National University of Computer and Emerging Sciences, Lahore Campus

TrojanScan: Ensuring Android Device Integrity

Waiza Zainab	Mansoor Tariq	Sana Ahmed Khan
l201253	l201369	l201376
7A	7A	7A

Table of Contents

Table of Figures

***Abstract*— The role of mobile devices, like tablets and smartphones, in people's daily lives is growing. Numerous operating systems provide a platform for intelligent technologies, such as iOS, Android, and others. Google's Android is among the most popular open-source mobile software platforms. Due to their convenience, people are likely to download and install malicious applications to trick users, opening the door for an attacker to compromise security. The rapid proliferation of smart devices and mobile applications has led to the emergence of security attacks. Attacks can take the form of various malware, including Trojan horses and spyware, which take advantage of flaws in operating systems, mainly the widely recognized Android OS. Comprehending Android device vulnerabilities and being informed about preventative measures is essential.**

Keywords—android, malicious, security, vulnerabilities.

Introduction

We are currently in the third decade of digital mobile telecommunication and making steady progress. Advancements, including mobile stations, have impacted every element of telecom. These mobile stations are cell phones that are intended to be used for making phone calls over circuit-switched (CS) networks and for critical text services like Short Message Service (SMS).

Cyberattacks are a constant in the modern cyber world. One of the main concerns for this critical environment is security. The testing phase takes less time as Android applications are produced faster, and security testing is occasionally overlooked. One of the most widely used open-source mobile operating systems, Android, is more vulnerable to attacks. Android apps are, therefore, susceptible to hackers and malevolent users who might obtain access to private data.

There are several malwares, such as spyware. Malware that watches a user's activities is called spyware. These activities could involve private files like chat logs and photos or log in details like username and password. Nevertheless, since spyware is difficult to disseminate, Trojan horses conceal it by giving users access to legitimate functionalities.

Trojan Horse Attack Examples

Three major examples of trojan horse discussed are:

- Trojan Horse through Notification Listener Service
- Trojan Horse through SMS Service
- Trojan Horse through Social Engineering attack CANDY
- Trojan Horse through Hidden Apps

A. Trojan Horse in Notification Listener Service:

An application serves as a trojan horse by asking the users to enter emails for backing up messages.

a) Design:

The user has the option to select between a backup and a personalised response. Every time the application is launched, the user is prompted to enable the "Notification Access" permission from the settings in order to use its features. Once permission has been granted, the user is asked to input their email address in the text field on the application's home screen in order to initiate the SMS backup feature.

The system notifies many Android services that run in the background, like NotificationListener and smsReceiver, when a new notification is posted or an SMS is received, respectively. Both services start only after the user grants "Notification Access" to the application. The "smsReceiver" service's onReceive() method, which notifies the user via email when the fresh SMS content is extracted, implements the backup functionality.

The user can receive an SMS notification with additional features by creating a duplicate within the "onReceive" method and deleting the original notification within the cancelNotification() method of the "NotificationListener" service.

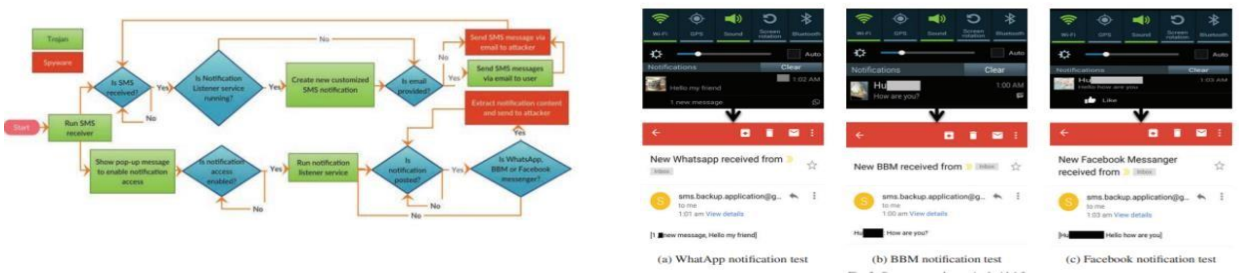
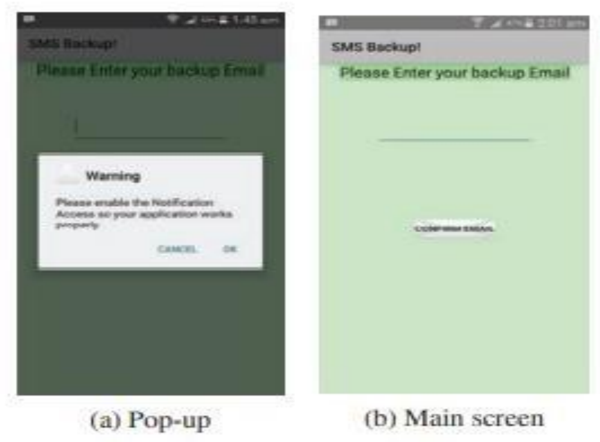

Trojan
Spyware
Start
Run SMS receiver
Is SMS received?
Yes
No
Is Notification Listener service running?
Create new customized SMS notification
Is email provided?
Send SMS message via email to attacker
Send SMS messages via email to user
Extract notification content and send to attacker
Show pop-up message to enable notification access
Is notification access enabled?
Run notification listener service
Is notification posted?
Is WhatsApp, BBM or Facebook messenger?
New Whatsapp received from
New BBM received from
New Facebook Messanger received from
(a) WhatApp notification test
(b) BBM notification test
(c) Facebook notification test

Figure 1

b) Spyware Activity Model:

The attack activities target BBM, Facebook Messenger, and Whatsapp notifications while they run in the background by using the "NotificationListener" service. Any posted notification regarding one of the applications that are being targeted has its content extracted and forwarded to the attacker's email address. The message extraction is done in the "onNotificationPosted" method by parsing the "bigContentView" or "TickerText" fields of the notification view. The "bigContentView" field contains the message's content, and the "TickerText" field contains its header. Messages from WhatsApp and Facebook Messenger can be extracted by parsing "bigContentView," and messages from BBM can be extracted by parsing "TickerText."


SMS Backup!
Please Enter your backup Email
Warning
Please enable the Notification Access so your application works properly
CANCEL
OK
CONFIRM EMAIL
(a) Pop-up
(b) Main screen

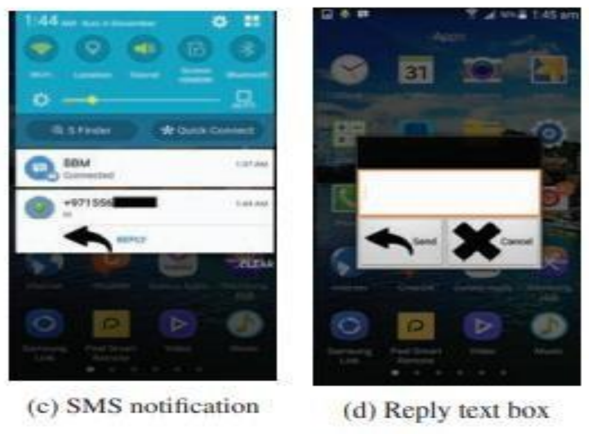


Figure 2

Figure 3

c) Trojan Views:

When the user launches the application for the first time, a pop-up window (see Figure 2a) appears, asking him to enable notification access. Clicking the "OK" button takes him to the notification access settings page. After granting notification access, the user will see the application's main screen, as seen in Figure 2b.

An EditText element on the main screen allows the user to enter his backup email address by accepting input as an email address and rejecting all other input. Furthermore, the newly displayed notification, as seen in Figure 2c, features a reply button that, when clicked, opens the reply field dialogue. By clicking the "send" button in the reply dialogue, the EditText-containing reply dialogue (Figure 2d) appears, allowing you to write and send a reply without ending the running application.

B. Trojan Horse in SMS Service:

The application appears typical, with the basic functionality of sending and receiving SMS. In order to specifically target the credit transfer through the SMS service, malicious code has been added to the application. To accomplish this, the application needs three service components and at least one activity.

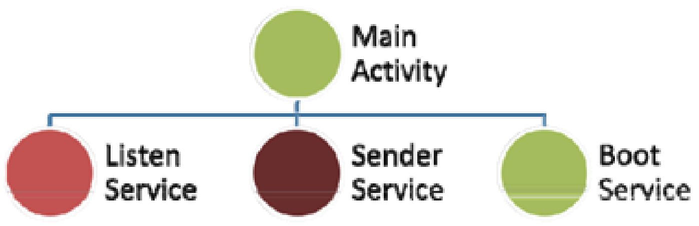


Figure 4

a) Components:

- **Main Activity:** The full graphical user interface for reading and sending SMS messages is provided by the Main Activity component. Android phones can insert and read SMS messages to the content provider content://sms/ using database queries. Moreover, this part is the basic component that, at least initially, starts the Listen Service and the Sender Service. On the other hand, after the main activity was first launched, the Boot Service keeps running on its own.
- **Listen Service:** This part watches for incoming SMS messages and responds according to preset rules. It is necessary to register this service as a broadcast receiver. This component is notified when it gets an SMS message over the radio. New messages are reviewed and verified. If the recently received message does not acknowledge the "illegal" credit transfer, the component either instantly adds it to the messaging database or allows it to travel through unmodified. This notification will be suppressed and never reach the database or any other programme if it is linked to malicious activity.
- **Sender Service:** This part manages the SMS application's unapproved sending procedure. This section controls the unauthorised sending process for the SMS application.
- There is no noise generated by this service. This is made possible by the ability to do malicious activities, like credit transfers, at random, widely separated time instants. This makes the attack unpredictable and undetectable to an ignorant bystander watching the operator use a phone. The user won't be able to tell when the transfer happened by looking at its messages because this component will stop them from being kept in the messaging database. Similar to the Listen Service, this one is sticky and will reload automatically even if the user intentionally quits it.

  Additionally, this service can monitor the user's activity level and then make fraudulent transfers during the times when the user is most active, such as when they are making phone calls or sending or receiving SMS messages. The likelihood that the user will notice the credit loss will decrease as a result. Boot Service This component is required for the above listed services to function when the OS is launched. Registering as a broadcast receiver for BOOT_COMPLETED events is how this is accomplished.
- **Permissions:** The "RECEIVE_SMS" and "SEND_SMS" permissions that SMS applications request are the bare minimum required to perform malicious actions. When this work was written, the most widely used SMS apps on the Android market also used the "READ_SMS" and "WRITE_SMS" permissions.

  As a result, the user would not be made aware of the malicious activity by requesting these permissions, which is not unusual.

b) Implementation:

The programme is installed on the victim's Samsung Galaxy SII phone. This phone comes with Android OS 2.3.5 (Gingerbread) pre-installed and has a kernel version of 2.6.35. All transferred credits, however, are stored on the attacker's SIM card, which is kept in the Sony Ericsson. In the experiment, a message with a predefined format is constructed in order to send credit. This format corresponds to one of the operators in the experiment and is operator-specific. Usually, the message is sent to a distinct 4-digit number. The credits (amount in US dollars) are added to the recipients' balances and deducted from the sender's upon message delivery. In the end, a notification informing the sender and the receiver that the credit transfer has been completed successfully is sent. It's crucial to keep in mind that this transfer from several carriers may come with transaction fees. Therefore, we believe that most operators would not financially discontinue this service even in the face of security concerns.

- All messages related to this operation are purposefully muted because the victim must remain uninformed of the transaction. The transfer was taking place, as evidenced by the "main" buffer's output.

C. Trojan Horse through Social Engineering attack CANDY

An Android in-car infotainment system, which gathered data about the car and its occupants, was the target of the CANDY social engineering attack.

An Android Infotainment Radio with an Android 4.4 KitKat is the intended system. The radio is mounted on a car and is linked to the CAN bus network, also called the Controller Area Network, utilizing a CAN bus decoder. This enables the radio to display various vehicle data via appropriate APPs.

a) Design:

A 3G dongle is used to connect the radio to the Internet. The driver can access features similar to those found on smartphones thanks to the constant Internet connection, which also ensures high coverage.

CANDY is a collection of remote attacks carried out via social engineering that take advantage of an Android app that we created, developed, and disseminated. In the Android In-Vehicle Infotainment system, the App Android functions as a Trojan horse. We selected the "Gas Station Finder" app, which displays gas stations near cars, as an enticing Trojan horse app for the victim. We added more code to the Trojan Horse app intended to steal driver data.

When the Trojan-horse APP is used, the infotainment system's microphone simultaneously records all conversations inside the car and opens a backdoor. The attacker uses a remote shell embedded in the malware payload to download the files to retrieve the recordings, GPS coordinates, and images.

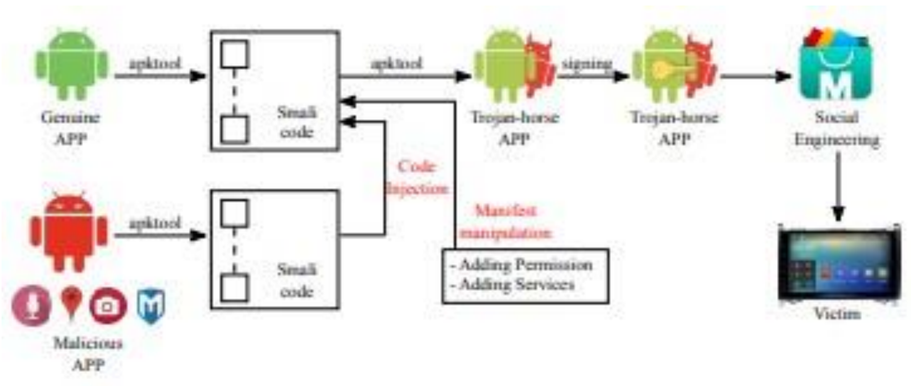


Figure 5

b) Running the attack:

The controlling platform is built to accept connections from the Trojan Horse APP and to spawn a shell that gives it access to the target device, all with the aid of Metasploit.

In order to take control of the intended entertainment system, the attacker configures Metasploit to accept incoming connections. Consequently, Metasploit is set up as follows:

- exploit -j: this command launches the handler; • set ExitOnSession false: the handler will remain in listening mode even if a connection is closed. Every connected session is kept running in the background by the -j option.
- Change the payload to android/meterpreter/reverse_tcp; it needs to match the payload from the malicious APP.
- Set the LHOST IP to the IP address of the attacker's computer, which is where Metasploit is installed.
- To instruct Metasploit to handle multiple sessions, use exploit/multi/handler.
- set LPORT Port: this option sets the handler's port of execution.

The attack can be leveraged once the Trojan-horse APP is set up and operational on the intended device. Metasploit now displays a line of text that reads, "Meterpreter session X opened."

Then, using the following command to access the infotainment system, the attacker launches a Meterpreter shell: sessions IX.

In this case, X represents the newly created session ID. When an attacker uses a Meterpreter shell, which functions similarly to a Unix-like command-line shell, they can execute commands to remotely investigate the infotainment system and carry out other nefarious operations, like uploading and downloading files to and from it. The attacker can use a variety of attacks—which we go over in more detail in the following section—to harm the victim once she has access to the infotainment system. We can take both the car and the driver's information.

D. Trojan Horse through Hidden APPS:

HiddenApp operates as a Trojan, surreptitiously installing additional applications or libraries on a user's device without their knowledge or consent. Frequently, these payloads harbor other malware or adware.

a) Infection:

HiddenApp employs camouflage tactics, masquerading as a legitimate app or adopting the guise of a fake one to conceal its true purpose. These deceptive applications are commonly discovered in third-party file shares and app markets. Additionally, Trojan can install itself as a Device Administrator, enabling it to install apps silently without user intervention.

b) Effects:

The onset of infection transpires when a user unwittingly installs an app containing malicious code. These infected apps, while outwardly appearing as innocuous games or productivity tools, clandestinely execute a payload in the background.
This multifaceted strategy allows HiddenApp to permeate various layers of user trust, exploiting both the allure of legitimate apps and the obscurity of fake ones, thereby posing a significant threat to the security and integrity of the user's device.

Detection and Prevention

This section delves into methodologies for detecting and preventing Trojan horses, focusing on robust security measures.

A. Detection through Event Monitoring:

Event Notification:

Events are generated to alert users when there is an attempt at communication, such as via email, SMS, Bluetooth connection, or file transfer. Users are promptly notified through an alert within the phone's graphical user interface, often accompanied by an audible signal.

Stealth Measures:

Trojans, especially in their endeavor to remain undetected, employ strategies to bypass alarms and user-alert mechanisms. The intent is to leave no trace of their activities. A notable example is the Neo-Call spyware, which allows surreptitious SMS messages without user knowledge and subsequently ensures that these messages remain concealed within the phone, escaping detection in SMS inboxes or outboxes.

Leveraging Low-Level Events:

Malicious operations will always result in low-level occurrences, even with best efforts to be covert. Through efficient surveillance of these occurrences, suspicious activities—including those intended to remain undetected—can be recognized. The development of an events-based technique to identify Trojan horses, both known and unknown, benefits greatly from this information.

B. Basic Client Design:

Monitoring Module Components:

The fundamental structure of the proposed method comprises two pivotal event monitoring components: the User Interface and Monitoring modules. Clients interact with the User Interface to gain insights into the monitoring state, including established connections, used ports, and recent SMS, MMS, and Bluetooth file activities.

User Interaction and Anomaly Detection:

Through the User Interface, users can approve or reject connection requests based on anomalies detected by the Monitoring Module in the provided, received, or requested data. If approved, the Monitoring Module issues commands to the operating system for connection or file delivery.

Connection and File Submodules:

The two submodules that make up the Monitoring Module—Connection and File—are responsible for handling connections and data transfers. The Connection Submodule manages the protocols that need to create connections and keeps track of and reports on erroneous connection attempts. By analyzing frequency and application source paths, the File Submodule carefully examines messages and files to spot any infections.

Real-Time Alerts:

Upon detection of a potential infection, real-time alerts are communicated to users through the User Interface module. These alerts include sender number, port, and program path for connection attempts, enabling users to decide whether to accept or deny.

Handling Accepted Connections or Files:

The Monitoring Module gives up control to the operating system to create relationships or organize files into specific directories if a connection or file is approved. Decisions and relevant information are logged into a text file for future reference.

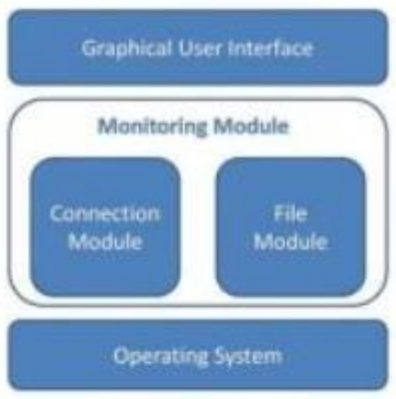


Figure 6

The File Submodule is tasked with managing messages and files, overseeing both their frequency and the source (path) of applications transmitting these files. In the context of a Trojan horse attempting to spread to other devices, the submodule's first feature can be configured to identify such attempts by analyzing the interval between two comparable events. For example, it is atypical for a user to send numerous SMS in a short time frame, given the time-consuming nature of composing messages and looking up contacts. The second feature scrutinizes an application's installation path, recognizing that some malicious programs deviate from default folders to avoid detection. Configuration parameters, including the shortest time between events, default installation paths, and the names and folders of approved Bluetooth and messaging applications, are housed within the File Submodule.

The Monitoring Module employs these submodules to detect files potentially harboring a Trojan horse infection. Consequently, when a file is either sent or received, the File Submodule triggers an event and checks for possible infections. Consider a background-running program employing this strategy. Upon the arrival of a message or connection request at the phone, the program creates an event, initiating an evaluation by the Connection Submodule or File Submodule using the Monitoring Module. If a potential infection is detected, a real-time message is promptly relayed to the user through the User Interface module, displaying connection or file details such as sender number, port, and program path. In the case of refusal, a connection request is not established, and a denied file, like an SMS or MMS, won't reach the inbox or outbox.

Ultimately, upon acceptance of a connection or file, the Monitoring Module relinquishes control to the operating system to establish the connection or place the file in the appropriate folder. The conclusive decision, along with information pertaining to the message or file, is recorded in a text file for future reference. Additionally, users possess the capability to edit and modify this file.

Refer to the accompanying flowchart for a visual representation of the sequence of events when a user attempts to connect or send a file.

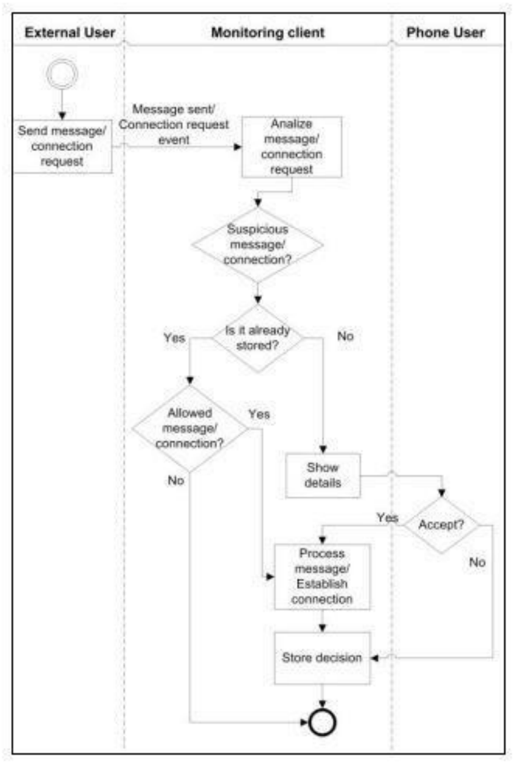

External User
Monitoring client
Phone User
Send message/ connection request
Message sent/ Connection request event
Analize message/ connection request
Suspicious message/ connection?
Is it already stored?
Yes
No
Allowed message/ connection?
Yes
No
Show details
Accept?
Yes
No
Process message/ Establish connection
Store decision

Figure 7

C. Dynamic Code Loading in Android Applications:

a) High-Level Overview:

Code Integrity Verification:

Recognizing the inherent security challenges in allowing application developers control over system security, our protection system introduces a rigorous code verification method that Android needs to improve. This mandates that all applications undergo code integrity checks before execution, mitigating threats associated with the ability to load external code.

Application Verifiers:

Several application verifiers, employing diverse techniques and criteria, are envisioned to examine apps. Once deemed benign, an application validator generates an equivalent of a signature for the application, publicly accessible to users. This decoupling of application verification from stores empowers users to select verifiers aligning with their priorities.

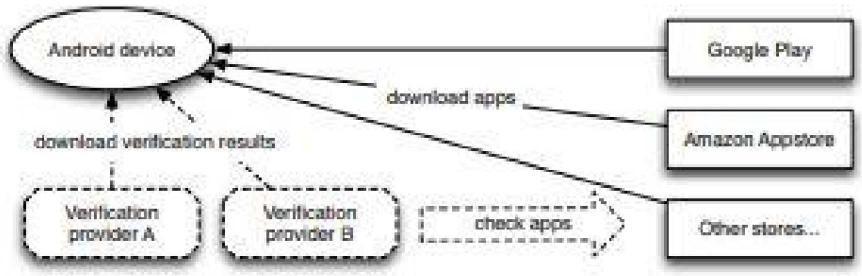


Figure 8

Advantages of Design Choice:

This design choice presents several advantages. Application verifiers can tailor their focus to distinct sets of criteria when evaluating applications, acknowledging the diverse priorities of individual users. For instance, businesses may employ different assessment standards for apps on work devices compared to private users managing apps on their phones or tablets. Users retain the flexibility to choose verifiers based on their specific preferences.

A notable distinction from the iOS environment is the absence of dependence on a single validator. In contrast to iOS devices, which typically only run software approved by Apple, Android users enjoy greater freedom in selecting validators that align with their trust preferences.

Importantly, our approach ensures consistent security for all apps, irrespective of the store distributing them. This is achieved by decoupling application verification from specific stores. Android now classifies apps into two modes:

- Allowing only apps from the recommended store.
- Allowing any application.

Consequently, users relying on stores other than those favored by the device manufacturer, often Google Play, face a security risk. Users of alternative stores are compelled to keep their devices open to all APKs, regardless of the source. However, our proposed method provides equal protection for customers across various retailers, focusing on evaluating specific binaries rather than stores.

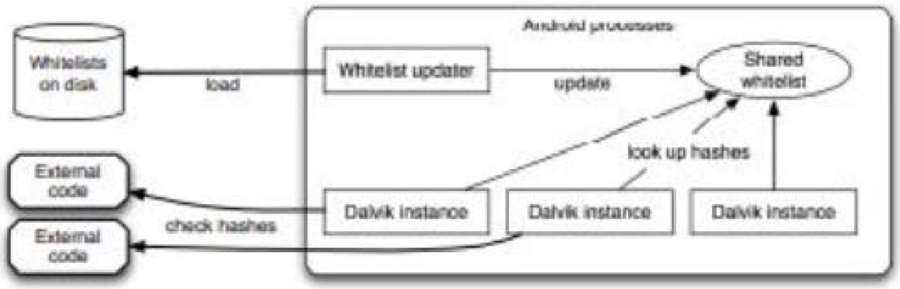


Figure 9

Integration with Java Virtual Machine (JVM):

Our approach aligns with the JVM's use of signatures for code verification. While the JVM trusts code from a developer with a certificate, our method demands verification checks before trust is established, elevating the security posture.

Comparative Analysis

This section explores two methodologies for preventing and detecting Trojan horse malware within a device. The first technique involves monitoring SMS and Bluetooth activities to identify malicious actions, such as transmitting the user's data to a third party or granting access to the user's device via Bluetooth. The second technique introduces a verification system designed to thwart the installation of malicious applications or the loading of harmful code from external sources. A set of verifiers is employed to assess the evil nature of an application or code.

Conclusion

This paper delves into the various methods through which Trojan horse malware infiltrates Android applications, executing activities that compromise user data security. Subsequently, preventive techniques are explored to fortify applications against malicious actions. The first technique involves monitoring SMS and Bluetooth to detect Trojan horse activities, such as data transmission or unauthorized device access via Bluetooth. In the second technique, a verification system is implemented to prevent the installation of malicious applications or the loading of external code.

While both techniques are equally effective in preventing and detecting Trojan horse malware within a device, the comparative Analysis slightly favors the second technique for its proactive approach to malware detection and prevention. This research contributes valuable insights into combating Trojan horse threats and reinforces the importance of proactive security measures in safeguarding digital ecosystems.

REFERENCES

[1] J. A. Ortega, “A novel approach to trojan horse detection in mobile phones messaging and bluetooth services,” KSII Transactions on Internet and Information Systems, vol. 5, no. 8, 2011.

[2] S. Poeplau, Y. Fratantonio, A. Bianchi, C. Kruegel, and G. Vigna, “Execute this! analyzing unsafe and malicious dynamic code loading in Android Applications,” Proceedings 2014 Network and Distributed System Security Symposium, 2014.

[3] K. Hamandi, A. Chehab, I. H. Elhajj, and A. Kayssi, “Android SMS malware: Vulnerability and mitigation,” 2013 27th International Conference on Advanced Information Networking and Applications Workshops, 2013.

[4] H. Abualola, H. Alhawai, M. Kadadha, H. Otrok, and A. Mourad, “An Android-based trojan spyware to study the NOTIFICATIONLISTENER service vulnerability,” Procedia Computer Science, vol. 83, pp. 465–471, 2016.

[5] A. Amin, A. Eldessouki, M. T. Magdy, N. Abdeen, H. Hindy, and I. Hegazy, “AndroShield: Automated Android Applications Vulnerability Detection, a hybrid static and dynamic analysis approach,” Information, vol. 10, no. 10, p. 326, 2019.

[6] Costantino, G., La Marra, A., Martinelli, F., & Matteucci, I. (2018). CANDY: A Social Engineering Attack to Leak Information from Infotainment System. 2018

[7] “Android/trojan.HiddenApp: Malwarebytes labs,” Malwarebytes. [Online]. Available: https://www.malwarebytes.com/blog/detections/android-trojanhiddenapp..