Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control
Journal homepage: http://kinetik.umm.ac.id
ISSN: 2503-2267
Vol. 7, No. 3, August, Pp. 289-298

Mobile Device Security Evaluation using Reverse TCP Method

Imam Riadi*[1], Sunardi[2], Deco Aprilliansyah[3]
Department of Information System, Universitas Ahmad Dahlan, Yogyakarta, Indonesia[1]
Department of Electrical Engineering, Universitas Ahmad Dahlan, Yogyakarta, Indonesia[2]
Master Program of Informatics, Universitas Ahmad Dahlan, Yogyakarta, Indonesia[3]

Article Info
Keywords:
Remote Access Trojan, Android, Reverse TCP, Exploit, Metasploit

Article history:
Received: April 06, 2022
Accepted: September 19, 2022
Published: September 28, 2022

Cite:
I. Riadi, D. Aprilliansyah, and Sunardi, "Mobile Device Security Evaluation using Reverse TCP Method", *KINETIK*, vol. 7, no. 3, Sep. 2022.
https://doi.org/10.22219/kinetik.v7i3.1433

*Corresponding author.
Imam Riadi
Email address:
imam.riadi@is.uad.ac.id

Abstract
Security evaluation on Android devices is critical so that users of the operating system are protected from malware attacks such as remote access trojans that can steal users' credential data. Remote access trojan (RAT) attacks can be anticipated by detecting vulnerabilities in applications and systems. This study simulates a remote access trojan attack by exploiting it until the Attacker gains full access to the victim's device. The episode is carried out with several steps: creating a payload, installing applications to the victim's device, connecting listeners, and performing exploits to retrieve important information on the victim's device. Test material using Android 12, problems occurred when trying to install the application because a harmful warning will appear from Play Protect due to not using the latest version of privacy protection which causes the application to be indicated as malware and the like. On Android 11, the application injected with the backdoor was successfully installed on the device and successfully accessed by the attacker. Attackers also get vital information, including system information, contacts, call logs, messages, and full access to the victim's device system directory. Based on this research, Android device users are expected to constantly update the Android version on the device they are using.

1. Introduction

The development of smartphone technology is increasing. Statistical data showed that in 2021, the number of smartphone users would reach 6.3 billion [1]. This data shows that in the industrial era 4.0, everyone has at least one smartphone to communicate, exchange data, find information, or as a form of entertainment such as playing video games, studying online, listening to music, and watching videos. This development has a negative impact, such as criminal in the form of theft of sensitive data and information for Android device users [2][3][4]. One of the most widely used smartphone technologies is Android, a mobile operating system [5]. Android is built using the Linux kernel and was developed by a large company, Google [6]. Android is open source. Any developer can participate in developing this operating system to become a better ecosystem [7]. The Android system is certainly not perfect. There are often security holes even though developers provide security patch updates every month to patch these loopholes. On the Android operating system, attacks occur through network vulnerabilities, applications, and firmware in the operating system. Attacks on the Android system can be divided into hardware attacks, Kernel attacks, hardware abstraction layer attacks, and application package attacks [8]. Security in the Android operating system is sometimes overlooked by users, such as malware and remote access exploits by third parties [9]. Android apps in mobile devices share data to support app operation and a better user experience, increasing security risks to the integrity and confidentiality of device data.

Nowadays, people's lifestyles increasingly depend on mobile applications (apps), such as B. shopping, managing money, surfing the Internet, etc. However, developers mainly focus on application implementation and user experience improvement while ignoring security issues [10]. With the application of users to an application to support their productivity, irresponsible parties are often used to infiltrate malicious code in several applications. Many researchers ignore that applications can share their resources and functionality with other applications using special permissions [11]. Communication security and data privacy are critical in application development [12]. While some of these attacks exploit vulnerabilities in the Android operating system, others are directly related to application-level code written by a large number of developers with varying experiences [13][14][15]. Apart from the application side, the Android Debug Bridge (ADB) Security has attracted much attention from researchers due to its high level of privileges and low level of protection. Many attacks against Android systems exploit ADB's security flaws [16]. Security risks on Android can be minimized by performing security analysis on system vulnerabilities.

Many trojan attacks have occurred in recent years. Criminals were seen deploying a mobile banking Trojan targeting users of Brazil's largest financial service, Itau Unibanco. There are 55 million subscribers worldwide. The

perpetrators use a unique trick to spread their homemade Trojan to victims' devices [17]. Criminals create pages similar to Google Play's official app store to deceive users. For unwary customers, they will think the fake Google Play Store page is official and immediately install the app on their Android tablet or phone [18]. In December 2021, security company Lookout discovered a new Trojan virus called AbstractEmu. The malware is capable of hijacking infected Android phones. Lookout researchers found there were at least 19 malicious applications. Even seven applications included in it could automatically perform the rooting process: All Passwords, Anti-ads Browser, Data Saver, Lite Launcher, My Phone, Night Light, and Phone Plus [19]. In addition, hackers also often take advantage of rising trends. One example is when the Korean drama series Squid Game, which is popular on Netflix with more than 111 million views, is exploited by hacker groups. Experts from Kaspersky found several dozen malicious files on the web with the index name Squid Game. Kaspersky discovered mobile malware exploiting Squid Game's reputation. Instead of a game, the victim downloads a Trojan. Trojan attacks can be anticipated by detecting vulnerabilities in applications and systems.

Vulnerability detection is a method for finding security holes designed to reveal potential security holes [20]. In a digital environment with many cyber threats, regular vulnerability assessment is the right to do to protect personal data. With the back of computer systems, data innovation is utilized as a medium for trading information and data [21]. A lot of information is confidential. Therefore, security is also essential [22][23]. As a result, sensitive data can be properly maintained and protected from various bad effects of cyber attacks. Detecting software vulnerabilities is a vital and challenging issue. Ideally, a detection system (or detector) can detect whether a program contains a vulnerability and be able to pinpoint the type of vulnerability in question [24]. This research was conducted with the scenario of using android devices via reverse TCP. The testing process on the android system is expected to help us understand how third parties carry out remote access trojan attacks and make smartphone users more careful in using their mobile devices.

2. Research Method

Android Security Labware framework was used in this study. Android Security Labware provides a hands-on mobile security experience, promotes interest, and keeps engaged in security [25]. This enables me to gain real-world experience in securing mobile devices, developing and securing mobile applications, and performing penetration testing for mobile devices and mobile applications [26]. This lab kit consists of seven standalone modules covering critical threats related to mobile device security and privacy, mobile application security, and mobile network and communications security (see Table 1). Each module includes:

a. a prelab activity (concept introduction and lab preparation),
b. two hands-on lab activities (one on analyzing threats and the other on associated protection solutions), and
c. a post-lab activity (review questions, assignments, and case studies).
d. Threats to mobile applications come in many forms. For example, mobile malware may collect data without the user's knowledge or consent, collect sensitive or personally identifiable information, or leave security holes in devices. Mobile malware capabilities include activity monitoring and data retrieval, system modification, and unauthorized calls.

Table 1. The Seven Modules in the Android Security Labware

Category	Modules	Information assurance and security topics
Mobile device's security and privacy	Lost or Stolen Mobile Devices	ISA fundamental concepts
	Unauthorized Mobile-Resources Access	Security architecture and system administration
	Mobile Privacy Threat	Cryptography
Mobile-App security	Mobile malware	ISA fundamental concepts
	Secure mobile-app development	Secure software design and engineering
		Security architecture and system administration
Mobile Network and communication security	Mobile SMS	ISA fundamental concepts
	Mobile phishing threats	cryptography
		Network security
		security policy and governance

2.1 Remote Access Trojans

Remote Access Trojans (RAT) are malware that gives the attacker direct interactive access to the victim's personal computer, allowing the attacker to steal personal data from the computer [27]. The process of spying on victims

in real-time using cameras and microphones and interacting directly with victims through dialogue [28]. RATs are the opening door before Hijacker and Ransomware attacks in the most severe cases.

2.2 Android

Android could be a Linux-based working framework planned for touch screen portable gadgets, such as smartphones and tablets. Operating systems have changed significantly from black and white cell phones to the latest smartphones or minicomputers in the last 15 years. One of the most widely used mobile operating systems is Android software, founded in Palo Alto, California, in 2003. Android is designed for everyone, including designers and device manufacturers. This means that many people will be able to experiment, imagine, and create things that have never existed in the world. The history of Android OS versions begins with the release of the Android 1.0 beta in November 2007. Since April 2009, every Android version has been developed with a code name supporting dessert items. These versions were released alphabetically: Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb, Frozen Dessert Sandwich, Jelly Bean, KitKat, Lollipop, Marshmallow, Nougat, Oreo, and Pie. Android 10 Google does not use fruit names for naming the android version. The latest version is Android 12, launching in May 2021 with SDK Build-Tools version 31.

2.3 Metasploit Framework

Metasploit Framework is an open-source penetration testing and development platform that provides access to exploit code for various applications, operating systems, and platforms [29]—written in the Ruby scripting language, with Ruby's status as an object-oriented language. Also, Metasploit is considered multi-platform, running on most Unix and Windows variants. The Metasploit Framework provides a great way of working, but it's difficult for new users to use because Metasploit on Linux doesn't offer a graphical user interface (GUI). It is necessary to understand the syntax and commands to use Metasploit. It requires information about the target system, such as the OS version and installed network services.

2.4 Meterpreter

Meterpreter is a Metasploit attack payload that provides an interactive shell that allows an attacker to explore the target machine and execute code. Meterpreter is deployed using DLL memory injection. Therefore, Meterpreter resides in memory and does not write anything to the disk. When Meterpreter injects itself into an infected process, it doesn't create a new process and can migrate from it to another running process. Therefore, the forensic traces of the attack are minimal. The meterpreter command on android includes several types of modules, namely reverse_http, meterpreter_reverse_http, reverse_https, meterpreter_reverse_https, reverse_tcp, and meterpreter_reverse_tcp

2.5 Testing Method

The testing technique used to analyze RAT attacks with the exploitation method on Android is using White Box Testing. The developers and testers will look for errors in the source code or ready-made software code to correct them again [30]. All white box testing work focuses on examining code flow, such as data flow, structure flow, input and output processing, loop testing, and security of the attached network.

2.6 Attack Scenario

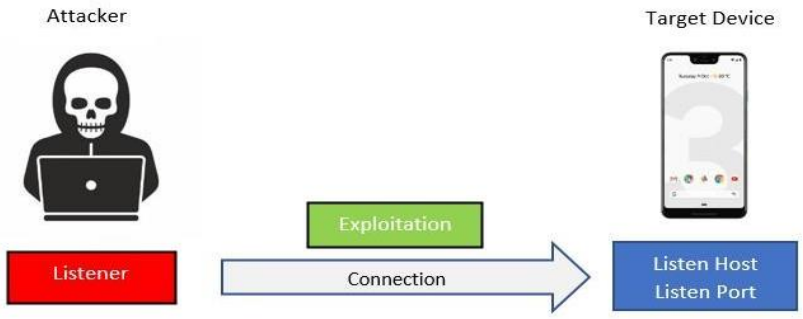

Figure 1. Inject Backdoor for Infected Application

This section provides an overview of Attack scenarios on Android smartphones. Smartphones act as a target for attacks. The design of the attack process is shown in Figure 1. It can be seen that the attacker injects the backdoor into the application, and then the infected application is pushed to the target device. Devices that have been infected are

forced to permit the connection process between the tools and the device being tested using HOST and PORT. Furthermore, the infected device will be remotely accessed from the attacker's computer.

2.7 Exploit

The exploit process is carried out using android devices with versions 11.0 and 12.0. The steps require a fair amount of knowledge about reverse Android apps and can be time-consuming. Before carrying out the exploit process, several steps must be done, namely configuring the payload, matching listeners, and running the exploit.

In the payload configuration, the attacker configures the payload. The payload is written using a bash shell script. The written payload will be executed on the terminal in a shell script. This shell script carries the process payload in the form of an android apk file. This payload itself functions as a backdoor. The attacker can execute the attack by creating an application payload by entering the attacker's IP address as LHOST and the attacker's PORT as LPORT in the command msfvenom -p **android/meterpreter/reverse_tcp LHOST**=(the attacker's IP address**) LPORT**=(the attacker's PORT number) **R > root/backdoor. apk**. The method used in this attack is reverse TCP, as shown in Figure 2. After the payload installation, the next step is configuring the listener to create meterpreter sessions.

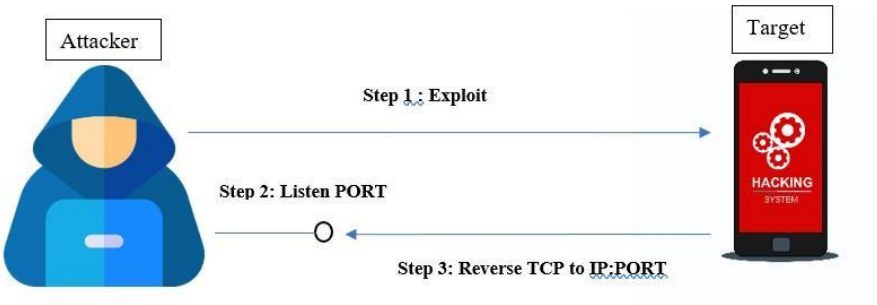

Figure 2. Reverse_tcp Scheme

In the connection to the listener for the experiment, an attack was carried out using the reverse_tcp method, namely the reverse meterpreter shell that passes through the firewall. The victim establishes a connection with the attacker, so the listening method on the attacker's machine will wait for incoming connections. Here's how to configure the listener:

a) Open the Metasploit console in the kali Linux terminal by typing "**msfconsole.**"
b) After the Metasploit console opens, then type the command "**use multi/handler**" so that the attacker can set **LHOST** and **LPORT** to create a listener.
c) Next set the payload by typing "**set payload android/meterpreter/reverse_tcp**"
d) Set **LHOST** <attack IP address> this is the IP address used as a host.
e) Set **LPORT** <PORT attacker> this is the PORT used as a connection entry point between the attacker and the victim.
f) **Exploit** is a command to start an exploit between the attacker and the victim's device that has been previously connected.

3. Results and Discussion

3.1 Injection Backdoor

At this stage, the tester will enter the payload into the application, which will later be installed on the victim's device. The backdoor injection process can be seen in Figure 3. Injecting payload on msfconsole uses the reverse_tcp method with LHOST as the attacker's IP address of 192.168.100.235 and PORT 4444. The resulting output is an android application with the name fuct.apk.

```
msf6 > msfvenom -p android/meterpreter/reverse_tcp LHOST=192.168.100.235 LPORT=4444 R > fuct.apk
```

Figure 3. Inject Payload

3.2 Installing the Application

Installing the application injected backdoor on Android 12 reads harmful by google play protect. Whereas on Android 11.0, it was successfully installed without warning at all. For more details, it can be seen in Figure 4. When an attacker tries to install an application filled with a backdoor payload, Google Play Protect immediately alerts that the application does not use the latest version of privacy protection. This causes the application to be indicated as containing malware and the like, whereas, on Android 11, the application was successfully installed without a harmful warning.

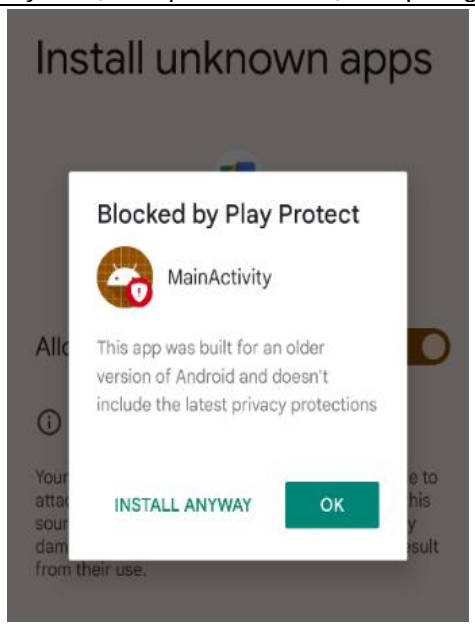

Figure 4. Harmful Application Detected

It can be seen in figure 4 that when installing an application that has been injected with a backdoor, a blocked by play protect warning popup appears. The application was successfully installed on the second try by performing a force install and passing the popup warning form.

3.3 Connecting Device

At this stage, the attacker connects to a device that has installed the application from the backdoor injection process. The connection is made by adjusting the attacker's IP and PORT with the targeted victim's device. More details can be seen in Figures 5, 6, and 7.

Figure 5 shows a page about mobile phones in which there is information on the Android version used, version 11, and the IP address for the IPv4 and IPv6 versions. The IPv4 address of the victim's device is 192.168.100.66 and IPv6 fe80::833c:53cc:eea:7ce. The IP address used is IPv4 which will later be associated with the Host IP address and PORT on the attacker's device.

Figure 5. IP Address of the Target Device

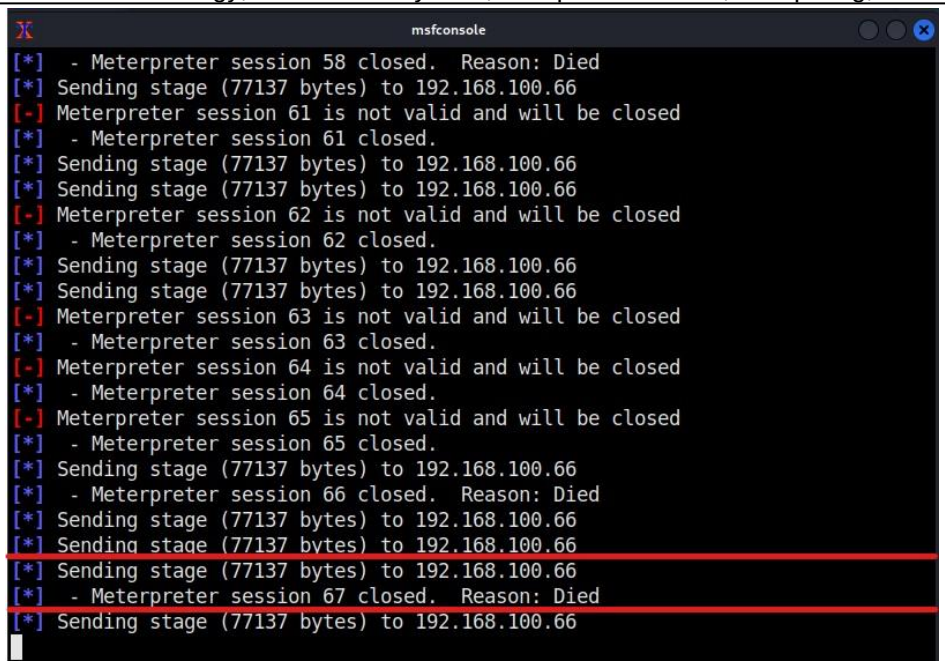

Figure 6. Meterpreter Session Closed

It can be seen in Figure 6 that the first attempt until session 67 was unsuccessful. Figure 6 shows msfconsole trying to send a stage with a data size of 77137 bytes to the victim's IP address. Still, it failed, so the meterpreter session opening process was unsuccessful, and the attacker could not access the victim's smartphone. While the meterpreter session process was successfully opened in the second experiment, more details can be seen in Figure 7.

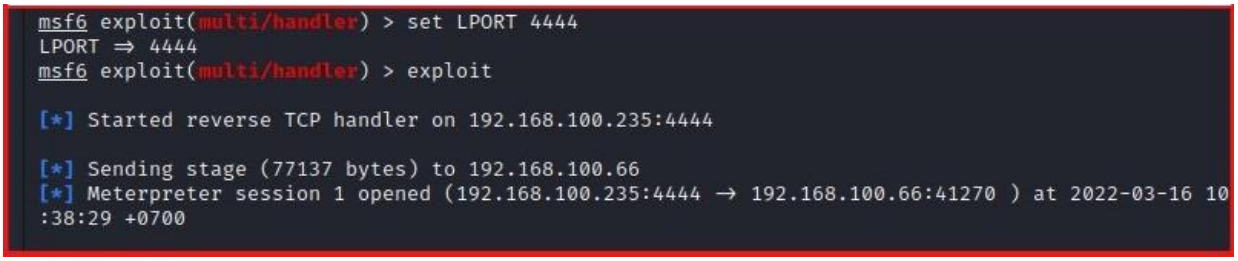

Figure 7. Meterpreter Session Opened

It can be seen in Figure 7 that The meterpreter session conducted using msfconsole was successfully opened. The victim's smartphone IP 192.168.100.66:41270 successfully connected to the Host IP 192.168.100.235:4444 on 2022-03-16 so that the attacker could execute on the victim's device.

3.4 Target Execution Process

The process can be performed after the Meterpreter session is open. Furthermore, the attacker can control the victim's device remotely and retrieve information from the victim's device. Some of the information and access obtained in this study can be seen in Figure 8. The victim's device uses Android version 11 and Linux kernel 4.4.283-StromBreaker-X7 (aarch64). Meterpreter runs via Dalvik/android. Dalvik is a software that runs applications on Android devices. Dalvik is an integral part of Android and is commonly used in mobile devices such as smartphones and tablets and with other Android operating systems such as smart TVs and media players.

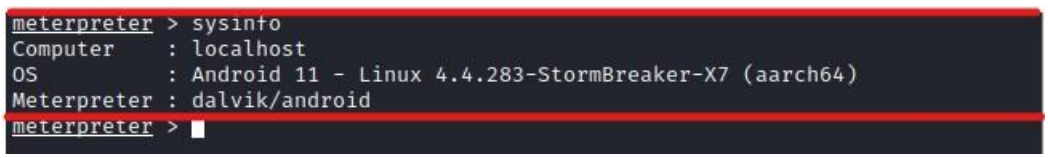

Figure 8. Information System on Android 11

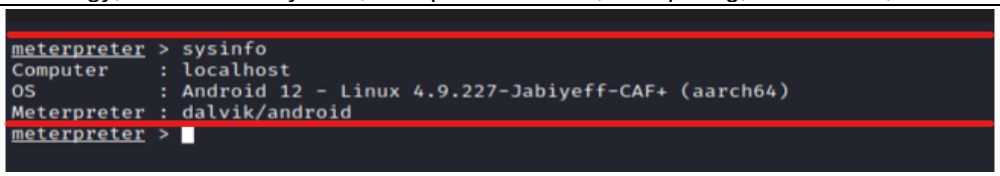

Figure 9. Information System on Android 12

It can be seen in Figure 9 that the system information has been successfully obtained. Some information that appears when the sysinfo command is executed is the operating system using Android 12, the Linux kernel 4.9.227-Jabiyeff-Caf+ (aarch64), and the virtual process on the operating system using Dalvik.

3.4 Dumping Data

This section describes the data obtained in the testing process. In this case, the data in question is test data from the exploit execution process. After carrying out the exploit process, the attacker will gain control of the victim's device and can retrieve the data on the device. The data obtained can be seen in Figure 10, Figure 11, and Figure 12.

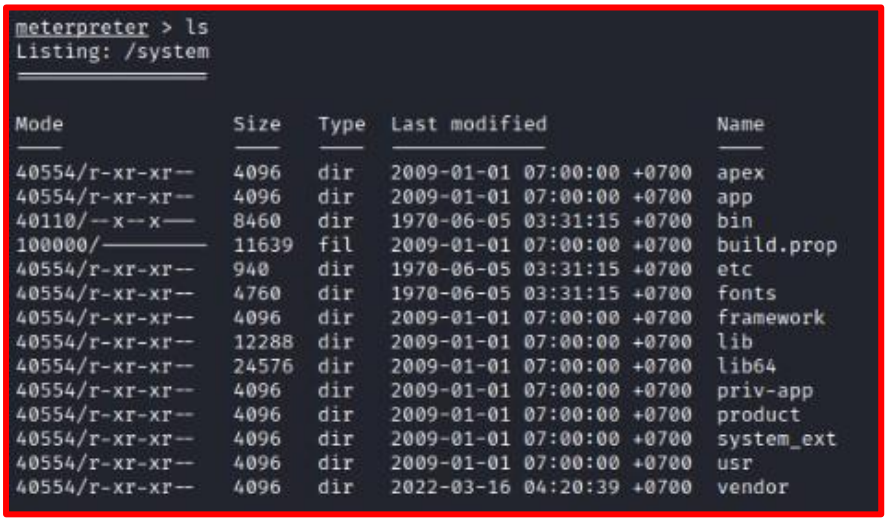

Figure 10. Access the Directory System on the Victim's Device

Figure 10 shows the attacker also managed to access the system directory, allowing him to add or delete files. For example, an attacker could delete a lock screen password.

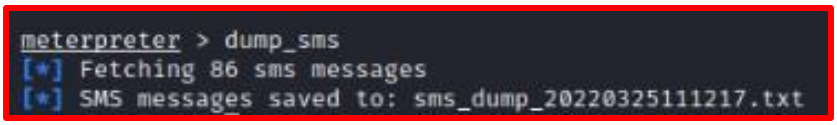

Figure 11. Dump SMS from the Victim's Device

Figure 12. Record Sound Through the Victim's Device

Figure 11 shows that the attacker got the overall message data from the victim's device. Attackers can use this message data to determine with whom the victim exchanged messages and the content of the messages sent and received by the victim. Then, Figure 12 shows the attacker managed to access the microphone so that all sounds that enter through the microphone, both conversations and sounds around the device, will be saved by the attacker. In addition to the data above, what successfully and unsuccessfully accessed several data by attackers. For more details, see Table 2.

Table 2. Test Results on the Victim's Device

Activity	Result	Information
Install the application on Android 11	Success	Successfully installed without warning from play protect.
Install the application on Android 12	Success	Force Install and skip popup warnings that the app is dangerous.
Get system information	Success	Successfully get system information on Android 11 and Android 12.
Open meterpreter session	Success	Meterpreter session opens on Android 11 and Android 12.
Snap webcam	Failed	Booth Android 11 and Android 12 failed when trying to open the front and back camera of the victim's device.
Sending SMS	Success	Message sent successfully from victim's device on Android 11.
Dump contact	Success	Successfully retrieved contact data on the target device on Android 11.
Dump call log	Success	Successfully retrieved call log on victim's device on Android 11.
Access system directory	Success	Successfully accessing the victim's device system directory in full on Android 11.

Table 2 shows some of the attacker's activities on the victim's device. The results are that installing the application on Android version 11 and Android 12 was successfully carried out without problems. Still, on Android version 12, a dangerous warning appeared because the application was detected not using the latest privacy protection technology from Google. Furthermore, the meterpreter session was successfully opened when starting the reverse TCP handler at 192.168.100.235:4444 and sent the stage to 192.168.100.66 on Android 11 and 192.168.100.116 on Android 12, the victim device. When the meterpreter is open, the attacker tries to access the camera on the target device but fails. The table also shows that the attacker attempted to send a message using the victim's device, which was successfully sent. Besides that, the attacker also managed to get contact data and the history of the last call made by the victim.

3.5 research summary

The conclusion that can be drawn from the test shows that on Android 11, the application installation runs without any obstacles. Meanwhile, on Android 12, several protections make it difficult for attackers to be able to remote access. The application installation process on Android 12 requires the attacker to force installation directly on the target device because Android 12 has three protection processes. The first protection will appear as a warning when you make the application in the form of a popup "allow install through unknown sources." A second warning will appear because Google play protect protection detects harmful. The third protection is on Android 12, the USB debugging option is disabled by default so that the attacker cannot connect the ADB server of the target device to the attacker's device; if this is done forcibly, then the attacker can only access the system information because by default Android 12 will block the permissions of apps that have been injected with payload. RAT. Meanwhile, on Android 11, the attacker can access the target device due to a lack of protection, and the USB debugging option on the tested device is active by default.

4. Conclusion

Research on this Android system uses the Metasploit framework to load, exploit, and control the victim's device as a remote access trojan using Android 11 and version 12. Previously research on the Android system using the Exploit method was carried out by Android 7.1.2 Nougat and Just ran the info system on the victim's device. In the study, the Android version used the latest version and managed to execute the device and get data from the victim's device. The process of making the payload, exploitation, and management of the victim's device has been successfully carried out. As a result, the attacker gets some data such as SMS, contacts, and the latest call history and gets full access to the victim's system directory. There are changes to the Security of the 12 version of the Android System where when installing an application that is considered dangerous will appear by the Google Play Protect. The attack process was conducted to test the security of the Android system against remote access trojan attacks. From the simulation results, Android operating system users need to be more careful about remote access to Trojan malware. Further research can strengthen the detection system on Android to identify whether an application is malicious.

References

[1] "• Smartphone users 2026 | Statista.".
[2] W. Khan, M. Kamran, A. Ahmad, F. A. Khan, and A. Derhab, "Formal Analysis of Language-Based Android Security Using Theorem Proving Approach," *IEEE Access*, vol. 7, pp. 16550–16560, 2019. https://doi.org/10.1109/ACCESS.2019.2895261
[3] G. M. Zamroni and I. Riadi, "Instant Messaging Forensic Tools Comparison on Android Operating System," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, no. 2, pp. 137–148, 2019. https://doi.org/10.22219/kinetik.v4i2.735
[4] I. Riadi, H. Herman, and A. Z. Ifani, "Optimization of System Authentication Services using Blockchain Technology," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, 2021. https://doi.org/10.22219/kinetik.v6i4.1325

[5] R. D. Putra and I. Mardianto, "Exploitation with Reverse_tcp Method on Android Device using Metasploit," *J. Edukasi dan Penelit. Inform.*, vol. 5, no. 1, p. 106, 2019. http://dx.doi.org/10.26418/jp.v5i1.26893
[6] R. Singh, "An Overview of Android Operating System and Its Security Features," Eng. Res. Appl., vol. 4, no. 2, pp. 519–521, 2014.
[7] V. G. Shankar, G. Somani, M. S. Gaur, V. Laxmi, and M. Conti, "AndroTaint: An efficient android malware detection framework using dynamic taint analysis," *ISEA Asia Secur. Priv. Conf. 2017, ISEASP 2017*, pp. 1–13, 2017. https://doi.org/10.1109/ISEASP.2017.7976989
[8] P. Bhat and K. Dutta, "A survey on various threats and current state of security in android platform," *ACM Comput. Surv.*, vol. 52, no. 1, 2019. https://doi.org/10.1145/3301285
[9] I. Riadi and D. Aprilliansyah, "Analysis of Remote Access Trojan Attack using Android Debug Bridge," vol. 10, no. 2, pp. 102–111, 2021. https://doi.org/10.14421/ijid.2021.2839
[10] J. Qin, H. Zhang, J. Guo, S. Wang, Q. Wen, and Y. Shi, "Vulnerability Detection on Android Apps-Inspired by Case Study on Vulnerability Related with Web Functions," *IEEE Access*, vol. 8, pp. 106437–106451, 2020. https://doi.org/10.1109/ACCESS.2020.2998043
[11] R. Li, W. Diao, Z. Li, S. Yang, S. Li, and S. Guo, "Android Custom Permissions Demystified: A Comprehensive Security Evaluation," *IEEE Trans. Softw. Eng.*, 2021. https://doi.org/10.1109/TSE.2021.3119980
[12] T. Moletsane and P. Tsibolane, "Mobile Information Security Awareness among Students in Higher Education : An Exploratory Study," *2020 Conf. Inf. Commun. Technol. Soc. ICTAS 2020 - Proc.*, pp. 1–6, 2020. https://doi.org/10.1109/ICTAS47918.2020.233978
[13] F. A. Garba, K. I. Kunya, S. A. Ibrahim, A. B. Isa, K. M. Muhammad, and N. N. Wali, "Evaluating the State of the Art Antivirus Evasion Tools on Windows and Android Platform," *2019 2nd Int. Conf. IEEE Niger. Comput. Chapter, Niger. 2019*, pp. 1–4, 2019. https://doi.org/10.1109/NigeriaComputConf45974.2019.8949637
[14] R. Surya Kusuma, R. Umar, and I. Riadi, "Network Forensics Against Ryuk Ransomware Using Trigger, Acquire, Analysis, Report, and Action (TAARA) Method," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, 2021. https://doi.org/10.22219/kinetik.v6i2.1225
[15] S. Syaifuddin, Z. Sari, and M. K. Masduqi, "Analysis of Uapush Malware Infection using Static and Behavior Method on Android," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 3, no. 1, pp. 81–90, 2018. https://doi.org/10.22219/kinetik.v3i1.265
[16] M. Xu, W. Sun, and M. Alam, "Security enhancement of secure USB debugging in Android system," *2015 12th Annu. IEEE Consum. Commun. Netw. Conf. CCNC 2015*, pp. 134–139, 2015. https://doi.org/10.1109/CCNC.2015.7157959
[17] "Waspada, Pelaku Kejahatan Sebar Trojan Android via Laman Google Play Store Palsu - Tekno Liputan6.com.".
[18] A. Mos and M. M. Chowdhury, "Mobile Security: A Look into Android," *IEEE Int. Conf. Electro Inf. Technol.*, vol. 2020-July, pp. 638–642, 2020. https://doi.org/10.1109/EIT48999.2020.9208339
[19] "Awas, Aplikasi Android Berikut Ini Bawa Virus Trojan | AsiaQuest Indonesia."
[20] D. Zou, S. Wang, S. Xu, Z. Li, and H. Jin, "μVulDeePecker: A Deep Learning-Based System for Multiclass Vulnerability Detection," *IEEE Trans. Dependable Secur. Comput.*, vol. PP, no. c, pp. 1–1, 2019. https://doi.org/10.1109/TDSC.2019.2942930
[21] A. Brusch, N. Nguyen, D. Schurmann, S. Sigg, and L. Wolf, "Security Properties of Gait for Mobile Device Pairing," *IEEE Trans. Mob. Comput.*, vol. 19, no. 3, pp. 697–710, 2020. https://doi.org/10.1109/TMC.2019.2897933
[22] D. C. Prakoso, I. Riadi, and Y. Prayudi, "Detection of Metasploit Attacks Using RAM Forensic on Proprietary Operating Systems," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, pp. 155–160, 2020. https://doi.org/10.22219/kinetik.v5i2.1037
[23] I. Riadi, I. T. Riyadi Yanto, and E. Handoyo, "Cyber Security Analysis of Academic Services based on Domain Delivery Services and Support using Indonesian E-Government Ratings (PEGI)," *Kinet. Game Technol. Inf. Syst. Comput. Network, Comput. Electron. Control*, vol. 4, pp. 263–270, 2020. https://doi.org/10.22219/kinetik.v5i4.1083
[24] X. He, J. Liu, C. T. Huang, D. Wang, and B. Meng, "A Security Analysis Method of Security Protocol Implementation Based on Unpurified Security Protocol Trace and Security Protocol Implementation Ontology," *IEEE Access*, vol. 7, pp. 131050–131067, 2019. https://doi.org/10.1109/ACCESS.2019.2940512
[25] M. Guo, P. Bhattacharya, M. Yang, K. Qian, and L. Yang, "Learning mobile security with android security labware," *SIGCSE 2013 - Proc. 44th ACM Tech. Symp. Comput. Sci. Educ.*, pp. 675–680, 2013. https://doi.org/10.1145/2445196.2445394
[26] T. Rocha, E. Souto, and K. El-Khatib, "Functionality-based mobile application recommendation system with security and privacy awareness," *Comput. Secur.*, vol. 97, p. 101972, 2020. https://doi.org/10.1016/j.cose.2020.101972
[27] M. Wazid, S. Zeadally, and A. K. Das, "Mobile Banking: Evolution and Threats: Malware Threats and Security Solutions," *IEEE Consum. Electron. Mag.*, vol. 8, no. 2, pp. 56–60, 2019. https://doi.org/10.1109/MCE.2018.2881291
[28] D. Jiang and K. Omote, "An approach to detect remote access trojan in the early stage of communication," in *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, Apr. 2015, vol. 2015-April, pp. 706–713. https://doi.org/10.1109/AINA.2015.257
[29] U. Timalsina, "Use of Metasploit Framework in Kali Linux," no. May 2015. https://doi.org/10.13140/RG.2.2.12377.93284
[30] T. Guarda, M. F. Augusto, I. Lopes, J. A. Victor, Á. Rocha, and L. Molina, *Mobile Communication Systems: Evolution and Security*, vol. 152. Springer Singapore, 2020. https://doi.org/10.1007/978-981-13-9155-2_8