Research Article

DroidDissection: A Hybrid Analysis Framework for Android Malware Detection and Analysis

İlker Kara[1*]

[1]Department of Medical Services and Techniques, Eldivan Medical Services Vocational School Çankırı, Karatekin University, Turkey, Orcid ID: https://orcid.org/0000-0002-9670-8964, E-mail: karaikab@gmail.com
[*] Correspondence: karaikab@gmail.com

Received: 12 March 2025
Revised: 11 September 2025
Accepted: 13 September 2025
Published: 17 September 2025

Reference: Kara, İ.(2025). DroidDissection: A Hybrid Analysis Framework for Android Malware Detection and Analysis. The European Journal of Research and Development, 5(1), 130-148.

Abstract

The Android operating system dominates the mobile ecosystem due to its flexibility, large application market, and open-source architecture. However, these same characteristics make Android an attractive platform for attackers who distribute malicious applications, particularly those designed to intercept banking transactions and steal confidential information. Existing security mechanisms mostly rely on either static or dynamic inspection, and these isolated techniques often fail to reveal concealed or runtime-triggered malicious behavior.
In this study, we present DroidDissection, a framework designed specifically for Android malware detection with an emphasis on banking-related threats. The framework combines static code and permission inspection with controlled dynamic execution, enabling deeper observation of behavior that only emerges during runtime. A real malware sample was examined to validate the approach. The experimental results show that the hybrid inspection strategy increases the accuracy of malware identification and helps uncover behaviors that traditional individual methods may overlook. These findings indicate that the proposed framework can strengthen defense mechanisms against evolving cyber threats targeting Android devices.

Keywords: *Android Malware Detection; Banking Malware; Hybrid Analysis; Static* and Dynamic Analysis; Mobile Security; Cyber Threats; Threat Intelligence

1. Introduction

Nowadays, smart phones have been more popular than computers in online actions and activities due to their mobility and multivariate functionalities [1]. Users can use their mobile devices for different purposes such as browsing the Internet, online banking, social media platforms, file sharing, messaging and entertainment [2]. However, this popularity makes smart phones more attractive for cyber attackers, also users' unawareness turns them profitable [3]. Installing malwares through phishing attacks in target devices is the most common way between attackers, because almost all the users click on “Next” or “Yes" buttons when they want to install an application from application markets [4]. Unfortunately, they don’t have any idea what the application to be installed asks them during installation and the users give any permission the application asks for [5].

Attackers can access all kinds of information and corporate data through malwares and use them for thier profits from selling information or blackmailing to defaming the victims [2]. Beside these they can turn the devices into botnets and use them in various types of cyber-attacks [6].

The statistics show that the market share of the iOS (iPhone Operating System) is lower than the Android operating system [7]. By principle, Apple Inc. does not share the source codes of the iOS in order not to expose its codebase, but Android is an open-source operating system for mobile devices and a corresponding open-source project led by Google. On the other hand, the Android operating system is the most widely used mobile operating system that is preferred because of its numerous advantages and ease of use [8]. For these reasons, malware developers, in general, tend to attack against Android-based systems rather than iOS mobile devices.

Cyber-attacks on mobile devices targeting mobile banking applications is one of the most common attacks, and attackers have been focusing on these applications due to their financial returns [9]. These malicious software’s, known as banking trojans, mostly deceives the user through phishing attacks to install an application that mimics an official banking application [10]. When the user enters his/her personal information into the malware application, this information is sent to the attackers for malicious purposes. Unfortunately, it is extremely difficult to detect the presence of a banking trojan, because they can pass the users to their real bank accounts, so the damage can cause huge financial losses [11].

Although new methods have been developed to mitigate this threat by google and android developers, an ultimate solution has not been found yet [1]. In malware students there are two categories; one is to protect devices from malwares and the other is forensic detection and analysis that should be performed on infected devices.

The importance of the second type is that the results can be used to develop applications which the aim is to protect devices also used in cybercrimes courts. Forensic

detection and analysis approaches can be categorized under two branches namely static and dynamic analysis as shown in Figure.1.

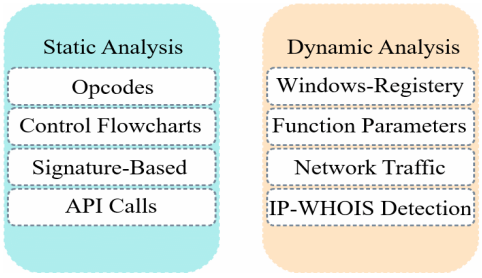


Figure 1: Most used android malware analysis methods in the literature [12].

In the static analysis method, suspicious software is analyzed to determine whether files and source codes contain malware by using the reverse engineering approach [4]. Malware is not actively executed in the static analysis approach, and it is based on examining the functions and code snippets (signature-based technique) of the software deemed to be harmful [4]. The static analysis is a quick way to get results, but the approach is easy to be bypassed through methods such as nesting, packing, hiding, etc. that would be implemented by malware developers. In addition, this method can only be tested on previously detected and pooled malware signatures.

On the other hand, dynamic analysis investigates malware operations. In dynamic analysis the malware is operated in a safe environment (virtual machine (VM) or the sandbox), then its behaviors (file-directory, Windows-Registry, IP movements) are analysed [10]. Dynamic analysis allows us to investigate the memory of a computer. One of the many advantages of this analysis method is "performing live analysis". In contrast to that, there is a limitation of time which means malware can vary according to the status of the lab environment and time [2].

The study presents the analysis of a real-world case of an Android malware attack. By examining an actual malware sample, the approach offers practical insights and demonstrates the application of the proposed method in a real scenario. The study incorporates the analysis of the malware's source code, permissions, installation stages, and behaviors, providing a detailed understanding of the attack.

A hybrid analysis approach is utilized to support studies on Android malware detection, integrating both static inspection and controlled dynamic execution.

- To support Android malware detection studies, the research applies a hybrid analysis method that combines static inspection with dynamic

execution monitoring. This integration allows for a more comprehensive examination of the mobile malware, leveraging the strengths of each method. Static analysis focuses on examining the APK file, permissions, and code structure, while dynamic analysis observes the malware's behavior during runtime, including network activities and interaction with system resources.

- The recommended method utilizes various free software tools to perform the static and dynamic analyses on mobile devices. Tools like virustotal.com, JADX, MEmu, Wireshark, and Autopsy (with the Android Module) are employed for different stages of the analysis. The reliance on open-source tools makes the approach accessible to a wider range of users and promotes transparency in the analysis process.
- To support Android malware detection research, a dataset of 1050 samples is released together with the malware specimen used in the analysis, FinansBank Mobil "Sube_com.Android.mob.apk". (https://ilkerkara.karatekin.edu.tr/RequestDataset.html).

The rest of this paper is organized as follows: In Section 2, we review recent related studies. Section 3 proposes a viable method for mobile malware detection and analysis. Section 4 discusses how it can be implemented in a real case. Finally, Section 5 concludes demonstrated remarks and explains possible future research directions

2. Related work

There are numerous approaches in today's Android market applications that use automated and manual tools to combat mobile malware attacks. This complicates the fight against Android mobile malware. It should be noted that the Android mobile malware uses a proactive approach with user intervention. The Android mobile malware designed by reactive approach, however, takes advantage of the vulnerabilities existing in the current application and system [13].

In Jacob et al. stated that signature-based malware detection is a quick method to get results [14]. This method is known to yield exact results and is used by companies such as Norton and McAfee, which produce antivirus software for commercial purposes. Conventional antivirus products are based on signature-based analysis but they fail to detect constantly updated Android mobile malware. For Android mobile malware detection and analysis, Enck et al., investigated anomalies (e.g. IMEI, phone number), external data (e.g. Internet, SMS), and working behaviors to detect the flow of information between specific sources of information [15]. However, creating a comprehensive and secure analysis environment is costly when automating the dynamic analysis.

Enck et al., have used a static approach to identify malicious behaviors and to alert the user at the installation time [15]. Barrera et al., [16] used self-organising tools to analyses individual permissions in Android mobile applications and used permission-based DroidRanger [8], and RiskRanker [17] approaches for filtering. This approach yields high rates of false positives and allows a limited number of analyses.

Zhou et al., have added manual analysis controls for native code architectures of certain Android applications to reduce false-positive rates [8]. Although existing markets are functional and relatively healthy, it was concluded that rigorous investigations are necessary, especially for unregulated alternative application markets with this approach. Anomaly, an application developed by Shabati et al., based on the anomaly-based detection method, uses a behavior-based malware detection technique [18]. The application continuously monitors system information such as device status, power level, processor usage, network traffic, and then classifies the applications by using its algorithm.

SMS Profiler and IDM (Internet Download Manager) applications developed for iOS are used to detect whether the system services are used by malicious software. This approach is used to detect malware only if the anomaly continues. On the other hand, a sudden attack is quite difficult to detect with this approach. With the DroidScope program, Yan et al., focused on dynamic monitoring of the whole system running on a virtual machine [19].

Since malware, in general, are not able to identify the real environment it runs on, it cannot detect the security measures to be taken and their behaviors will be revealed. The major problem of these applications is uncertainty for the detection of when the malware will start running.

The Droid Legacy program developed by Deshotels et al., is a tool that monitors system calls of the application and creates its signature by dividing the malware into multiple modules [20]. This approach was found to have a low success rate. The Droidma application developed by Wu et al., has a higher success rate and uses various features such as system calls, hardware, and software components, and permissions to determine whether an application is harmful [21]. The Crowdroid program is developed by Burguera et al., for malware detection, monitors system calls on the device, and sends those calls to the central server [22]. These data were used to determine the behaviors of applications by creating a data set for a method to determine the difference between malicious software and non-malicious software, and the applicability of this method was tested in malware detection. It was concluded that this method is an effective tool for isolating malware and alerting users about the downloaded malware. Dendroid, which is an application developed by Suarez-Tangil et al., attempts to detect malware by text mining [23]. Dalvik codes were reverse-engineered in a set of known malwares. The

common code structures of the application were identified, collected under one structure, and used during the process of analysis.

Ayed stated certain permissions used by malware. In the research, Ayed introduced that it is not sufficient for a security measure to ask the user about the required permissions before installing the applications [24]. There should be a separate mechanism for checking the required permissions while raising awareness of the end-user for the control of these permissions.

Peng et al., have analyses not only the permissions but also the existence of suspicious code in applications [25]. For this purpose, the authors used the official Google Android market and other malware data set. The McBoost uses N-gram analysis and multilayer Perceptron heuristics to detect packed Android mobile malware codes [26]. ForeCast performs dynamic analysis for the static and behavioral activities of Android mobile malware. With this approach, only static detection of malware fails for the packed ones. Thus, the packed structures should be executed, and dynamic analysis should be used for detection.

Yunus et al., in their study, explained the use of static analysis, dynamic analysis and memory analysis in android malware detection and analysis, also stated that hybrid analysis approaches that use both static and dynamic analysis can be used [27]. Emphasising that each analysis has advantages and disadvantages, they suggested the hybrid analysis method, which is the most effective method in which all of these analysis approaches are used together, and recommended to be used in android malware detection and analysis. In this study, similar to the work of Yunus et al., we performed the detection and analysis process, based on the proposed hybrid analysis method on a real attack case. Studies in the literature have addressed the problem while using different approaches without either justifying their results in a real case environment or providing a method of step-by-step analysis.

In this study, the methods proposed in the literature are reviewed, and we discuss the detection and analysis of a real Android mobile malware attack based on the proposed method using freely available programs. Kara provided an algorithm that explains how to analyze malware [28].

In another study, Kara et al., argued how to analyze and detect malware in a real cyber-attack case. Furthermore, we define two important shortages in the used method: (i) most of the programs used in the recommended algorithm are operated by using a dongle; (ii) the authors did not apply any access link to evaluate the usability of the proposed method [29].

Therefore, in this study, we use an access link in analyzing the real attack case and propose an approach to mobile malware detection and analysis with free software.

3. Material and method

In this section, we present the methodology employed in our study, which comprises two main steps: image copy and a hybrid analysis process divided into static and dynamic analysis stages, as illustrated in Figure 2. The steps involved in applying these methods are detailed in the following sections.

3.1 Malware detection and analysis process

Mobile malware analysis starts with static analysis, which collects information through various methods such as open-source research (e.g., www.virustotal.com), signature-based verification, control flow charge analysis, and hash value confirmation (e.g., MD5, SHA1). Subsequently, dynamic analysis follows, involving the examination of IP traffic activities, functions, Windows registry records, and file-array structures used by the mobile malware. Dynamic analysis offers a comprehensive review of the features, activities, and capabilities of the mobile malware.

3.2 Basic methodology for android malware detection through static and dynamic analysis

In this section, we delve into the detailed detection and analysis process using our mobile malware analysis approach, conducted on a real case study. We employ commonly used methods for detecting and analyzing Android mobile malware attacks. Free analysis programs are selected to detect and analyze Android malware attacks, utilizing both static and dynamic analyses.

3.3 Hybrid analysis model for android malware detection

In this section, we introduce the Android malware detection and analysis model, as depicted in Figure 2. The model consists of three stages:

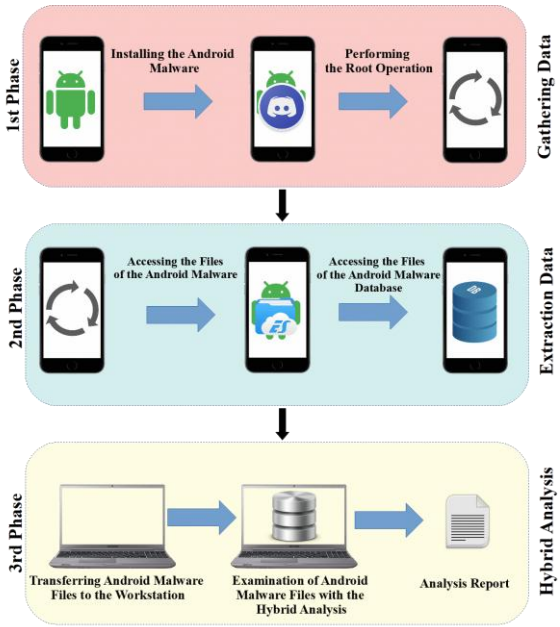


Figure 2: The overall workflow of the proposed model.

First Stage: Android malware execution after transferring to the smartphone, necessitating rooting the device to copy Android malware files to the examination computer.

Second Stage: Data extraction process and transfer of files for analysis using UFED 4PC program, facilitating image copy acquisition.

Third Stage: Android malware analysis processes carried out using the hybrid analysis approach, presenting the results.

3.4 Hybrid analysis model for android malware detection

Mobile malware detection in forensic analysis requires conducting analysis on image copies to be sure of evidence integrity. The image copy procedure involves bit-to-bit copying of all sectors or specific portions of data stored in digital devices. Adherence to international standards, such as using write blockers, is imperative, especially in forensic investigations. The image confirmation process includes hash values (e.g., MD5, SHA-1, or SHA-256) to verify the integrity of the imaging process.

There are physical and logical image copy methods. Generally, the physical method is used for data storage units where all units will be investigated. Logical image copy methods are used when a certain section/file will be investigated within the data storage unit. The physical image acquisition process refers to the process of making a full bit-to-

bit copy of the target mobile device, since the selected sample for the study is a real case, so all the examinations were made by taking a physical image. Moreover, having an image of an infected device is still won't be enough if proper and well-thought analysis methods, like the methods represented at Figure3, are not followed.

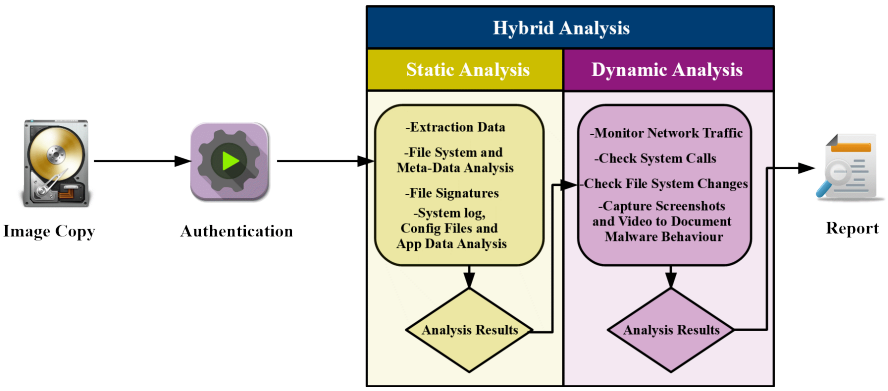


Figure 3: The workflow of the proposed android malware analysis algorithm.

3.4 Hybrid analysis for android malware detection

Combining static and dynamic analysis techniques, hybrid analysis provides a robust approach to analyzing malicious files. This technique integrates code analysis and memory analysis to uncover malware roots, even with sophisticated evasion tactics. In static analysis, APK files, user permissions, and suspicious code are examined, followed by dynamic analysis using an emulator to observe application behaviors.

This analysis approach is designed to overcome the shortages available in both static and dynamic analyses. In static analysis we will analyze the APK file, user permission and identify suspicious code using virustotal.com, signature-based verification, text statements, control flow charge and hash confirmation. Then based on the static analysis report we will start the dynamic analysis using an emulator which will be used to run the suspicious APK file to view and analyze the application behaviors.

3.5 Experimental setup

All analyses were performed on a Dell Precision T5820 brand workstation with Xeon W-2133 CPU, 16GB RAM and 512GB SSD running Windows 10 Pro operating system. To prevent being affected by possible attacks from the mobile malware sample on the image copy taken from the victim's mobile phone, we established a virtual environment using VirtualBox emulator, created a virtual machine dedicated 6 cpu-core, 8GB RAM and 256GB SSD.

A forensic image of the victim's Android device was acquired and transferred to a virtual machine on the workstation. The purpose of performing the analyses on the virtual machine is to minimize the possible risks by running the image in an isolated environment.

"VirtualBox v.6.1.16 (Free Version)", UFED 4PC v.7.34.1.23 (Free Version), Volatility Framework v.2.6, www.virustotal.com, Autopsy v.3.1 (Free Android Module) JADX v.1.2.0 MEmu v.7.0.1., and "Wireshark v.3.2.7 (Free Version)" are free software to perform static and dynamic analyses on mobile devices for all users. It must be known that the essence of a well-curated and correct case study is vital. To establish a case study, we worked with a cyber-security company in Turkey to obtain real life mobile malwares.

In the analyses, static analysis was firstly performed using www.virustotal.com, JADX, and MEmu programs to examine the suspicious activities of the mobile application such as changes in mobile devices, areas for providing access to system calls. At the end of the static analysis, the possible IP address that the suspect application tries to contact with the attacker was examined using the Wireshark program while running the application in a secure analysis environment.

3.6 Analysis of case study

The investigation of the mobile phone is conducted using the UFED 4PC program, with respect to international standards with image copy acquisition guarding image with write blocker to preserve the data integrity (Table 1). Suspicious applications are identified, analyses, and queried using various tools, revealing the presence of Android malware. Dynamic analysis is carried out to detect system activities, and network traffic analysis identifies connections to malicious domains.

Table 1: Image information of victim system.

Device, product name	Total size	Acquisition MD5	Acquisition SHA1
Samsung Galaxy S24, Android 15	122.016 Bytes (122,1 GB)	015c13844eb7078509bcf67b8764c3a1	920189a558bda71cf7d8b6433f531b434d4ed3fd

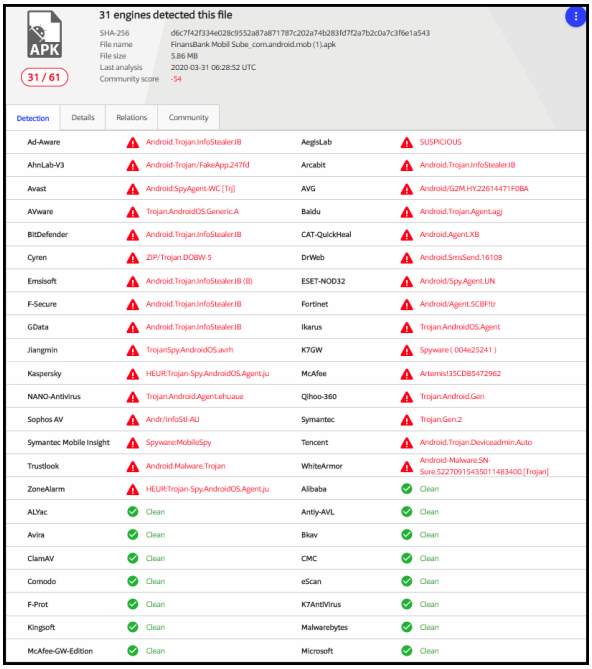


Figure 4: The result of an online query was performed through the www.virustotal.com website.

Malware detection and analysis for the mobile device showed that there was a suspicious application once the last downloaded applications using network browser history were checked. Then, this suspected application, named "FinansBank Mobil Sube_com.Android.mob.apk", was analyzed. For this purpose, the suspect application was queried through www.virustotal.com website, which has many antivirus programs and can perform online queries on suspicious files with MD5 hash validation values in the scope of static analysis as shown in Figure 4.

In the query output, shown in Figure 5, the "FinansBank Mobil Sube_com.Android.mob.apk" file was found to be an Android mobile malware. To access the source codes of the "FinansBank Mobil Sube_com.Android.mob.apk" malware, the "Java Decompiler JD-GUI" program was used for analysis.

Decompiled codes of the malware were obtained in the analysis as shown in Fig. 4. After accessing the source code of Android malware, suspicious files were investigated. Analysis of the "AndroidManifest.xml" file of the Android malware showed that the malware has permissions on numerous features on mobile devices as shown in Figure 5.

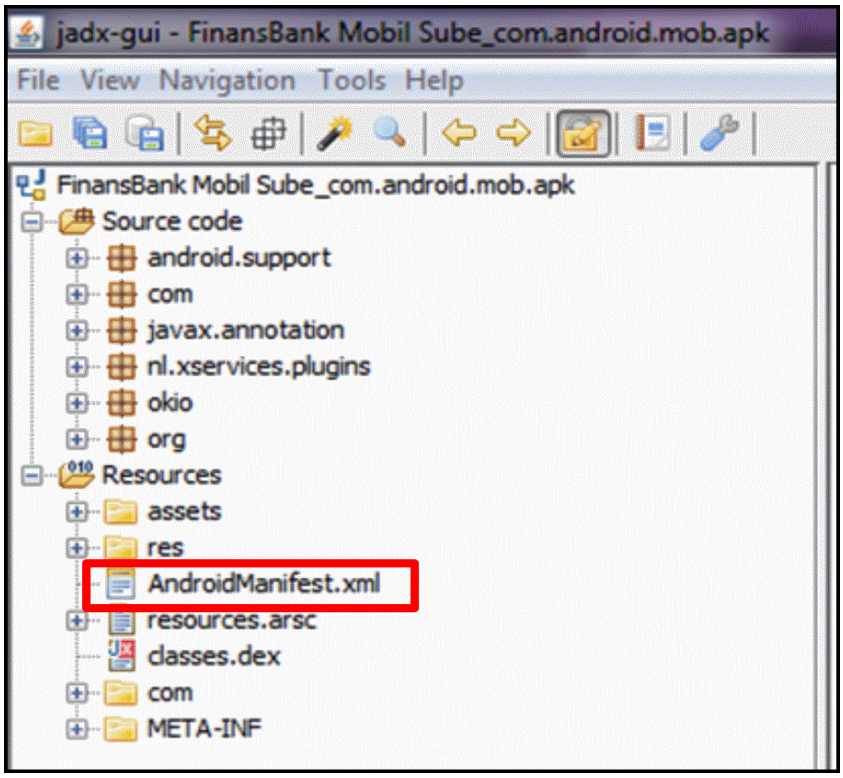


Figure 5: Screenshot of the decompiled file "FinansBank Mobil Sube_com.Android.mob.apk".

```
?xml version="1.0" encoding="utf-8"?>
manifest xmlns:android="http://schemas.android.com/apk/res/android" android:version
   <uses-sdk android:minSdkVersion="11" android:targetSdkVersion="19" />
   <supports-screens android:anyDensity="true" android:smallScreens="true" android:
   <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
   <uses-permission android:name="android.permission.CALL_PHONE" />
   <uses-permission android:name="android.permission.READ_CONTACTS" />
   <uses-permission android:name="android.permission.WRITE_CONTACTS" />
   <uses-permission android:name="android.permission.READ_PROFILE" />
   <uses-permission android:name="android.permission.RECEIVE_SMS" />
   <uses-permission android:name="android.permission.WRITE_CALL_LOG" />
   <uses-permission android:name="android.permission.READ_CALL_LOG" />
   <uses-permission android:name="android.permission.SEND_SMS" />
   <uses-permission android:name="android.permission.VIBRATE" />
   <uses-permission android:name="android.permission.INTERNET" />
   <uses-permission android:name="android.permission.READ_SMS" />
   <uses-permission android:name="android.permission.WRITE_SMS" />
   <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
   <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
   <uses-permission android:name="android.permission.READ_PHONE_STATE" />
   <uses-permission android:name="android.permission.WAKE_LOCK" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.INSTALL_DRM" />
   <uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
   <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
   <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
   <uses-permission android:name="android.permission.READ_LOGS" />
   <uses-permission android:name="android.permission.MMS_SEND_OUTBOX_MSG" />
   <uses-permission android:name="android.permission.INSTALL_SHORTCUT" />
   <uses-permission android:name="android.permission.SYSTEM_TOOLS" />
   <uses-permission android:name="android.permission.WRITE_CONTACTS" />
   <uses-permission android:name="android.permission.READ_PROFILE" />
   <uses-permission android:name="android.permission.RECEIVE_MMS" />
   <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
   <uses-permission android:name="android.permission.INSTALL_DRM" />
   <uses-permission android:name="android.permission.WRITE_APN_SETTINGS" />
   <uses-permission android:name="com.android.launcher.action.UNINSTALL_SHORTCUT" /
```

Figure 6: The contents of the file "AndroidManifest.xml".

Moreover, "AndroidManifest.xml" malware shown in Fig.6 was found to have permission to turn off notifications, read messages, receive all requests of the phone services, activate five stages of the installation challenges, and have receivers capture personal information on the mobile device as shown in Fig.7. For these installation stages, the source code was analyzed, and to see if a local web server, "apache.cordova", existed in the libraries of the Android malware.

In the analysis of the directories of the Cordova web server, the "index.html" file in the "web" directory was found to have HTML codes of the web page to be opened on the screen, and the "MainActivity.class" file was found to define all installation steps of the malware as shown in Figure 8. Moreover, at the Figure 9 it is pointed out how the malware install itself and its services to run properly.

Figure 7: The contents of the Android malware activities.

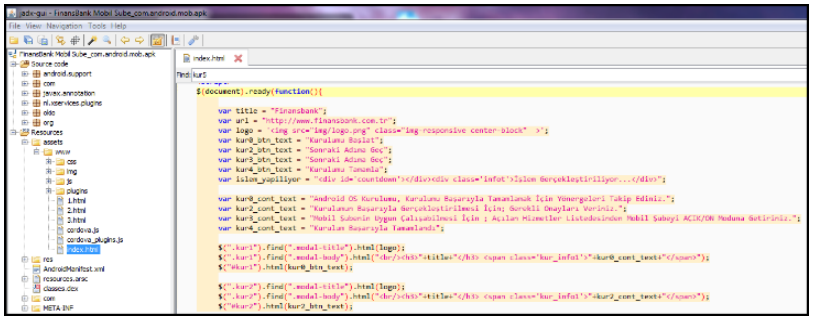


Figure 8: The contents of the file "index.xml".

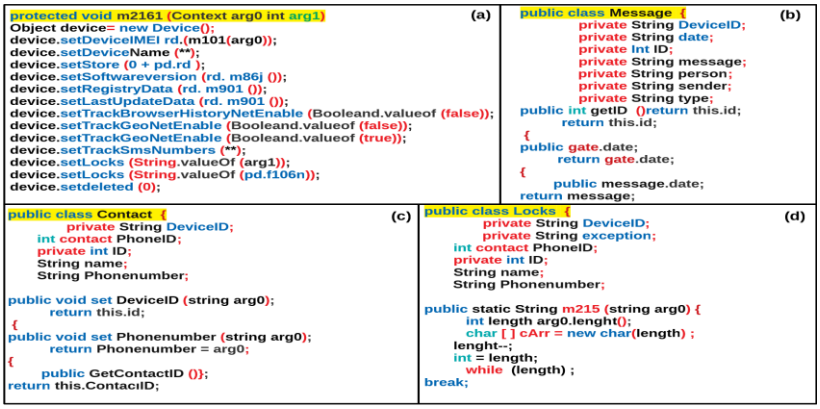


Figure 9: Five-phase installation model for Android malware detection and analysis.

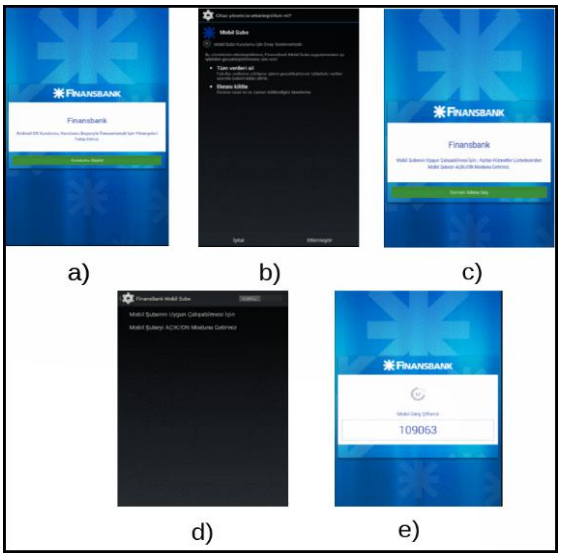


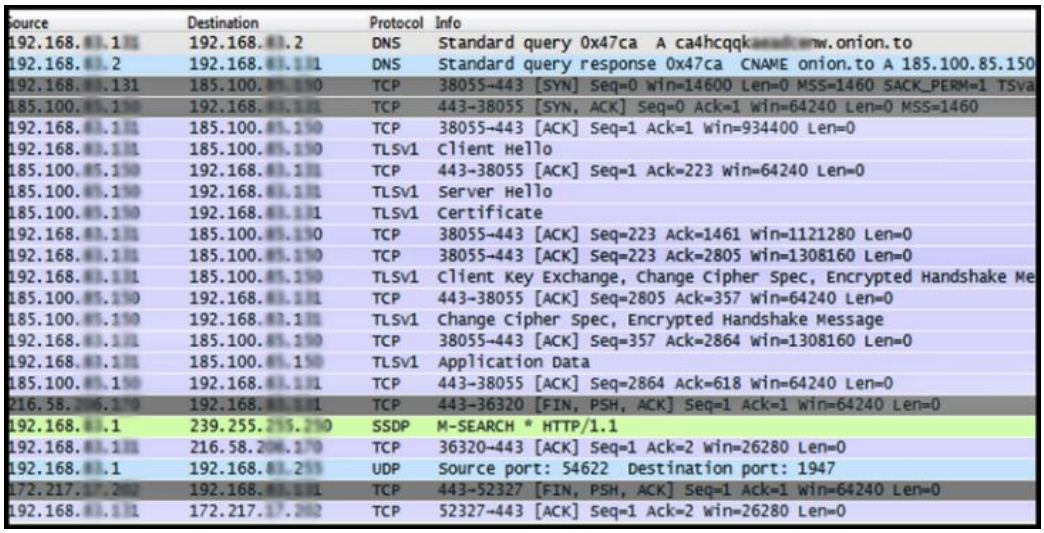
Figure 10: The contents of a) the file named "MyDevicePolicyReceiver.class"; b) the file named "Message.class", which includes the source code used to collect messages in the device; c) the "Contact.class", which contains the source codes for capturing the contacts information; d) the file named "Locks.class", which includes the source code that captures the passwords in the device.

After the static analysis of Android malware is completed, the dynamic analysis phase was started in a safe environment with the program "MEmu" to detect the activity on the system. In the analyses performed with the Autopsy 3.1 (Free Android Module) program, the International Mobile Equipment Identity (IMEI), Identification Number (ID), file directory, installed apps, GPS, phone book, Internet history, SMS, and Call History, and device password obtained, and some of them are shown in Figure 10.

The IP address is an identification number assigned by Internet Service Providers (ISPs) to connect the Internet by information devices. An IP address is of two kinds: static and dynamic IP addresses. As the name implies, a Static IP address identifies the fixed IP address. A dynamic IP address can be changed over time, and the connection can be changed once the IP address is re-established, or can be changed manually to avoid detection.

Thanks to the IP address, it is possible to access the location of the connected subscriber. Due to this feature, IP address is one of the most important distinguishing things that define the information of system users. There is no doubt that IP address information is needed to determine both the perpetrator and the location in crimes committed with information systems. It may be considered that the detected IP may belong to the attacker. However, there are specific difficulties in detecting IP (e.g., dynamic IP) and the reliability of the detected IP is controversial. Therefore, the IP address is not a result of forensic investigation, but it can be considered as the starting point.

Upon detection of the fact that personal data in the victim device had been sent to the attacker before network activities of the "FinansBank Mobil Sube_com.Android.mob.apk" Android malware were analysed by the "Wireshark 3.2.0" program. The analysis showed that Android malware connected to the "caxxxxxxxxxxxxxx.onion.to" domain name and "185.XX.85.XXX" IP address as shown in Figure 11.



Source	Destination	Protocol	Info
192.168.[redacted].1[redacted]	192.168.[redacted].2	DNS	Standard query 0x47ca A ca4hcqqk[redacted]nw.onion.to
192.168.[redacted].2	192.168.[redacted].1[redacted]	DNS	Standard query response 0x47ca CNAME onion.to A 185.100.85.150
192.168.[redacted].131	185.100.[redacted].150	TCP	38055→443 [SYN] Seq=0 Win=14600 Len=0 MSS=1460 SACK_PERM=1 TSva
185.100.[redacted].150	192.168.[redacted].1[redacted]	TCP	443→38055 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460
192.168.[redacted].1[redacted]	185.100.[redacted].150	TCP	38055→443 [ACK] Seq=1 Ack=1 Win=934400 Len=0
192.168.[redacted].1[redacted]	185.100.[redacted].150	TLSv1	Client Hello
185.100.[redacted].150	192.168.[redacted].1[redacted]	TCP	443→38055 [ACK] Seq=1 Ack=223 Win=64240 Len=0
185.100.[redacted].150	192.168.[redacted].1[redacted]	TLSv1	Server Hello
185.100.[redacted].150	192.168.[redacted].1[redacted]	TLSv1	Certificate
192.168.[redacted].1[redacted]	185.100.[redacted].150	TCP	38055→443 [ACK] Seq=223 Ack=1461 Win=1121280 Len=0
192.168.[redacted].1[redacted]	185.100.[redacted].150	TCP	38055→443 [ACK] Seq=223 Ack=2805 Win=1308160 Len=0
192.168.[redacted].1[redacted]	185.100.[redacted].150	TLSv1	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Me
185.100.[redacted].150	192.168.[redacted].1[redacted]	TCP	443→38055 [ACK] Seq=2805 Ack=357 Win=64240 Len=0
185.100.[redacted].150	192.168.[redacted].1[redacted]	TLSv1	Change Cipher Spec, Encrypted Handshake Message
192.168.[redacted].1[redacted]	185.100.[redacted].150	TCP	38055→443 [ACK] Seq=357 Ack=2864 Win=1308160 Len=0
192.168.[redacted].1[redacted]	185.100.[redacted].150	TLSv1	Application Data
185.100.[redacted].150	192.168.[redacted].1[redacted]	TCP	443→38055 [ACK] Seq=2864 Ack=618 Win=64240 Len=0
216.58.[redacted].170	192.168.[redacted].1[redacted]	TCP	443→36320 [FIN, PSH, ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.[redacted].1	239.255.[redacted].250	SSDP	M-SEARCH * HTTP/1.1
192.168.[redacted].1[redacted]	216.58.[redacted].170	TCP	36320→443 [ACK] Seq=1 Ack=2 Win=26280 Len=0
192.168.[redacted].1	192.168.[redacted].255	UDP	Source port: 54622 Destination port: 1947
172.217.[redacted].202	192.168.[redacted].1[redacted]	TCP	443→52327 [FIN, PSH, ACK] Seq=1 Ack=1 Win=64240 Len=0
192.168.[redacted].1[redacted]	172.217.[redacted].202	TCP	52327→443 [ACK] Seq=1 Ack=2 Win=26280 Len=0

Figure 11: The recorded data through the Wireshark software.

4. Discussion

Various methods exist for Android mobile malware detection and analysis, but attackers continuously evolve tactics, presenting challenges in combating malware threats. Our study presents a novel approach to detect and analyze Android mobile malware, demonstrated through a real case study. The selected method utilizes freely available software tools for analysis, offering step-by-step guidance for detecting and analyzing Android malware attacks. The study underscores the importance of addressing the unique challenges posed by banking Android malware, emphasizing the need for ongoing research and vigilance in cybersecurity efforts.

In this study, a robust method used in the detection and analysis of Android mobile malware was represented. In addition, the detection and analysis of a real Android mobile malware attack were discussed based on the proposed method using freely available software tools. In this study, a mobile banking Android malware attack, which is the most preferred by attackers, was chosen as an example case. The chosen method was ensured to be applicable by other researchers. For this purpose, free analysis programs were selected for all analyses, and the detection and analysis of the sample banking Android malware were step-by-step demonstrated.

Unlike other malware, banking Android mobile malware is specially designed since each one of them mimics a specific bank's application. Therefore, each cyberattack must be separately evaluated. When the code structure of this malware is examined, it should be considered that the malware can be easily modified for other bank companies so that the malware can pose a major threat to different bank users.

5. Conclusion

The use of smart mobile devices has become very common thanks to their convenience, portability and ability to perform almost all online transactions. Considering the market values, especially the android operating system is more preferred by users and application developers because it is easy to use and open source. For these reasons, android application markets and devices have also turned into an attractive area for attackers.

Firstly, the study incorporates a comprehensive two-step process for mobile malware analysis. This process includes an initial static analysis phase, which gathers information through various methods such as open-source research, signature-based verification, and hash value confirmation. It is followed by a dynamic analysis phase that examines IP traffic activities, functions, windows registry records, and file-array structures used by the mobile malware. This two-step approach allows for a thorough investigation of the malware's features, activities, and capabilities.

Secondly, the study utilizes a hybrid analysis technique that combines static and dynamic analysis methods. By combining these two approaches, the researchers can overcome the limitations present in each method individually. Static analysis provides insights into the APK file, user permissions, and suspicious code identification, while dynamic analysis offers a real-time examination of the malware's behaviors by running it in an emulator. This hybrid approach enables a more comprehensive understanding of the malware, ensuring a thorough analysis of both its static and dynamic aspects.

Additionally, the study emphasizes the importance of creating image copies of the digital devices under investigation. This process involves bit-to-bit copying of all sectors or specific data portions, following international standards and utilizing hash values for verification. The inclusion of image copy process helps ensure the preservation of evidence and maintains the integrity of the original data.

Moreover, the study specifies the hardware and software environment used for the analyses. It mentions the workstation specifications and the use of virtualization (VirtualBox) to create a secure and isolated analysis environment. The employed tools, such as UFED 4PC, Volatility Framework, and various free software programs, are also listed, highlighting the technological aspects of the approach.

Overall, the combination of static and dynamic analysis, the use of open-source tools, emphasis on image copy process, analysis of a real case study, and the detailed

experimental setup make the approach described in the study considerable and contribute to the field of mobile malware analysis with its exclusive usage on open-source tools makes this method transparent and accessible to a variety of users.

6. Acknowledge

The author would like to thank the colleagues who shared their technical opinions during the analysis process. Their practical suggestions helped shape certain parts of the study. This research was completed without any external funding or institutional support.

References

[1] Albakri, A., Fatima, H., Mohammed, M., Ahmed, A., Ali, A., Ali, A., Elzein, N. M. (2022). Survey on reverse-engineering tools for Android mobile devices. Mathematical Problems in Engineering, 2022, Article 4908134.

[2] Sharma, T., Rattan, D. (2021). Malicious application detection in Android: A systematic literature review. Computer Science Review, 40, 100373.

[3] Zaidi, S. F. A., Shah, M. A., Kamran, M., Javaid, Q., Zhang, S. (2016). A survey on security for smartphone device. International Journal of Advanced Computer Science and Applications, 7(4), 1-7.

[4] Muzaffar, A., Hassen, H. R., Lones, M. A., Zantout, H. (2022). An in-depth review of machine learning based Android malware detection. Computers Security.

[5] He, D., Chan, S., Guizani, M. (2015). Mobile application security: Malware threats and defenses. IEEE Wireless Communications, 22(1), 138-144.

[6] Kara, I. (2022). Fileless malware threats: Recent advances, analysis approach through memory forensics and research challenges. Expert Systems with Applications.

[7] Statista. (2022). Global market share held by mobile operating systems since 2009. Retrieved from https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/ (https://www.statista.com/statistics/272698/global-market-share-held-by-mobile-operating-systems-since-2009/)

[8] Zhou, Y., Wang, Z., Zhou, W., Jiang, X. (2012, February). Hey, you, get off of my market: Detecting malicious apps in official and alternative Android markets. In NDSS Symposium*(pp. 50–52).

[9] Zimba, A. (2022). A Bayesian attack-network modeling approach to mitigating malware-based banking cyberattacks. International Journal of Computer Network & Information Security, 14(1).

[10] Moret, J. D., Todd, A., Rose, L., Pollitt, E., Anderson, J. (2022). Mobile phone apps for intimate partner and sexual violence prevention and response: Systematic search on app stores. JMIR Formative Research, 6(2), e28959.

[11] Conti, M., Losiouk, E., Poovendran, R., Spolaor, R. (2022). Side-channel attacks on mobile and IoT devices for cyber-physical systems. Computer Networks.

[12] Sihwail, R., Omar, K., Zainol Ariffin, K. A., & Al Afghani, S. (2019). Malware detection approach based on artifacts in memory image and dynamic analysis. Applied Sciences, 9(18), 3680.

[13] Arif, J. M., Ab Razak, M. F., Mat, S. R. T., Awang, S., Ismail, N. S. N., Firdaus, A. (2021). Android mobile malware detection using fuzzy AHP. Journal of Information Security and Applications, 61, 102929.

[14] Jacob, G., Debar, H., Filiol, E. (2008). Behavioral detection of malware: From a survey towards an established taxonomy. Journal in Computer Virology, 4(3), 251-266.

[15] Enck, W., Ongtang, M., McDaniel, P. (2009). Understanding Android security. IEEE Security & Privacy, 7(1), 50-57.

[16] Barrera, D., Kayacik, H. G., Van Oorschot, P. C., Somayaji, A. (2010, October). A methodology for empirical analysis of permission-based security models and its application to Android. In Proceedings of the 17th ACM Conference on Computer and Communications Security (pp. 73-84).

[17] Grace, M., Zhou, Y., Zhang, Q., Zou, S., Jiang, X. (2012, June). RiskRanker: Scalable and accurate zero-day Android malware detection. In Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (pp. 281-294).

[18] Shabtai, A., Kanonov, U., Elovici, Y., Glezer, C., Weiss, Y. (2012). "Andromaly": A behavioral malware detection framework for Android devices. Journal of Intelligent Information Systems, 38(1), 161.

[19] Yan, L. K., Yin, H. (2012). DroidScope: Seamlessly reconstructing the OS and Dalvik semantic views for dynamic Android malware analysis. In USENIX Security Symposium (pp. 569–584).

[20] Deshotels, L., Notani, V., Lakhotia, A. (2014, January). DroidLegacy: Automated familial classification of Android malware. In Proceedings of ACM SIGPLAN on Program Protection and Reverse Engineering Workshop 2014 (pp. 1-12).

[21] Wu, D. J., Mao, C. H., Wei, T. E., Lee, H. M., Wu, K. P. (2012, August). DroidMat: Android malware detection through manifest and API calls tracing. In 2012 Seventh Asia Joint Conference on Information Security (pp. 62-69).

[22] Burguera, I., Zurutuza, U., Nadjm-Tehrani, S. (2011, October). Crowdroid: Behavior-based malware detection system for Android. In Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices (pp. 15-26).

[23] Suarez-Tangil, G., Tapiador, J. E., Peris-Lopez, P., Ribagorda, A. (2013). Evolution, detection and analysis of malware for smart devices. IEEE Communications Surveys & Tutorials, 16(2), 961-987.

[24] Ayed, A. B. (2015). A literature review on Android permission system. International Journal of Advanced Research in Computer Engineering & Technology, 4(4).

[25] Peng, H., Gates, C., Sarma, B., Li, N., Qi, Y., Potharaju, R., & Molloy, I. (2012, October). Using probabilistic generative models for ranking risks of Android apps. In Proceedings of the 2012 ACM Conference on Computer and Communications Security (pp. 241–252).

[26] Neugschwandtner, M., Comparetti, P. M., Jacob, G., Kruegel, C. (2011, December). Forecast: Skimming off the malware cream. In Proceedings of the 27th Annual Computer Security Applications Conference (pp. 11–20).

[27] Yunus, Y. K. B. M., Ngah, S. B. (2020, February). Review of hybrid analysis technique for malware detection. In IOP Conference Series: Materials Science and Engineering, 769(1), 012075.

[28] Kara, I. (2019). A basic malware analysis method. Computer Fraud & Security, 2019 (6), 11–19.

[29] Kara, I., & Aydos, M. (2022). The rise of ransomware: Forensic analysis for Windows-based ransomware attacks. Expert Systems with Applications, 190, 116198.