alliance [30]. Android is an open development platform that provides to the developers the ability to create an advanced applications using any kind of resources of the platform such as hardware, services, databases etc. Each Android application runs in its own process on Dalvik, a custom virtual machine designed for embedded use. The handset also provides a convenient mechanism for application distribution – the Android Market, where software providers can publish their applications, making them available to the smartphone users.

3.1.1 Android security model

Android's security model combines the standard Linux OS features that control the security at the process level and the permission based mechanism. **The permission** [34] is a right that a developer has to declare in its application to be able to interact with the system or access the components of other applications. As each program runs in a separate process, typically applications neither read nor write each others data or code and sharing data between applications must be done explicitly. However, after the declaration of permissions an application has access to the protected features of Android to which the permission refers. A permission is generally simple text string assigned to a predefined list of functionalities of the system, i.e., "INTERNET" to connect to the Internet, "READ_SMS" to read SMS messages, and so on. Permissions have to be statically defined in the application package so that during the deployment the user could grant them to the application, or abort the process. For example, the permission that allows the application to write the data to the SD card looks as presented in the listing 1.

```
1 <permission xmlns:android="http://schemas.android.com/apk/res/android"
2     android:name="com.isecpartners.android.READ_CONTACTS"
3     android:description="@string/access_contacts_for_reading"
4     android:protectionLevel="normal"
5     android:label="@string/access_perm_label">
6 </permission>
```

Listing 1: An example of an Android OS security permission

Each permission contains the following characteristics: name, description, label and the protection level.

Generally Android provides a list of standard permissions that grant the applications access to such functionalities as Internet, SMS/MMS messaging, contacts, external memory card etc. However, the developers are free to create their own permissions making some particular component available to the other applications installed on the device.

As mentioned, the role of the permissions is not only to grant the access to the system's components but also to inform the user while installing the application about the features that it is going to use. Such notifications allow the user to evaluate the dangerousness of each application and accept it or dismiss. Once the application is installed its permissions cannot be changed.

3.1.2 Android's issues with threats

With the growing trend of smartphones the focus of malicious software developers has broaden from computer based malware to the one committed to harm smartphone systems including the Android OS. As an open source handset that relies on Linux Kernel, Android is considered to be a secure platform. Nevertheless the Android malware market is still in an infancy stage, the detection of some malfunction applications on Android Market had proven that the devices can be easily exploited by attackers.

Recently, a report of SMOBILE [33] considering 48694 applications, found 29 of them to be possibly spyware available on the Android Market, while for other 383 it is possible to access authentication credentials stored on the mobile phone. The presented analysis has some commonalities with the approach adopted by SMOBILE, because both of them use applications' permissions as a privileged source of information on mobile applications. The report of SMOBILE confirms the urgency of developing anti-malware systems and checks, as well as forensics methodologies.

The example of Android malware is an application called "Android FakePlayer" that was discovered in August 2010 [32]. The application is distributed as a simple media player application, however in the background it acts as Trojan horse and attempts to send SMS messages to predetermined numbers without the user's knowledge or permission. The Trojan sends the SMS message with the texts "798657" to the premium numbers 3353 and 3354. Despite the fact that the risk rating of the "Android FakePlayer" was evaluated as low, it results in a extra charge of the users according to the respective number's rate.

However, in order to develop efficient forensics tools there should be a clear definition of what kind of applications should be considered as suspicious in Android. Basically, all applications that have the ability to spy the users sensitive data in a specific way by hiding them and transmitting it outside the local system can be considered as malfunction. Some mobile forensics tool developers suggest to follow the rule of thumb and to regard as spyware the applications that allow 3rd party to spy on the activities of the user and that hide themselves from the 3rd party at the same time. Generally, if the user cannot determine the functionality of the certain application during its running time or cannot look up its name and icon in the applications list, such application is considered to be a suspicious application [33].

3.1.3 Existing Android forensics strategies

It is obvious that data extraction task is one of the most critical and sensitive in mobile forensics process and the security framework created by Android developers makes it even more complicated. The protection of the applications and the data is achieved by combination of two enforcement mechanisms: one at the system level using standard Linux operations and at the inter-component communication level either through the "permission" mechanism or through per-URI permissions. The critical point for the data extraction task from Android devices is that each Android application runs as a unique user identity and by default has no permission to perform any action that could have impact on the other application, the system or any other data on the device. The application's process is as a "sandbox", meaning that all the data that application collects and stores is isolated so that other processes cannot see or use it. So far there are four different techniques available for Android forensics [35]:

- SD Card analysis;
- logical acquisition;
- physical acquisition;
- chip-off.

Every Android device contains a **microSD card** that uses FAT32 file system, therefore it is possible to remove the card from the device and perform its analysis using standard forensic methods. However this method does not contribute much to the forensics of the overall system as the SD card contains only the data directly stored there by the installed applications, the part of the data still remains in the isolation of the application process.

Logical acquisition technique is the method that performs data extraction on the application level. With the logical acquisition approach is usually possible to retrieve such information as call logs, browser history, contacts and SMS messages. The example of an application that takes advantage of logical acquisition technique is “The Open Source Android Forensics” application that gathers call logs, browser history, contacts and SMS messages. Obtained data are stored in CSV files on the SD card inserted into device, so that the standard forensic analysis methods can applied to them [31].

For extraction of more significant data a **physical acquisition** might be applied. The physical acquisition methodology for Android OS is called “dd image. The “dd” image makes a complete copy of different system’s partitions, which can be analyzed using traditional forensics methods. These partitions use the open source file system YAFFS2 (Yet Another Flash File System 2) which is one of the significant challenges related to the Android platform. However, the “dd” image technique requires root privileges.

Chip-off analysis technique focuses on the explicit examination of Androids NAND flash memory and requires the analyst to have the full forensics lab facilities.

The approach presented in this study is considered to be a logical acquisition methodology as it examines only the file system level data. Moreover as the application discussed in [31] the tool developed during this research works as a usual Android application that can be installed on the device only for the data collection purposes and uninstalled after the collection phase is finished.

3.2 The role of data mining

The proposed methodology aims at detection of suspicious applications. Android security permissions are used as characteristics by which it is possible to identify the applications that may hide their real features or manage sensitive user’s data without their notification. The approach is mainly focused on the applications that explicitly declare the use of permissions connected with personal information: credentials, calendar, events, email, SMS/MMS and so on.

The technique adopted for suspicious application detection refers the widely used data mining methodology – **data classification**. Classification is “a data mining function that assigns items in a collection to target categories or classes” [37]. The typical classification problem deals with the class prediction for the analyzed data unit based on the information retrieved from initial dataset also called a **training dataset**. Classification process is a supervised learning process as the objects in the training set are accompanied by labels indicating the class. The classification process is based on two main steps: model construction and model usage. The model describes a set of predetermined classes using the information of the training dataset. To create a model, the classification algorithm typically performs the analysis of the dataset characteristics with the known classes trying to discover the specific patterns called also “predictors”. The discovered patterns identify the “rules” by which the one or the another category can be assigned to the object. The model might be represented as classification rules, decision trees, or mathematical formula.

In the second phase of the classification process previously created model serves as a base for classifying future or unknown objects. The classes to the unknown data units are assigned by comparing values of its attributes with the patterns that describe the predefined classes.

In the scope of this research the objects that need to be classified are the Android applications retrieved from the device. The applications might be categorizes as “suspicious”, “not suspicious” or “possibly suspicious”

according to the set of security permissions that they declare to use. The classification model is represented by the set of patterns – subsets of security permissions that identify the level of “suspiciousness” of analyzed objects. In order to discover the specific subsets of security permissions the data mining technique - frequent set discovery was applied.

3.2.1 Frequent set mining

Frequent set mining is one of the most basic data mining tasks that tries to identify some specific patterns in large data loads. This task is essential when dealing with clusters, classifiers, association rules or other kinds of correlations. The frequent set mining problem originated in market basket analysis, which examines the sets of items purchased by customers and tries to discover some specific dependencies between those items [38]. The marketing specialists use the results of association rule mining for item positioning in the store.

The detection of frequent patterns in the sets of Android permissions can be equated with the market basket problem. The security permissions can be thought as the products that occur together. Then, the applications that declare the permissions play the role of the customers. In this context the frequent itemset is a set of Android security permissions that very often occur together.

Let's formalize the definition of the a frequent itemset. Let $I = \{i_1, i_2, \dots, i_m\}$ be a set of items.

The couple (t', I') is then the **transaction** over I , where t' is the transaction identifier and I' is a set of items from I .

A **database** D over I is a set of transactions over I such that each transaction has a unique identifier.

A transaction $T = (t', I')$ *supports* a set X if $X \subseteq I'$. The set of transactions identifiers of transactions in D that support X compose a *cover* of a set X . The *support* of a set X in D is equal to the number of transactions in the cover of X in D .

A set is called **frequent** if it satisfies the minimal support threshold s_0 , with $0 \leq s_0 \leq |D|$, where the support of an itemset X is determined as a number of transactions T containing X :

$$s(X) = |\{T \in D : X \subset I\}| \quad (1)$$

The choice of the support threshold determines the number and usefulness of the found frequent itemsets. Generally, the collection of frequent sets with respect to chosen support is defines as follows:

$$F(D, s) := \{X \in I | s(X, D) > s_o\} \quad (2)$$

The frequent set mining is a first phase of well known data mining problem – association rule mining. There is a big number of algorithms that provide an effective solution for frequent itemset retrieval. Considering the suspicious application identification problem the Apriori association rule mining algorithm was chosen to perform the discovery of security permissions that frequently occur together.

The Apriori is an improvement of first proposed association rule mining algorithm called AIS [38]. The main advantage Apriori is that instead of determining the support of all possible itemsets and selecting those that satisfy the support threshold it performs first breadth-first search starting the itemsets of length “1” on each iteration identifying the candidates for frequent itemset. More specifically, it finds sets C_k of k-itemsets based on so called apriori principle, which states that the set is a frequent set if all its subsets are also frequent sets[39].

However, before the actual pattern discovery the classification model requires to complete some additional data preparation steps, mainly the choice of the suitable training dataset, the support threshold as well as data cleaning and data preprocessing if needed.

The data preprocessing phase as well as the frequent set discovery in transformed dataset using Apriori approach are performed with the help of specific data mining called Weka. The Waikato Environment for

Knowledge Analysis (Weka) is a software that has collected together a set of machine learning algorithms for data mining tasks such as attribute selection, association rules, clustering, prediction and classification algorithms. The tool was chosen generally by the following reasons:

- is an open source Data Mining software package written in Java
- capable to manage big data loads
- suitable also for not data mining specialists
- supports data preprocessing and frequent itemset discovery
- support many different parameter for Apriori, namely lower and upper bound for the support threshold, different metric types etc.

3.2.2 Applications' categorization profiles

As already stated before the methodology is mainly focusing on the applications that have an access to user's personal data. Therefore we have defined a set of categorization profiles, to describe applications that in some way manage user's sensitive data. This categorization (table 1) is based on the analysis of the default set of Android permissions and takes into account features that are connected with each sensible data category profile. To analyze an application, multiple profiles could be considered, according to the specific functionalities offered by the application. Regarding the aim of the approach, the first demand for the applications in the training dataset is to fit at least one categorization profile.

In order to build an accurate data classification model the large training dataset has to be considered. To complete this step we have taken the advantage of AppAware, a tool developed at Information Management group at ETH Zurich to monitor the diffusion of Android mobile apps [40]. In this way, at present we have analyzed over 42000 different Android applications and have selected a training set of 13098 applications that fit the categorization profiles meaning have access to the private user's information. All the objects in the training dataset are assumed to be safe as they are not reported to be malicious by users with the AppAware features and distributed worldwide. This allows us to state that the sample considered is heterogeneous enough for the build of an appropriate classification model.

3.2.3 Data preprocessing

Classification models perform the best on sparse categorical data. Here sparse data is data for which only a small fraction of the attributes are non-zero or non-null in any given row. The typical example of sparse data is already mentioned market basket, where we try to identify collection of items that customer purchases and how the purchase of one item depends on others. In the scope of presented methodology we can think of applications as customers and permissions used together in an application as a set of items purchased. To show that permission dataset is sparse and that association rules model can be applied, some data preprocessing has to be applied, mainly the transformation into appropriate format. The data transformation was performed following 3-step approach:

- Create a matrix, where each row indicates an application and each column indicates a separate permission.
- The matrix contains 2 values "1" or "?", where one specifies the presence of permission in the certain application and "?" indicates that permission is not present (See Figure 5).

Profile	Set of permissions
Contacts	WRITE_SYNC_SETTINGS,WRITE_CONTACTS ,READ_CONTACTS, MANAGE_ACCOUNTS, GET_ACCOUNTS, ACCOUNT_MANAGER
SMS-MMS messages	WRITE_SYNC_SETTINGS, WRITE_SMS, VIBRATE, SET_ORIENTATION, SEND_SMS,RECEIVE_SMS,RECEIVE_MMS, READ_SMS, FLASHLIGHT, BROADCAST_SMS
Call Log	VIBRATE, PROCESS_OUTGOING_CALLS, FLASHLIGHT, CALL_PHONE, CALL_PRIVILEGED
Audio & Video	WRITE_EXTERNAL_STORAGE, SET_ORIENTATION, RECORD_AUDIO, MODIFY_AUDIO_SETTINGS, GLOBAL_SEARCH, FLASHLIGHT, BLUETOOTH, BLUETOOTH_ADMIN, CAMERA
Tasks & Calendar & Organizer	WRITE_CALENDAR, REORDER_TASKS, READ_CALENDAR, GLOBAL_SEARCH, GET_TASKS, BLUETOOTH_ADMIN
Browser History	WRITE_SYNC_SETTINGS, WRITE_HISTORY_BOOKMARKS, READ_HISTORY_BOOKMARKS
Images	WRITE_EXTERNAL_STORAGE, SET_WALLPAPER_HINTS, SET_WALLPAPER, SET_ORIENTATION, READ_FRAME_BUFFER, GLOBAL_SEARCH, FLASHLIGHT, BLUETOOTH, BLUETOOTH_ADMIN, CAMERA
Phone Settings	WRITE_SYNC_SETTINGS, WRITE_SECURE_SETTINGS, WRITE_GSERVICES,WRITE_APN_SETTINGS, SET_WALLPAPER, SET_WALLPAPER_HINTS, SET_ORIENTATION, READ_SYNC_STATS, READ_SYNC_SETTINGS, READ_PHONE_STATE, MODIFY_AUDIO_SETTINGS, FLASHLIGHT, ACCESS_CHECKIN_PROPERTIES, CHANGE_CONFIGURATION, DEVICE_POWER
Email & services related to web	WRITE_SYNC_SETTINGS, WRITE_GSERVICES, USE_CREDENTIALS, SET_ORIENTATION, MANAGE_ACCOUNTS,GLOBAL_SEARCH, GET_ACCOUNTS, FLASHLIGHT, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_NETWORK_STATE, ACCESS_WIFI_STATE,ACCOUNT_MANAGER, CHANGE_NETWORK_STATE, INTERNET
Location Information & Tracking	WRITE_GSERVICES, INSTALL_LOCATION_PROVIDER, ACCESS_COARSE_LOCATION, ACCESS_FINE_LOCATION, ACCESS_LOCATION_EXTRA_COMMANDS, ACCESS_MOCK_LOCATION, ACCESS_NETWORK_STATE, CONTROL_LOCATION_UPDATES, INTERNET

Table 1: Set of classification profiles, defined on the basis of the relevant Android security permissions.

- Save the created matrix to the CSV file, that later will be passed as a input to the chosen data mining tool. This step is needed in order to remove not necessary attributes from the dataset and to specify that values of the attributes are not numerical (as it is by default), but categorical.

```
App_ID, WRITE_SYNC_SETTINGS, WRITE_SMS, VIBRATE, SET_ORIENTATION, SEND_SMS,
1881 ?, ?, 1, ?, 1, 1, 1, 1, 1, 1, 1, 1, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?,
1882 ?, ?, ?, ?, 1, ?, ?, 1, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, 1, 1, 1, 1, 1, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?,
1883 ?, ?, ?, ?, 1, 1, ?, ?, 1, 1, ?, ?, ?, ?, ?, ?, ?, ?, 1, ?, ?, ?, ?, ?, ?, 1, 1, 1, ?, ?, ?, ?, ?,
1884 ?, ?, 1, ?, 1, ?, ?, 1, 1, 1, ?, ?, 1, 1, ?, ?, ?, ?, ?, 1, 1, ?, ?, ?, ?, ?, ?, ?, 1, 1, 1, 1, 1,
1885 ?, ?, ?, ?, 1, 1, 1, 1, 1, 1, 1, ?, ?, 1, ?, ?, ?, ?, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, 1, ?, ?, ?,
1886 ?, ?, ?, ?, 1, 1, 1, ?, 1, ?, 1, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 1, 1, ?, ?, ?,
1887 ?, ?, ?, ?, 1, ?, ?, 1, 1, ?, ?, ?, ?, ?, ?, ?, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, 1, ?, ?, ?,
1888 ?, ?, ?, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, 1, ?, 1, ?, ?, ?, ?, ?, 1, ?, ?, ?, 1, ?, ?, ?,
1889 ?, ?, ?, ?, 1, 1, 1, 1, 1, ?, 1, ?, 1, 1, ?, ?, ?, ?, ?, 1, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?
```

Figure 5: Data Matrix

3.3 Apriori frequent set mining example

The following example demonstrates the use of Apriori algorithm for the construction of the classification model applied to the identification of suspicious applications on Android devices. Lets consider that we have 8 different applications that are requesting a set of permissions (Table 2):

Application ID	Set of permissions
1	internet, write_storage, read_sms, global_storage, read_calendar
2	internet, read_calendar, write_contacts, camera, read_sms, global_search
3	call_phone, read_calendar, write_contacts, camera, internet
4	read_calendar, write_contacts, write_storage, read_sms, global_search
5	read_calendar, write_contacts, camera, read_sms, internet
6	global_search, write_contacts, camera, read_sms
7	global_search, write_contacts, write_storage, read_sms
8	global_search, write_sms, vibrate, install_package

Table 2: The example of applications with their permissions

Lets suppose the minimum support count required for this dataset is 3 (itemset must occur in 3 or more permission lists).

First, generate the set L1 of frequent 1-itemsets. Since Apriori performs bottom-up approach, in the first iteration every item itself is a candidate for itemset. The set L1 will consist of all 1-itemsets (Table 3) that a support count equal or greater than min. support. The dataset is scanned to calculate support for all items and then only those items that satisfy min. support threshold are chosen.

Itemset	Support
{internet}	4
{write_storage}	4
{read_sms}	6
{global_search}	6
{read_calendar}	5
{write_contacts}	6
{camera}	4

Table 3: The set L1

The support vibrate, install_packages and call_phone is less than 3, therefore they are not present in L1.

As a second step all frequent 2-itemsets are generated. The set of frequent 2-itemsets L2 is discovered by joining the set L1 calculated in the previous step with itself. The result of such join operation is a set of candidate itemsets for L2. The dataset is scanned to determine the support count for each of candidate 2-itemsets. L2 will consist of candidates with support equal of greater than minimum support (Table 4).

Itemset	Support
{internet,read_sms}	3
{internet,read_calendar}	4
{internet, write_contacts}	3
{internet, camera}	3
{write_storage, read_sms}	3
{write_storage, global_search}	4
{read_sms, global_search}	5
{read_sms, read_calendar}	4
{read_sms, write_contacts}	5
{read_sms, camera}	3
{global_search, read_calendar}	3
{global_search,write_contacts}	4
{read_calendar, write_contacts}	4
{read_calendar, camera}	3
{write_contacts, camera}	4

Table 4: The set L2

The generation of the set of candidate 3-itemsets, is taking advantage of the Apriori Property. Apriori Property states that all subsets of a frequent itemset must also be frequent. The use of Apriori Property lets to avoid a heavy computation due large candidate itemsets. For example: lets take an 3-itemset internet, read_sms, global_search. The 2-itemsets are: internet, read_sms, internet, global_search, read_sms, global_search. The itemset internet,global_search is not member of L2 and violates Apriori Property, so the 3-itemset won't be added to L3. The set L3 is discovered by joining set L2 with itself, eliminating candidate 3-itemsets that violate Apriori and then checking remaining candidates against min.support threshold (Table 5).

Itemset	Support
{internet, read_calendar, write_contacts}	3
{internet, read_calendar, camera}	3
{write_storage, read_sms, global_search}	3
{read_sms, global_search, read_calendar}	3
{read_sms, global_search, write_contacts}	3
{read_sms, write_contacts, read_calendar}	3
{read_sms, write_contacts, camera}	3
{read_calendar, write_contacts, camera}	3
{write_contacts, camera, internet}	3

Table 5: The set L3

The set L4 is discovered by joining set L3 with itself, eliminating candidate 4-itemsets that violate Apriori and then checking remaining candidates against min.support threshold (Table 6).

Itemset	Support
{internet, read_calendar, write_contacts, camera}	3

Table 6: The set L4

The algorithm terminates here, because there are no 5-itemsets that would satisfy min.support threshold.

The itemsets of all length that have satisfied the chosen support threshold form a classification model – the basis of the further classification of unknown Android Application classification. New coming application is categorized by comparing the set of security permissions it declares against created classification model. For sake of classification correctness, the applications assigned to the training dataset are all proven to be not anyhow malfunction, so that all the itemsets of the classification model are not suspicious too. For the same purpose the support threshold was chosen very low, only 9%, which means that certain itemset has to occur in 9 different applications. In such a way much more “safe” subsets can be discovered what increases also the efficiency of further analysis. If the candidate application contains a set of security permissions that is not present in the classification model, it receives the “suspicious“ label. In case when only the subset of security permissions matches with the itemset from classification model, the analyzed application is defined as “possibly suspicious”. The application is assigned to “not suspicious” class only when it fully matches one of model itemsets.

However the classification process is semi-automated as it does not determine clearly the maliciousness of a certain application, it simply signals that some applications could require additional specific investigations. Such approach was chosen due to the fact that there are many applications which make use of security permissions related to the sensitive data in a good way. Consider for example Google Gmail application that has access to the user’s email accounts credentials or save the attachments on the SD card. In such cases it is very difficult to identify if the application actually misuses sensitive data, or in the contrary brings benefits to the user. As these particular case require some additional examination the final decision about the suspiciousness level is left to the analyst.

4 Suspicious Application Detection Algorithm

The Suspicious Application Detection Algorithm is the algorithm that was created for the classification of Android applications, relying on the previously built classification model. It consists of frequent itemsets of different size. The Detection Algorithm takes as input the set of security permissions of the analyzed application and outputs the class label (Figure 6). More explicit flowchart can be found in the Appendix (figure 15) The algorithm is based on so called *level-by-level subset creation and pruning*. As already mentioned before, the classification label for the application is assigned by matching its set of permissions with the sets in the classification model. Therefore, as a first step, the input set of security permissions is compared with collection of itemset. If the initial itemset has a match, it means that it is safe, therefore the “non suspicious” class is assigned to the application, and algorithm terminates. However, if on the top level no match was found, it could happen that some items in the are initial setsuspicious. In order to identify these items, the algorithm proceeds level by level in a tree-wise manner; it generates all subsets of initial set of security permissions, until the first subset with a match is found. According to the Apriori Property, such a matching itemset with all its subsets can be pruned. We have considered that all the itemsets of the classification model are non-malicious, therefore the match means that the analyzed set does not contain any elements that cause suspiciousness, so also none of its subsets will. Such a pruning strategy significantly minimizes the search space and the complexity of the algorithm. After the pruning, the same steps are performed of the remaining set of security permissions, until the subset has less than 2 elements or no more matches in the classification model are found. If the final set is empty - the application is considered to be “possibly suspicious”, meaning that it might have a combination of functionalities belonging to different classification profiles. However, if the final set contains some permissions, all of them are identified as suspicious and the application is assigned to the class with the label “suspicious”.

4.1 The use case

In order to show a use case of the described Suspicious Application Detection Algorithm, we assume that an application hides its steganographic functionalities, pretending to be a simple image editor. This application, called ANSAN (Apparently Non-Suspicious Application Name), is able to store a set of sensitive data such as secret contacts into images, for instance using collage steganography [41]. For example, a criminal may use ANSAN to hide contacts of other criminals, without storing them in the SIM card and any phone call to these contacts should be managed via ANSAN, to bypass traces left in the regular contact list and recent call list. This situation seems not the easiest to detect with traditional approaches. As ANSAN manipulates images, it shall be able to write inside images stored on the external memory card – the typical location for pictures. Of course, to be able to deal with secret contacts, the application ANSAN shall provide a way for starting phone calls directly without requiring the user to save secret contact information on the phone memory. For this purpose lets consider that the application uses the following permissions: `write_contacts`, `call_phone`, `write_external_storage` and `camera`

It is obvious, that only a subset of security permissions is actually related to the image manipulation, namely the permissions `write_external_storage` and `camera`. Moreover this subset is proven to be not suspicious as it is present in the classification model. This leads to the conclusion that the remaining permissions `write_contacts`, `call_phone` are the ones that do not match the usual patterns of the applications that operate images and thus cause the “suspiciousness”.

Step 1. To identify the malicious items first the full set is compared with the elements of the classification model. As already mentioned, the set of permissions used by ANSAN is not typical for the applications with such a functionality, therefore no match will be found. Then, the following steps has to be executed.

Step 2. Generate all the subsets of length one unit smaller than the length of the initial set, in this case 3. The created subsets are then $:\{ \text{write_contacts}, \text{call_phone}, \text{write_external_storage}\}, \{ \text{write_contacts},$

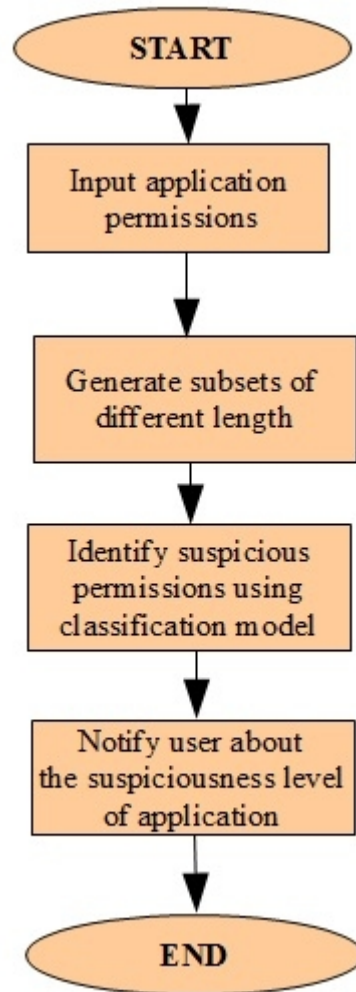


Figure 6: Main steps of Suspicious Application Detection Algorithm

`read_contacts, camera`}, `{call_phone, write_external_storage, camera}`, `{write_contacts, write_external_storage, camera}`. Since each itemset contains or the permission `write_contacts`, or `call_phone`, or both of them, no match is found also on this level.

Step3. Generate all the subsets of length 2. The new itemsets are: `{write_contacts, call_phone}`, `{write_external_storage, camera}`, `{write_contacts, write_external_storage}`, `{write_contacts, camera}`, `{call_phone, camera}`, `{call_phone, write_external_storage}`. It is possible to see that on this level the itemset `{write_external_storage, camera}` that corresponds to a pattern in the classification model appears. According to the Apriori rule it can be safely eliminated from the initial set of permissions.

Step 4. Execute the detection on the remaining security permissions. After the pruning the following itemset has to be analyzed: `{write_contacts, call_phone}`. As discussed above, both permissions in this set do not correspond to any patterns of the image editing applications and in addition the itemset of size 2 is the smallest considered unit, meaning that the subset cannot be divided anymore. The subset `write_contacts, call_phone` is identified as suspicious, so are the permissions `write_contacts` and `call_phone`.

The Detection Algorithm clearly identifies the ANSAN as a malicious application and explicitly indicates

the elements that are not typical for certain type of applications and may cause a danger. Such a indication allows both the user and the forensics specialist to understand better the actual behavior of the application and take appropriate decisions on how to proceed further. In the case of the Android device user ANSAN can be simply removed from the smartphone. The forensic examiner in his turn has a good hint on what kind of sensitive data might be exploited by the application. In this particular case from the scope of the ANSAN and the suspicious permissions detected it is clear that some contacts might be hidden in the images and that the application might have the possibility to call certain numbers. Having all this information, the forensic examiner can make use of additional forensic tools and deeper to investigate the suspicious application.

4.2 The pseudo-code

The following pseudo code give the explicit and detail explanation of the Suspicious Application Detection Algorithm execution (listing 2).

```

1      Input: app_permissions(P_1, P_2, ..., P_n), N – number of permissions
2      Compare(app_permissions, CM), where CM is a Classification Model
3      IF match found
4          Assign the label the same as the as the label application to which
           found match belongs
5          TERMINATE;
6      ELSE
7          N=N-1;
8          IF N<2
9              Check if app_permissions is empty
10             IF is empty
11                 Declare application as ‘‘possibly suspicious’’
12                 TERMINATE;
13             ELSE
14                 Declare the application as ‘‘suspicious’’
15                 Report the remaining permissions as the suspicious ones
16                 TERMINATE;
17         ELSE
18             Generate subsets of size N
19             S={s_1, s_2, ..., s_m}
20             IdentifySuspicious (s_1);
21
22
23     The function IdentifySusoicious(s_k)
24     Input: A subset s_k, at the first iteration k=1
25     Compare(s_k, CM)
26     IF match found
27         Delete elements of s_k from app_suspicious
28         IF |app_suspicious|<2
29             IF is empty
30                 Declare application as ‘‘possibly suspicious’’
31                 TERMINATE;
32             ELSE
33                 Declare the application as ‘‘suspicious’’
34                 Report the remaining permissions as the suspicious ones
35                 TERMINATE;
36     ELSE

```



```

37         IdentifySuspicious ( app_permissions );
38     ELSE
39         Check if s_k is the last element of S
40         IF is the last
41             N=N-1;
42             IF N<2
43                 Check if app_suspicious is empty
44                 IF is empty
45                     Declare application as ‘‘possibly suspicious’’
46                     TERMINATE;
47                 ELSE
48                     Declare the application as ‘‘suspicious’’
49                     Report the remaining permissions as the
                        suspicious
50                     TERMINATE;
51             ELSE
52                 Generate subsets of size N
53                 S={s_1 , s_2 , ... , s_m}
54                 IdentifySuspicious (s_k) , k=1;
55     ELSE
56         k=k+1;
57         IdentifySuspicious (s_k)

```

Listing 2: Suspicious Application Detection Algorithm - Pseudo Code

4.3 The complexity

The complexity of the algorithm strongly depends on the application’s presence in the classification mode, the number and type of permissions the it contains. The most expensive operation that actually decides the computational complexity of the Detection Algorithm is the function IndetifySuspicious which includes the subset generation and the itemset’s comparison with the classification model elements. However, the comparison function for any kind of itemset is defined as a single access to the database and its complexity does not increase together with the input. Therefore it does not have an impact on the overall complexity of the algorithm. Generally, the estimation of the complexity depends on how many times the function IdentifySuspicious will have to be executed.

The best case of the algorithm occurs when the application is already present in the classification model, meaning that it was analyzed previously and thus all the information regarding the suspiciousness in known. The label then can be directly retrieved from the classification model without any additional steps. Yet another situation that has low complexity is when the analyzed application does not contain any permissions ($N=0$) and it can immediately be declared as not malicious one. In both cases the complexity of the algorithm is $O(N)$.

In the case when the application’s set of security permissions does not have a match in the classification model, meaning that the application is either ‘‘suspicious’’ either ‘‘possibly suspicious’’, the complexity of the algorithm is lower if the malicious permissions are identified on the high level of the hierarchy of all possible subsets. The Hierarchy starts with full set of permissions on top, and goes down level by level decreasing the length of subsets and generating all itemsets of that length. Lets suppose that the set consists of the following permissions: P_1, P_2 and P_3 , where P_2 is suspicious and the itemset P_1P_3 is present in the classification model. The suspicious permission will be identified on the level 2, after checking only 4 different combinations of permissions. Generally, for N permissions assuming that all not suspicious itemsets of the set are included into classification model and there is only one malicious permission in the set, N combinations will have

to be generated and checked. The generation of N combinations knowing that 1 permission is suspicious implies selecting 1 combination from N , which is $C(N, N - 1)$, thus the IdentifySuspicious will be executed N times. The number of combinations that have to be checked during the runtime of the algorithm depends on the number of suspicious permissions in the initial set (this all applies to the case when all not suspicious subsets of analyzed set are present in the classification model). For N - number of permissions, given that 2 permissions are suspicious we would have to check

$$C(N, N - 2) + 1 = \frac{N!}{(N - 2)!(N - (N - 2))!} + 1 = \frac{n(n - 1)}{2} + 1 \quad (3)$$

combinations, as not only all the combinations of size $N - 2$ have to be checked but also the combination of two suspicious permissions itself.

The general case for the m suspicious permissions when $m > 1$ is at most:

$$\sum_{k=1}^m C(N, N - k) \quad (4)$$

different combinations to check. Therefore the average computational complexity of the Detection Algorithm is exponential in the number of suspicious permissions - $O(2^m)$.

The worst case for the algorithm is when the suspicious permissions are identified on the lowest level of the hierarchy, then there will be generated:

$$\sum_{1 \geq m \leq N} C(N, N - m) + 1 = 2^N \quad (5)$$

This implies that the overall computational complexity is then exponential $O(2^N)$. The typical scenario of the worst case is when the itemsets of the higher levels in the hierarchy ($length(itemset) > 2$) are not known to the classification model. Another scenario is when the number of suspicious permissions m in the application is equal to $N - 2$. Having the condition that all not suspicious itemsets of the application are present in the classification model, the number of suspicious permissions in the analyzed set identifies on which level of the hierarchy of the combinations (tree of combinations) the algorithm will identify suspicious items and the analysis will terminate.

To illustrate the worst case of the algorithm we have performed the analysis of the application that has declared 19 permissions. All the subsets down to the level 2 were not known to the classification model. Therefore the 665057 combinations of different length had to be checked. The over time of the analysis execution was about 18 hours on the system with 4GB of RAM and the processor Intel Core Duo 2.1GHz

However the worst case of the Detection Algorithm occurs rarely, as the total number of permissions in the application typically does not reach 20. Most of the applications declare to use between 1 and 10 permissions (figure 7). Only a very small number of applications accesses the features of the system with more than 20 different permissions. Typically, the applications with a big number of permissions are the standard ones that are provided together with the mobile device or the applications that provide system management facilities or include Google services. The number of malicious permissions in one application is typically between 1 and 5.

Moreover, whenever a new application is analyzed by the algorithm, all its features such as permission set, all possible itemsets and level of suspiciousness are added to the classification model. Due to such a constant update of the model than more applications are discovered than lower becomes the computational complexity of the further analysis.

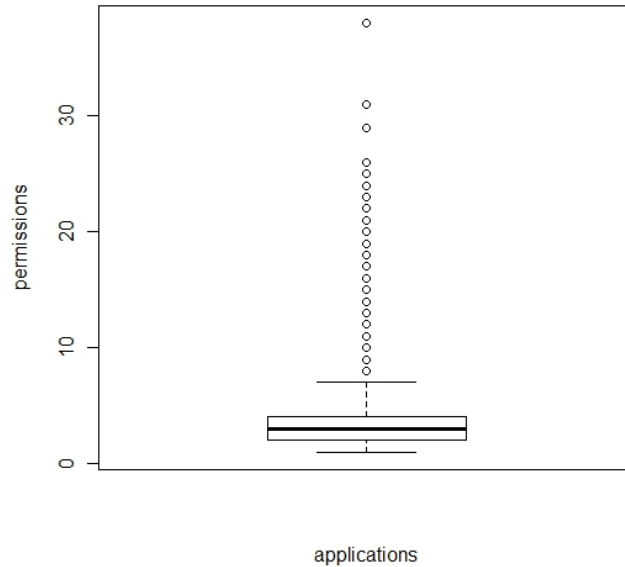


Figure 7: The distribution of permissions in the classification model

4.4 Evaluation of Detection Algorithm

When dealing with algorithms, two main aspects need to be taken into consideration: how correct are the results, and how quickly it performs a given task.

To evaluate the Suspicious Application Detection Algorithm we have chosen two kinds of tests, namely the correctness test that refers to identification of false positives and false negatives, and the execution time test. For the execution of the experiments, a sample of 450 different applications was created; it was collected from the real mobile devices running Android OS. Correctness and response time strongly depend on the number of permissions that an application uses. Therefore, the data sample has been divided into subsets, taking into account the number of permissions that applications require. 3 different application groups were created for the tests:

- applications with 1-5 permissions;
- applications with 5-10 permissions;
- application with more than 10 permissions.

Each group contains 150 application with the corresponding number of permissions. All applications are not present in the classification model, so that the full analysis has to be executed on all of them. The applications that do not require any permissions were not considered, simply for the fact that they have no impact neither on the execution time nor on the correctness of the algorithm. They could still use native API to deceive Android security framework. However, this case is not covered in this thesis.

4.4.1 Correctness evaluation

The task: To identify how many false positives and false negatives occur on average during the analysis process.

Here **false positive** is defined as an application that does not have malicious intentions but nevertheless is assigned to the class of suspicious applications. The **false negative** is the malicious application that has been identified as not harmful to the system on which it is running.

The dataset: the same 3 groups of applications are taken into consideration. In order to test the algorithm for the appearance of false positives 50 items of each group were substituted with 50 applications that were already present in the classification model. However, the names of the applications were changed, so that the sets of permissions that they define would have to be analyzed again. Moreover, to identify false negatives to each group there have been added 25 suspicious applications. Due to the fact that there are not many applications with the confirmed suspiciousness, the sets were generated artificially using the permissions of applications known to be malicious, such as MobileSpy or MobileStealth [33]. From 25 generated applications 10 of them were added to the classification model, however, also with the changed names. In this case it is possible to check how the algorithm classifies the applications that were already analyzed once but without having the additional information, such as a real application name.

	Total number of apps	Correctly classified apps	Not correctly classified apps
Not suspicious apps, not present in DB	100	70	30
Not suspicious apps present in DB	50	50	0
Malicious app not present in DB	15	15	0
Malicious apps present in DB	10	10	0

Table 7: The correctness test for the applications with 1-5 permissions

	Total number of apps	Correctly classified apps	Not correctly classified apps
Not suspicious apps, not present in DB	100	63	37
Not suspicious apps present in DB	50	49	1
Malicious app not present in DB	15	15	0
Malicious apps present in DB	10	9	1

Table 8: The correctness test for the applications with 6-10 permissions

The results

False Positives. The number of false positives in the set of applications that were already present in the classification model and renamed is very insignificant. In all 3 groups such application were classified with the correctness almost of 100%. The occurrence of the outliers can be explained by some inconsistencies in the

	Total number of apps	Correctly classified apps	Not correctly classified apps
Not suspicious apps, not present in DB	100	43	57
Not suspicious apps present in DB	50	48	2
Malicious app not present in DB	15	15	0
Malicious apps present in DB	10	9	1

Table 9: The correctness test for the applications with more than 10 permissions

classification model introduced during its creation. The absence of additional information did not have any impact on the correctness of the results as subsets of permissions were already present in the the classification model and the analysis is performed based on them. However the significant percentage of false positives has occurred during the analysis of the new not suspicious applications. It is possible to see that the number of outliers depends on the size of the permission set of the applications. In the group of applications with the 1 to 5 permissions about 30% of false positives has been noticed (table 7). The set of the applications with 6 to 10 permissions contained about 40% of false positives (table 8). In the last group more than half applications have been identified as malfunction when they are not. Such a number of false positives indicates the correlation with the number of permissions in the analyzed application (table 9). Than bigger the set of permissions than bigger the probability that it will contain at least one permissions that won't fit any of the frequent patterns.

Another reason for the such significant number of false positives it the ambiguity of the permissions used by the applications. The same permissions that allow the access to sensitive data can be very useful in the context of one application and cause the danger or misuse the sensitive information in the context of another one.

False Negatives. In all 3 groups only couple of false negatives have occurred. Both, the malicious applications that were included into classification model and the ones that were not, show similar results. The applications with the malicious intentions are classified correctly in most of the cases not depending on number of permissions declared by the application. Nevertheless it is impossible to draw the precise conclusions from a sample of 25 malicious applications, it can be noted that the classification of suspicious items tends to be more correct than the classification of non malicious items.

Summarizing the results of the correctness analysis of Suspicious Application Detection Algorithm it can be stated that the algorithm is false positive oriented. This means that some applications with non malicious intentions can mistakenly identified as harmful to the system. However, as already mentioned before the algorithm is semi-automated, meaning that it only gives a suggestions on which applications could require additional specific investigations.

4.4.2 Performance evaluation

The task: Measure the execution time of the Suspicious Application Detection Algorithm.

The system parameters: 1GB RAM, processor Intel(R) Xeon(R) CPU X5650 2.67 GHz, cash size - 12288KB.

The dataset: 3 sets of applications grouped by the number of permissions.

In order to make more precise performance estimation on each collection of the applications the test has been executed multiple times. The graphs (figure 8) present average execution time for each group of 150 applications depending on how many times the set was run. Additionally, the comparison of the average time performance of all 3 groups is made.

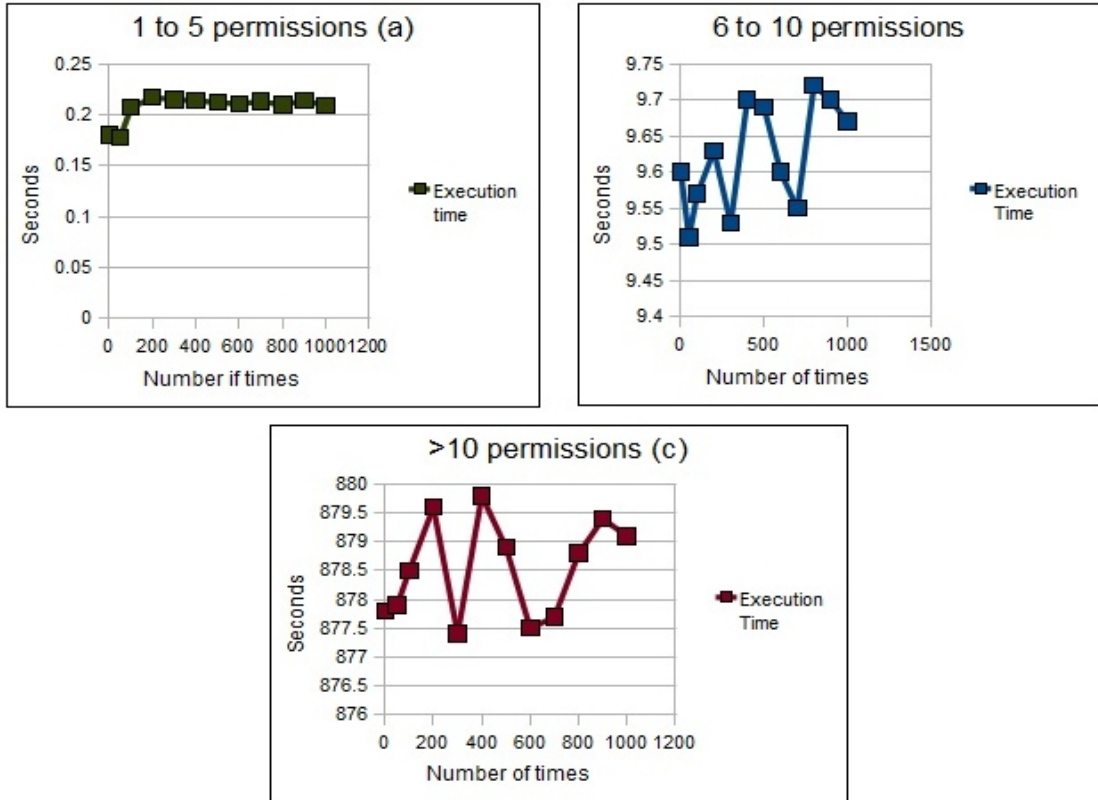


Figure 8: Execution time graphs form apps with 1-5 permissions (a), with 6-10 permissions(b), with more than 10 permissions (c)

Results: from the graphs representing the efficiency of the algorithm by means of execution time (figure 8) it's clearly seen that the number of permissions declared in the applications has a big impact on the performance. As expected, the analysis of the applications with smaller set of permissions is completed much faster than the analysis of the applications from two other groups. However, with the increasing number of execution performed at once, the time needed to analyze one application changes only very slightly, therefore it can be concluded that the amount of executions does not influence the efficiency of the algorithm.

The comparison graph (figure 9) indicates a very big difference between the average time performance of 2 first application groups and the applications with more than 10 permissions. The analysis of the group application that uses no more than 5 permissions takes 30 sec. The group with 6 to 10 permissions on average requires about 20 min. However the analysis process of the group of applications that tend to declare 10 and more permission completes only after about 22h (The worse case scenario is not taken into consideration). Such a big gap is caused by the fact that number of subsets that needs to be checked against the classification model grows exponentially with the number of permissions in the set.

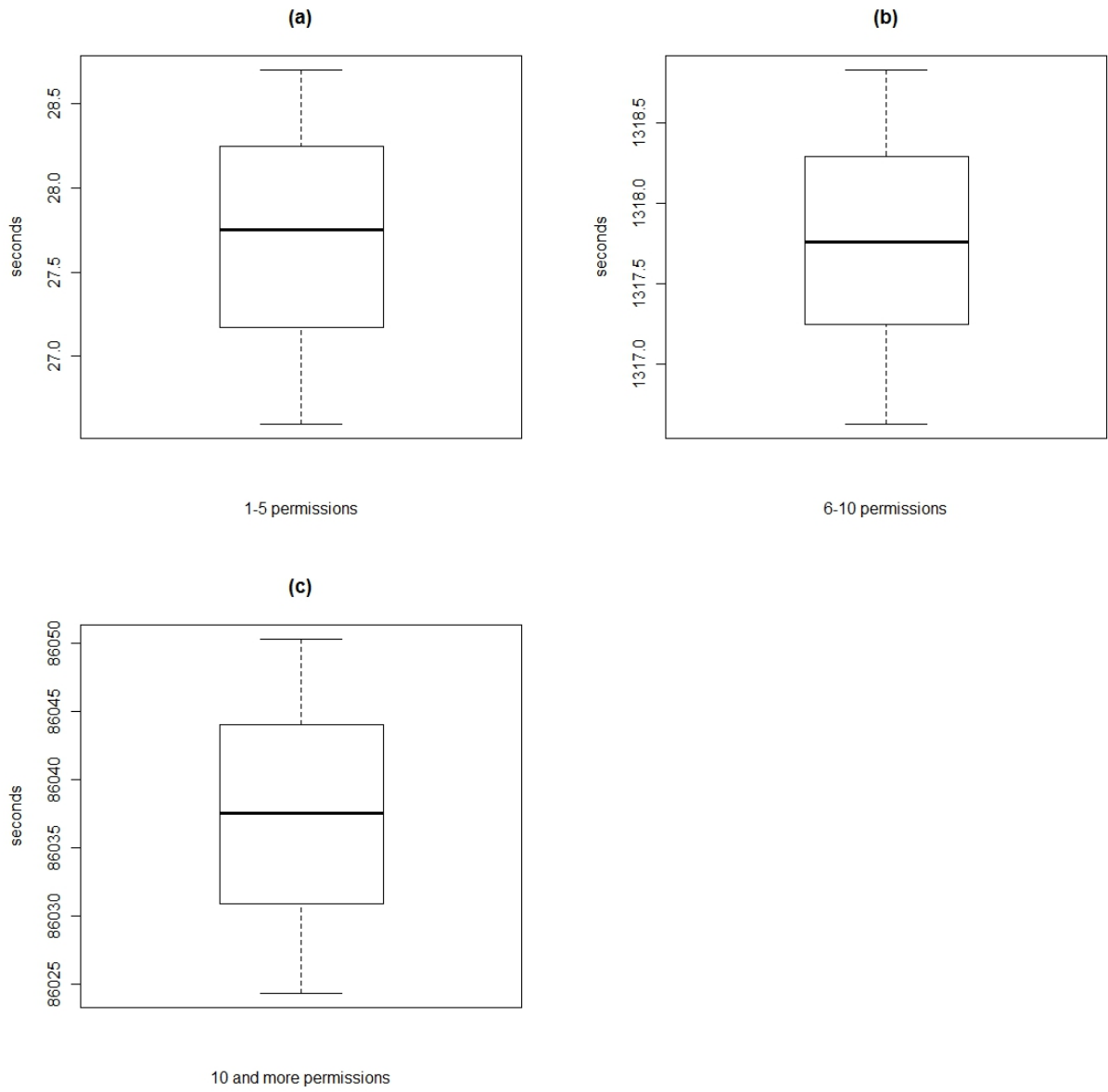


Figure 9: Execution time comparison

5 Proposed System Prototype

This chapter presents the design and development process of the software components that demonstrate the use of the Suspicious Application Detection Algorithm described in the previous section. Moreover the chapter includes the description of most common use case of the created system from the point of view of the professional forensics examiner as well as the Android Smartphone user.

5.1 Outline

The development process has started with the definition of set of system requirements, main functionalities and the analysis of possible development approaches. After the completion of these steps the appropriate system architecture that fulfills all the stated requirements as well as implements all the defined functionality has been designed. During the development phase the following technologies were used: PostgreSQL Database Server, Android Application Framework, Apache CXF web services and Java Servlets. For the implementation of the Suspicious Application Detection Algorithm the Java Programming Language was chosen.

5.2 Proposed system architecture

The high level system architecture proposed for the appropriate demonstration of the usage of Suspicious Application Detection Algorithm (figure 10) consists of several software components that form the fully integrated system. As the system is intended to assist two different types of users, namely the professional forensic analysts and the typical Android Smartphone users, at the software level it has two components. The web based application *Web Forensics* serves as a helping tool for forensics examiners and allows to perform the malicious applications detection phase on the chosen data source, giving an explicit report about the applications contained in the source, their maliciousness level and the rights that actually are causing the suspiciousness. Moreover the *Web Forensics* offers some other functionalities such as maintaining the database of the applications and their characteristics with the newly discovered ones, viewing the list of suspicious applications and editing the level of suspiciousness of certain elements. In addition to the a web-based application for forensic professionals an Android *AForensics* application that in its turn communicates with AForensics Webservice and delivers the shorten version of detection report to the mobile device end user was created. *AForensics* allows the user to collect the data about the applications installed on the device and submit them for the analysis. As the result the short report that contains only the names of suspicious application, if such were found, is presented. In such a way the smartphone user is warned about the possible threats and may take a decision to remove the applications that were signaled as dangerous or continue to use them.

5.3 AForensics for Android

As remarked AForensics represents a service intended for smartphone users that enables to run the check for suspicious applications on the device. However, in addition the application serves as a data collection tool for the forensic professionals. In this particular case during the investigation process AForensics shall be installed on the analyzed device and uninstalled after that necessary data has been collected.

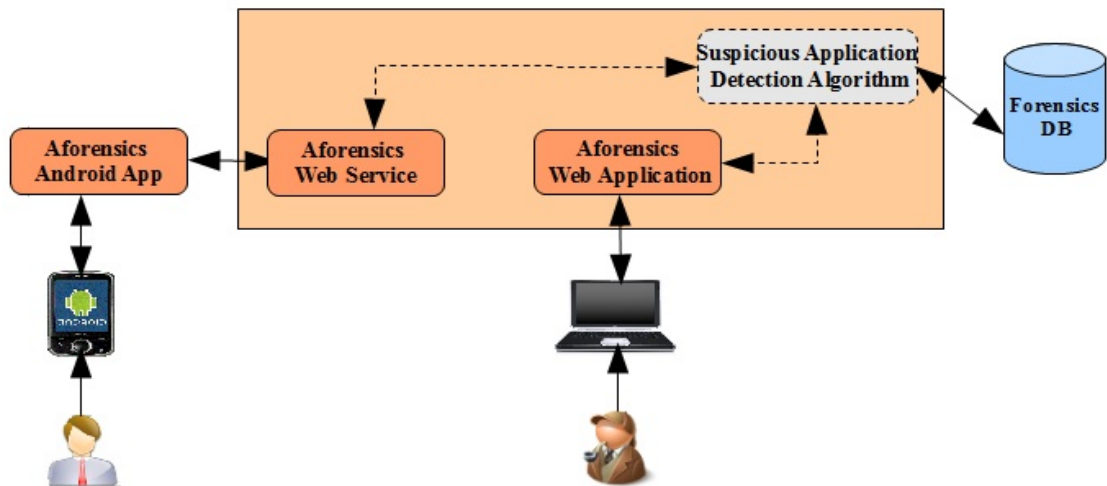


Figure 10: System Architecture

5.3.1 User interface

The tool presents itself as menu based graphical user interface enabling the navigation between the offered features: Data Collection, Final Report, Instructions and About (figure 11a).

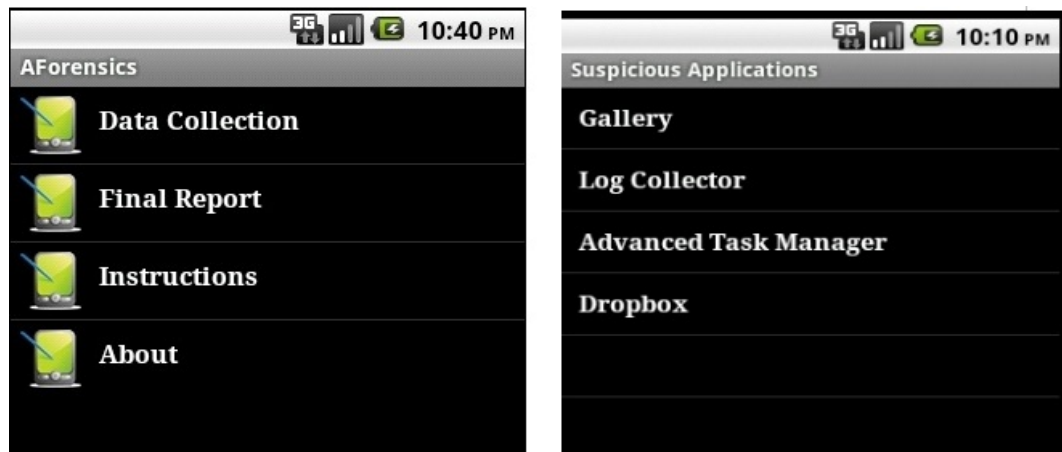


Figure 11: AForensics main menu(a), AForensics final report representation (b)

Data Collection

The first user interface launches the data collection that runs as a background process. During this phase the application looks up the third-party applications installed on the device and gathers permissions that the applications request from Android OS and permissions declared by applications. Moreover, there is being collected the complementary information like services, broadcast receivers and content providers that each application could have defined. The whole data is organized using specially for this purpose defined structure

and saved into XML file (listing 3). The structure includes the details about the application itself such as name, package name and version, as well as the set of permissions that has been declared. The generated file is then stored on the SD card of the device and also it is sent to the server for the examination. The user is notified about the ending of the data collection process with the short message (figure 12)

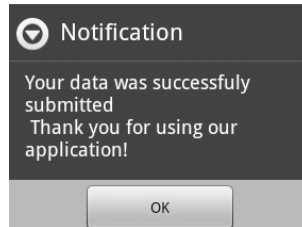


Figure 12: User notification

In the case of forensic professional analyst the created source can be used for the further analysis. The source can be either directly taken from the server or be retrieved for the system's SD card applying some traditional data forensics methods.

```

1 <application name="Earth" package="com.google.earth" version="1.1">
2   <requested-permission permission="android.permission.
3     ACCESS_FINE_LOCATION" />
4   <requested-permission permission="android.permission.
5     ACCESS_COARSE_LOCATION" />
6   <requested-permission permission="android.permission.INTERNET" />
7 </application>

```

Listing 3: The structure used for to refine the collected information

Final report

After the data has been sent to the server, the analysis process of the committed source is started. The communication between AForensics and the server is supported via the AForensics Webservice that performs the transfer of collected data and forms the final report presented to the end user. The final report is available to the Android device user as soon as the analysis is terminated. The application offers a possibility to launch the analysis multiple number of times and in this case only the latest report is being delivered. In order to make it simple and understandable for the users, the report contains only the names of the applications that have been identified as a malfunction (figure 11b). The phone owner is then free to decide what further steps to take - to continue using the suspicious applications or remove them from the device.

5.4 The Web Forensics application

Web Forensics is a managing tool that intends to help the forensics professionals in the investigation process of smartphone running Android OS. The application has been developed using a well known Java Servlet Technology and located on Tomcat6 web server. The tool enables such functionalities as analysis, the view and management of discovered suspicious applications, and the update of the database that stores all information about analyzed applications. In addition there has been implemented the reporting system that provides the examiner with the specific details of conducted analysis has been implemented.

5.4.1 User interface and functionality

The graphical user interface of Web Forensics does not differ from the interface of simple web application. The one level on top positioned menu allows the user to switch between the pages of the application that represent the different features of the system. There are 5 different menu sections: Analysis, Maintenance, Suspicious Apps, Update Suspicious and Downloads (figure 13).

Analysis

The Analysis section provides the user with the possibility to submit the collected data for the actual analysis. The data has to follow the predefined structure (listing 3) and be passed as an XML file. The file with such a structure can be retrieved directly from the investigated phone by taking the advantage of the Android AForensics application. After the data has been submitted to the web application, the set of applications is processed by the Suspicious Application Detection Algorithm and the final report is delivered to the examiner. Moreover the results of each performed analysis are stored in the temporary location so that they could be accessible for the maintenance process.

Maintenance

Since the collection of discovered malicious and non malicious applications has to be constantly updated in order to have consistent and more efficient analysis results, the data retrieved during each algorithm iteration must be added to the classification model. The maintenance process operates with the same kind of data source as the analysis phase. The labels of applications from the submitted source are compared to those stored in the temporary location and if match is found the application and its other characteristics are added to the classification model. Such a management mechanism eliminated the possibility of adding the inconsistent data into the model as well as leaved the forensic professional freedom to choose which applications to include into the collection and which ones not.



Figure 13: Web Forensics, Menu

Suspicious Apps

As already mentioned before, the overall system is semi-automated, meaning that the final confirmation of the application's suspiciousness level is the task of the user, in this case of the forensics analyst. The "Suspicious Apps" in section simply presents the list of the Android applications that have been confirmed as the ones that may perform malicious activities and conduct frauds. Besides the name of the application the description includes the name of its package and the version.

Update Suspicious

"Update Suspicious" is a suspicious application management section that views all the confirmed and discovered but not yet confirmed applications and enables that editing mode.

Downloads

In the “Downloads” section the version of the Android AForensics is made available to the public.

5.4.2 The reporting mechanism

The final report is delivered to the forensics examiner immediately after the termination of the analysis phase. The output of the Suspicious Application Detection Algorithm is retrieved from the temporary location and viewed to the user in a form of a list of all applications with their suspiciousness levels and the set of identified malicious elements if such are present (figure 14). For the sake of the visual representation the level of suspiciousness is defined with the different colors. Red square signals the highest level of suspiciousness and warns the analyst that some additional investigation steps has to be taken. The orange in its turn identifies the applications with a medium level of suspiciousness, which means that the application is not necessary suspicious but may simply combine two different activities as for example the image editor that is able to take pictures and send them via SMS. The green color is assigned to applications that do not contain any malfunction elements.

Forensics Report		
Application Name	Suspiciousness	Suspicious Permissions
File Manager		KILL_BACKGROUND_PROCESSES, INSTALL_SHORTCUT
Clock Widget >		No suspicious permissions
People Widget		READ_ACCOUNT, WRITE_ACCOUNT
News and Weather		No suspicious permissions

Figure 14: Web Forensics, report

6 Related Works

As already discussed before, the Android OS forensics is relatively new and not a well researched field. However, there already exist some applications that deal with the sensitive data retrieval from Android OS.

6.1 Open Source Android Forensics application

The Open Source Android Forensics application is the small software that extracts the basic data from the mobile devices running Android OS. The initial version of the application is able to retrieve such information as call logs, browser history, contacts and SMS messages. Acquired data are stored in the CSV file on the SD card of the investigated device and can be downloaded onto PC for the further investigations. The application has been released under GNU General Public License v3 (GPL) allowing developers to build additional plugins and improve the functionality of the tool. Moreover it is freely distributed for the law enforcement officers that deal with mobile forensics issues [31].

6.2 Permission based technology by SMobile

In the report published in June 2010, the SMobile has presented the idea of permission-based malicious application detection approach [33]. Despite the fact that the approach presented by SMobile has commonalities with the methodology described in this thesis report, we confirm that our research in this particular area has started before the SMobile's publication. Moreover the first submission to the conference of the initial research idea is also dated earlier than the report of SMobile.

In their publication the SMobile clearly states that signature-based detection mechanism is not able to identify all the threats on mobile devices and alerts the importance of permission - based techniques. However no details on how actually the methodology uses the information about application's granted permissions are provided. The publication identifies only the set of 23 permissions that allow the applications to access sensitive services and user's data. In addition, the statistics of metadata collected from Android Market is presented, namely the statistics of 68% of applications that are available for download. According to this statistics the permissions that grant the access to the sensitive information are declared by each fifth application. Moreover 29 applications were found to declare the same set of permissions as known Android malware.

7 Conclusions and future work

The fascinating evolution of mobile technologies, especially in consumer market, has increased the amount of sensitive data stored on portable devices and thus their importance in the forensics investigation. As shown in chapter “State of the Art in Digital and Mobile Forensics”, the number of trends involving mobile devices has grown rapidly in the past years. Despite many technologies and methods were developed to support the professionals in forensics when analyzing small scale devices, these are far behind when it comes to the investigation of smartphones. In the forensics tools market, there is a lack of software solutions that support the investigators in their activities while dealing with innovative platforms, as for instance on Android Handsets.

For this purpose we have developed a semi – automated methodology that will aid in the identification of “suspicious” applications on smartphones running Android OS. With “suspicious”, in this context I refer to applications that might hide, steal or in any other way misuse user’s sensitive data or exploit sensitive services. The technique relies on the classification of Android applications in “suspicious”, “possibly suspicious” and “not suspicious” groups, according to the set of permissions granted to them and taking into the consideration the type of sensitive information that is handled by them. The classification base model was built applying known data mining technique – the Apriori Frequent Set Mining and includes more than 13000 safe applications that manage user’s personal data. The sample was chosen from around 40000 applications hosted on the Android Market and collected with AppAware, a specific tool for Android OS.

As a result of the described methodology, the Suspicious Application Detection Algorithm was implemented. It performs all the classification steps relying on the base model. Additionally, in order to proof the efficiency of developed technique, multiple experiments have been executed. The main focus of the tests was on the overall correctness of the classification process and on the average execution time of applications.

The correctness of the algorithm has been measured by the number of false positives and false negatives that occur during the classification process. The results indicate that the malicious items are classified with the greater accuracy than the applications with non malicious intentions. Moreover it was concluded that the Suspicious Application Detection Algorithm is false positive oriented. This means that some not suspicious applications are mistakenly identified as harmful to the system.

The results of the time execution test have shown that the time performance of the algorithm strongly depends on the size of permission set that the analyzed application uses. The Suspicious Application Detection Algorithm performs best on the applications with the small number of permissions. As the number of permissions in the set increases the execution time grows. There was also noticed the big difference between the time in which the analysis of an application with small number of permissions (up to 5) and the analysis of the application with more than 10 permissions is completed. The difference of on average 14 min occurs due to the exponential growth of the permission combinations that have to be analyzed in order to complete the classification. However, according to our data, Android applications that request permissions generally limit them to 3.

The methodology described within the thesis report at its initial stage was presented at the 4th International Workshop on Computational Forensics (IWCF) in November, 2010. The workshop is intended as a convention of experts in all areas of computational and forensics sciences, to present and discuss recent progress and advances [1]. The presented paper has been evaluated positively and generally received a good feedback.

Beyond the research and development in the field of mobile forensics described in this thesis, other activities can be considered for future researches. With the respect to results achieved during the research on the malfunction application detection in the context of Android mobile forensics, the following points can be considered for the future investigation: the Suspicious Application Detection Algorithm efficiency improvement, usability testing of the AForensics web application from the point of view of forensics professionals and the improvement of malicious application detection service offered to the Android device users.

In order to minimize the execution time and improve the efficiency of the Suspicious Application Detection Algorithm the parallel computing technique could be applied. In this way multiple processing elements could be used to solve the detection problem. On the level $N-1$, when N is the number of permissions, the collection of the permission subsets could be broken into smaller parts. In this way each processing element would execute one branch of the algorithm starting from the single permission set containing $N-1$ items simultaneously with others. In the end, the combined result of all independently running branches will contain all the suspicious permissions of the initially given set, if such exists and the classification label. Additionally, to achieve the best possible performance, the different resources such as a multiple computers with single processors, a single computer with multiple processors, a set of networked computers or specialized hardware could be tested.

Besides the parallel computing approach, different heuristics could be taken into consideration. Such heuristics could help to reduce the runtime of the algorithm by letting to eliminate some subset branches on the early steps of the analysis process.

Regarding the minimization of occurring false positives, new cluster of Android applications known to be not suspicious can be added to existing classification model. Such an addition could not only improve the correctness of the analysis and minimize the number of false positives but also to improve the efficiency of the detection service for the Android device users. In this case the extended classification model would speed up the the response time of detection result provided to the users.

We are also considering to include the outcome of the research directly into AppAware, in order to warn all Android end users about suspicious software before installing it. Moreover, it could be possible to consider to retrieve applications directly from the Android Market, in order to automatize the analysis process as soon as new application or new versions of already evaluated applications are released. At present, this possibility is explicitly prevented by Android Market terms of use.

References

- [1] Di Cerbo F., Girardello A., Michahelles F. and Voronkova S. 2011. Detection of Malicious Applications on Android OS. *Computational Forensics, Lecture Notes in Computer Science*, volume 6540, pp. 138–149.
- [2] Electronic discovery definition. 2008. http://searchfinancialsecurity.techtarget.com/sDefinition/0,,sid185_gci1150017,00.html. Accessed: 27-02-2011.
- [3] Scientific Working Groups on Digital Evidence and Imaging Technology: Combined Master Glossary of Terms. 2009. http://www.theiai.org/guidelines/swgit/swgde/swgde_swgit_glossary_v2-3.pdf. Accessed: 02-10-2010.
- [4] Robbins, J. 1998. An Explanation of Computer Forensics, PC Software Forensics An Explanation of Computer Forensics by Judd Robbins. <http://www.pivx.com/forensics>. Accessed: 02-07-2010.
- [5] Meadaris K. 2006. Grants to help develop ways to improve digital evidence collection, Purdue University.
- [6] United States Department of Justice.2001. Crime Scene Investigation: A Guide for First Responders, National Institute of Justice, N.W.Washington.
- [7] Wasik M. 1989. Law Reform Proposals on Computer Misuse. *The Criminal Law Review* 257.
- [8] Leigh J. Computer Forensics Chain Of Custody. <http://www.articledashboard.com/Article/Computer-Forensics-Chain-of-Custody/1479133>. Accessed: 10-07-2010.
- [9] Jansen W. and Ayers R.2007. Guidelines on Cell Phone Forensics. National Institute of Standards and Technology (NIST), Special Publication 800-101. Gaithersburg 2007.
- [10] Rizwan, A. and Rajiv, V. D. 2009. Mobile Forensics: an overview, tools, future trends and challenges from Law Enforcement perspective, *Communications in Computer and Information Science*, volume 31, Part 7, pp. 173-184.
- [11] Zheng, P., and Ni, L. M. 2006. The Rise of the smartphone. *IEEE Distributed Systems Online*, volume 7, no. 3, pp. 3.
- [12] Carrier B.D. 2006. Risks of live digital forensics analysis, *Communications of the ACM* , volume 49, no. 2, pp. 56-61.
- [13] Butler, J. 2010. Forensic Analysis of Mobile Phones, A white paper on the nature of mobile phone evidence and how to secure it, Geode Forensics Ltd, Lasswade, Midlothian.
- [14] Curran, K., Robinson,A., Peacocke, S. and Cassidy, S. 2010. Mobile Phone Forensic Analysis, *International Journal of Digital Crime and Forensics*, volume 2, no. 1, pp. 15-27.
- [15] Keonwoo, K., Dowon, H., Kyoil, Ch. and Jae-Cheol, R. 2007. Data Acquisition from cell phone using Logical Approach, *Proceedings of World Academy of Science, Engineering and Technology*, volume 32, pp. 29-33.
- [16] Marcel, B., Martien de, J., Coert, K., Ronald van der, K. and Mark R. 2007. Forensic Data Recovery from flash Memory. *Small Scale Digital Device Forensics Journal*, volume 1, no. 1, pp. 1-6.
- [17] Kessler, G.C. 2004. An overview of steganography for the computer examiner. *Forensics science communications*, volume 6, no. 3, pp. 1-29
- [18] Gostev A. 2006. Mobile Malware Evolution: An Overview, Part 1. <http://www.viruslist.com>. Accessed: 12-12-2010.

- [19] Kaspersky Lab. 2008. <http://www.kaspersky.com>. Accessed: 12-12-2010.
- [20] Mobileedit user guide. 2008. <http://www.mobileedit.com/guide.htm>. Accessed 12-12-2010.
- [21] Oxygen Forensic Suite. 2000. <http://www.oxygen-forensic.com>. Accessed 12-12-2010.
- [22] Schmidt, A.D., Schmidt, H. G., Batyu, L.K., Clausen, H., Camtepe, S.A. and Albayrak, S. 2009. Smartphone Malware Evolution Revisited: Android Next Target?, 4th IEEE International Conference on Malicious and Unwanted Software (Malware), Montreal, Quebec, Canada, 2009, pp. 1-7.
- [23] US-CERT, United States Computer Emergency Readiness Team. 2010. Technical Information Paper-TIP-10-105-01, Cyber Threats to Mobile Devices.
- [24] US-CERT, United States Computer Emergency Readiness Team. 2009. Avoiding Social Engineering and Phishing Attacks.
- [25] Halfacree, G. 2010. Banking trojan hits Android. <http://www.bit-tech.net/news/bits/2010/01/11/banking-trojan-hits-android>. Accessed 15-12-2010.
- [26] Higgins, K. J. 2010. Smartphone Weather App Builds A Mobile Botnet. <http://www.darkreading.com/insider-threat/167801100/security/application-security/223200001/index.html>. Accessed: 15-12-2010.
- [27] Irinco, B. 2010 .Malicious Android App Spies on Users Location. <http://blog.trendmicro.com/malicious-android-app-spies-on-users-location/>. Accessed 15-12-2010.
- [28] Baggili, I. M., Mislan, R., Rogers, M. 2007. Mobile phone forensics tool testing: A database driven approach, International Journal of Digital Evidence, volume 6, no. 2, pp. 168-178.
- [29] Ayers, R., Jansen S., Moenner, L. and Delaitre A. 2007. Cell Phone Forensic Tools: An Overview and Analysis Update, National Institute of Standards and Technology Gaithersburg, MD 20899-8930.
- [30] Open Handset Alliance Android. 2009. http://www.openhandsetalliance.com/android_overview.html. Accessed 02-10-2010.
- [31] Hoog A. 2010. Open Source Android Forensics application – beta released! <http://viaforensics.com/android-forensics/open-source-android-forensics-application-beta-released.html>. Accessed: 02-10-2010.
- [32] Trend Micro, Threat Encyclopedia. ANDROIDOS_DROIDSMS. 2010. http://about-threats.trendmicro.com/Malware.aspx?language=us&name=ANDROIDOS_DROIDSMS.A. Accessed: 16-01-2011.
- [33] Vennon, T. and Stroop, D. 2010. Android Malware, A Study of Known and Potential Malware, SMobile Systems, Columbus, Ohio.
- [34] Android Developers. 2009. Android Community: What is Android?. <http://developer.android.com/guide/basics/what-is-android.html>. Accessed: 02-10-2010.
- [35] Hoog A. 2010. An Introduction to Android Forensics. <http://www.dfinews.com/articles.php?pid=974>. Accessed: 16-01-2011.
- [36] Systems and Internet Infrastructure Security. 2008. Android Security Framework. <http://siis.cse.psu.edu/slides/android-sec-tutorial.pdf>. Accessed: 20-05-2010.
- [37] Oracle Database. 2005. Data Mining Concepts. http://download.oracle.com/docs/cd/B28359_01/datamine.111/b28129/classify.htm. Accessed 16-01-2011.

-
- [38] Agrawal, R., Imielinski, T., and Swami, A. 1993. Mining association rules between sets of items in large databases. Proceedings of the 1993 ACM SIGMOD International Conference, Volume 22, no. 2, pp. 207-216.
 - [39] Hegland M. 2000. Algorithms for Association Rules, Australian National University, Canberra ACT 0200, Australia.
 - [40] Giradello, A. and Michahelles F. 2010. Explicit and implicit ratings for mobile applications, 3. Workshop Digitale Soziale Netze and der 40 Jahrestagung der Gesellschaft für Informatik, Leipzig, September.
 - [41] Shirali-Shahreza, M. and Shirali-Shahreza, S. 2006. Collage Steganography, proceedings of the 5th IEEE/ACIS International Conference on Computer and Information Science and 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse, Honolulu, HI, USA, pp. 316-321.

A APPENDIX

A.1 Flow chart of the Suspicious Application Detection Algorithm

