

REMOTE MALWARE DETECTION USING PATTERN BASED ANALYSIS FOR ANDROID DEVICES

Afrah Fathima Assistant Professor Department of Computer Science and Information Technology,
Maulana Azad National Urdu University, Gachibowli, Hyderabad Telangana

M. O.Nadeem Project Engineer Centre for Development of Advanced Computing, Hyderabad,
Telangana

Abstract

Android has conquered the smartphone industry and is the most widely used mobile operating system. However, as Android's popularity has grown, so has the number of security threats in Android apps. As a result, the malicious attackers have been drawn towards it. In comparison to other mobile phones, statistics suggest that Android-based smart phones are more susceptible to malware. According to the data, the total amount of malware attacks against Android operating systems was about 5.6 billion in 2020. And the fact that constant attempts have been made to counter these risks. Several identification methods and technologies have significant drawbacks. There is no one solution that will guarantee Android malware security. There is a lot of research being done on developing malware detection tools for Android devices. There is no research done on scanning an Android device remotely. This research aims at developing a tool Remote Malware detection(RMD) tool which focuses in identifying the hidden malware's in an Android mobile phone and also aim to scan a device remotely. This paper also aims at experimenting remote malware detection of an Android device located with in the same network or located remotely using simple python programming

Keywords: Android, Malware Detection, Remote Malware

Introduction

Android has conquered the smartphone market and is now the most widely used mobile operating system. Security concerns in Android applications, on the other hand, have grown in lockstep with Android's growth. Despite the fact that constant research efforts have been made to address these dangers, there were more than 3 million new malware samples targeting the Android OS were found in 2017[1]. Malware has evolved in recent years by employing various encryption tactics as a result of this evolution, malware detection has become difficult. Neither signature-based nor classic behavior-based malware detectors is able to detect these malware's[2]. It's a well-known fact in computer security that the user often unwittingly aids the infection on his system. Android's architecture is largely in accordance with the principle of least privilege, as stated by Saltzer and Schroeder in their seminal 1975 paper [3], which requires that an application have only the most restrictive set of permissions possible while still allowing it to perform its intended task. However, it is ultimately up to the individual user to decide whether or not to install an app and allow which permissions to be given. When users install an app, they are presented with a screen that describes what information and system resources the app has access to, eg. phone's GPS location. The user has to explicitly authorize this access in order to proceed with the installation, and they even have the option to uninstall applications immediately or at any moment of time[4]. They can also look at ratings and reviews to assist them to determine which apps to install. It is also advisable only to install trustable nevertheless, while deciding whether or not to install an app on their device, the user is frequently forced to act more on instinct than on data. While the user has access to the rights sought by the app, he may be unaware of the numerous ways in which those rights might be abused to undermine the integrity, and confidentiality of his data. Users that download software from third-party marketplaces face even greater dangers, as these sites commonly contain lot of vulnerabilities.

For most people, antivirus software remains the only and the last line of defence. In fact, some of the anti-viruses examined were unable to detect even a single dangerous programme in the testing dataset. Even some of the worst Anti-virus software's carry malwares. Malware is defined as

software that performs malicious actions on a victim's computer. To make analysis and detection more difficult, sophisticated malware employs packing and obfuscation techniques [5]. The amount of malware attacks, their sophistication, and the economic damage inflicted by malware have all increased significantly in recent years. Malware for mobile devices is on the rise As per the scientific reports around 1 million malware files are created everyday. Cybercrime is expected to cost the global economy \$6 trillion per year by 2021, according to cybersecurity ventures and Ransomware malware costs roughly \$11.5 billion globally, according to the same research from 2019 [6]. As per the McAfee mobile threat report, backdoors, and banking Trojans for mobile devices have increased significantly [7].

Between 2016 and 2017, the number of new mobile malware variants surged by 54% [8], while the majority of unknown and mobile malware are evolved copies of existing malware [9]. Additionally, malware attacks involving the healthcare industry, IOT applications, cloud computing, social media, and cryptocurrency are on the rise . It's nearly impossible to come up with a method or system that can detect every new type of sophisticated malware.



Figure No.1: Types of Malware attacks

Malwares are classified in four types

- **Signature Based Malware Detection:** It examines the features that encapsulate the program's structure and uniquely identify the malware. Detects known efficiently, not detect unknown malware.
- **Behavioral Malware Detection:** It observes program behaviors using monitoring tools and determines whether the program is malware or benign. Although program codes change the behavior of the program will remain relatively the same: thus new malware can be detected with
- **Heuristic Malware Detection:** These methods are compare detection methods that apply both experience and different techniques such as rules and machine learning techniques. However, even if the heuristic technique can detect various forms of known and unknown malware it cannot detect new malware that is quiet different from existing malware.
- **Model Checking Detection:** Checking Detection Malware: Behaviors are manually extracted and behavior groups are coded using linear temporal-logic (LTL) to display a specific feature. Although model checking-based detection can successfully detect some unknown malware that could not be detected with previous malware detection methods.

Literature Survey

To detect Android malware, Naway and LI [7] offered an overview of static, dynamic, and hybrid analysis utilizing deep learning approaches. They described the procedures in full, including their advantages and disadvantages. Alzahrani and Alghazzawi [8] conducted another study on Android

malware detection. They studied eight articles on deep learning for Android Ransomware2 detection and analysis. Rubiya and Radhamani [9] investigated nine Android malware detection techniques and identified their strengths and drawbacks. Yan and Zheng investigated dynamic mobile malware detection in their study [10]. Previous studies on detecting malware in smartphones were evaluated, synthesised, and compared.

Arshad et al. [11] explored at static and dynamic strategies for recognizing and mitigating against Android malware in another study. The strategies examined are categorised based on the detection mechanism employed 2. Code clone detection technologies are used to detect malware. Chen et al. [12] investigated at how to utilise a code clone detector to find known dangerous Android apps. They made use of statistics. They successfully detected malware by examining the source code of the applications using static analysis. This method allows dangerous apps from certain malware samples to be located quickly and accurately. Indeed, 95 percent of previously known malware was found using a dataset of 1170 dangerous programmes from 19 different malware families. Potharaju et al. [13] aim to detect malware-infected repackaged programmes (which they refer to as duplicated programmes) at various levels of obfuscation. The authors created the following three approaches to detect such applications.

Liu et al. [14] offer NSDroid, a tool based on the assumption that malware families exhibit a high level of code resemblance. The programme uses androgexf [14] to extract the call graph from the apps. The information is then further abstracted by NSDroid, which creates a signature for each app. Zhou et al. [15] wanted to find and analyse repackaged apps in a systematic way. They created DroidMOSS, an application similarity measurement framework that uses a fuzzy hash approach to discover and detect changes. They created DroidMOSS, an application similarity measurement framework that uses a fuzzy hash approach to discover and identify small in an application's behaviour. DroidSieve (version 2.1) Suarez-Tangil et al. [18] introduced the DroidSieve technique. In order to detect and categorise Android malware, it looks at numerous syntactical properties of the apps. The list of API calls made in the code, the permissions it seeks, and the collection of all application components are instances of totally static features. They serve as the foundation for a thorough examination of the application in order to find distinguishing traits. Qiao et al. [17] introduced a malware detection system on automatic learning of Android App permissions including API function calls. To begin, this method checks the AndroidManifest.xml file to determine the permissions that the application requires. However, the authors point out that this may be an excess of privileges actually used by the application. To characterise the behaviour of Android apps, Jie Wu et al. [18] introduced DroidMat, a program that utilizes numerous aspects of static information, such as permissions, objectives (messaging objects that hold information about other modules), and API calls. In the case of API calls, their model comprises not only the API calls themselves, but also the type of component (service, activity) in which the API is called. In the presence of dynamic loading and reflection, static analysis may be insufficient to detect malware. As a result, it should be accompanied with a dynamic examination, particularly since malware was discovered in Android apps in a recent survey. Dynamic loading was commonly used in the programme during the repackaging process, according to khanmohammadi et al. [19], potentially to mask the existence of malware.

Khanmohammadi et al. investigated Androzoo, a widely known dataset of Android malware, and discovered that the network address was hidden in most of the samples using dynamic loading. Sarma et al. [20], in order to improve the existing permission-based detection mechanism, created an alarm system that considers both the permissions requested by the app, the app's category and sub-category (as stated in the Google Store), as well as the permissions requested by other apps in the same category. Peng et al. [21] employ a probabilistic model to provide a risk score to Android applications based on the rights they request and the category they belong to. Rather than a classification system that categorises each application as dangerous or benign. Peng et al. @seek to offer the user with an useful rating that captures the probability of the app being malicious, with

higher values reflecting that the app is more likely to be hazardous. Each user can then make an informed decision about whether or not to install the app based on the risk-reward trade-off.

Statement of the Problem

Android is the most popular mobile operating system and has dominated the smartphone market. However, the amount of security risks in Android apps has increased along with Android's popularity. Statistics indicate that Android-based smart phones are more vulnerable to malware than other mobile devices. Additionally, ongoing efforts have been undertaken to mitigate these hazards. Numerous identifying processes and technologies suffer from serious shortcomings. Android malware security cannot be ensured by a single solution This study aims at developing a tool Remote Malware detection(RMD) tool which focuses in identifying the hidden malware's in an Android mobile phone and also aim to scan a device remotely and perform an experiment on remote malware detection of an Android device located with in the same network or located remotely using simple python programming

Objectives of the study

- To developed a Remote Malware detection (RMD) tool using ADB for Android mobile devices.
- To identify the hidden malwares in an Android device using Remote Malware Detection(RMD) Tool.
- To demonstrate the effectiveness of this tool which shows that RMD can detect more hidden malwares which even an Malware detection tool cannot.
- To perform a practical implementation of local scanning of a device with in the same network and remote scanning of an Android device.

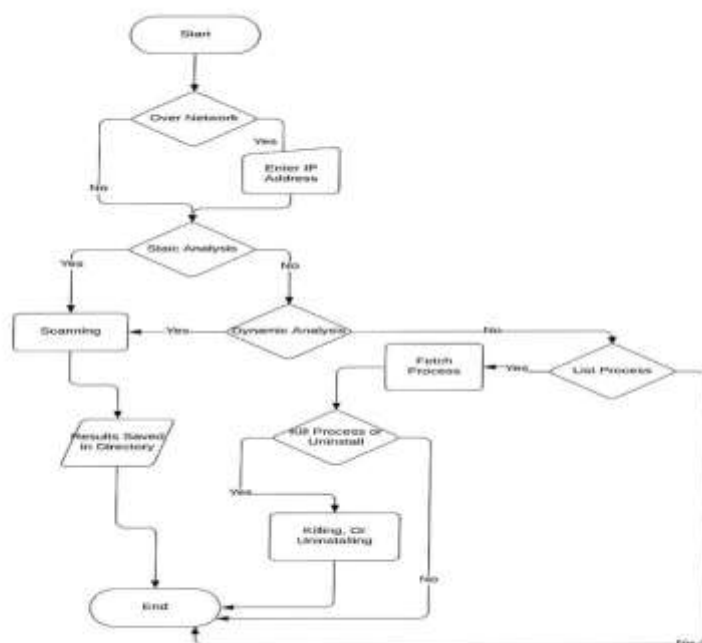


Figure No.2: Flow chart for the proposed work

Proposed Work:

S.No	Features	Description
1.	Known & Unknown Malwalre	ActionSpy WolfRat Anubis, DSenCrypt, List dangerous permissions

		and revokes them Compare with permissions used by malware. generate report in JSON format
2.	Detection Rate	90%
3.	Accuracy Rate	100%
4.	False Prediction Rate	10%
5.	Data Sets(With Score&Without score)	https://github.com/ashishb/android-malware
6.	Samples checked Malwares Benign	Anubis, WolfRat, AndroidSpy, DSenCrypt
7.	Analysis performed	Stattic and dynamic analysis can be used.

Table 1. Remote Malware Detection Features

Architecture of the Proposed Model.

Figure 3 depicts the architecture of the suggested malware detection model. After collecting and analysing programme samples with our RMD Tool, the behaviour is determined based on the results of the analysis. Next, behaviours are categorised using the established rules, and characteristics are extracted. Finally, the most relevant characteristics are chosen and paired with one another. ADB and Androguard library, pwntools are used in this study, The proposed system is implemented using the Python scripting language, and classification is done on Based on malicious patterns. To demonstrate the efficacy of the proposed methodology, live testing is carried out on an Android device, and malware is scanned. However, different tools can be used to improve the findings of the suggested model. As a result, the suggested model's implementation does not impose any limitations on RMD Tool. The Android Realme X3 Superzoom is used for testing.

Remote Malware Detection over Network Scanning for Malware Scan

The following Steps should be followed at the User Side→

Step.1: Enable USB debugging in mobile phone from Developer option. And connect your phone to PC.

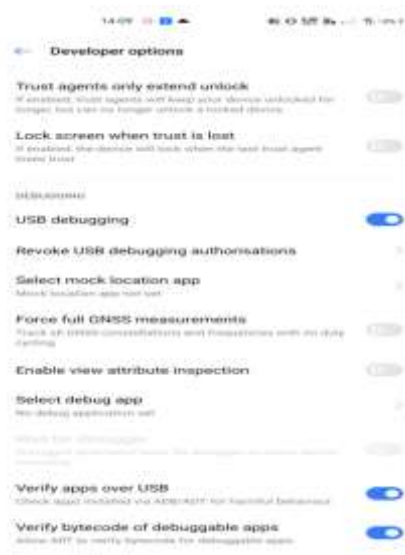


Figure No.3: USB Debugging Setting to enable in phone

Step2:.Go to the overNetwork Module in your computer and Run overNetwork.bat

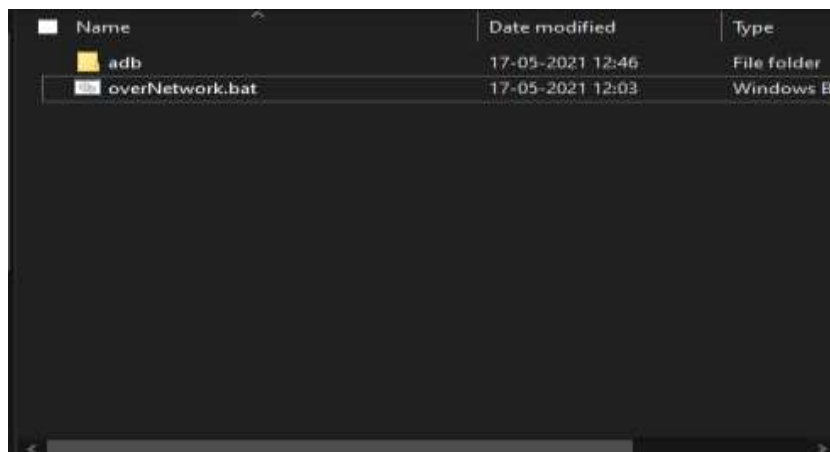


Figure No.4: Over network Module

Over Network module its turns on the ADB on network mode and forward the port to 5555 and it shows the Network Ip address to the user..

Step3: After running overNetwork.bat, you will see two IP address, Note down the second one, and Send it to the person who is going to scan for malware.



Figure No.5: Output window after running overNetwork.bat

Remote Malware Detection at the Remote Side for Malware Scan :

Step1: Go to DroidMalware Folder and open CMD

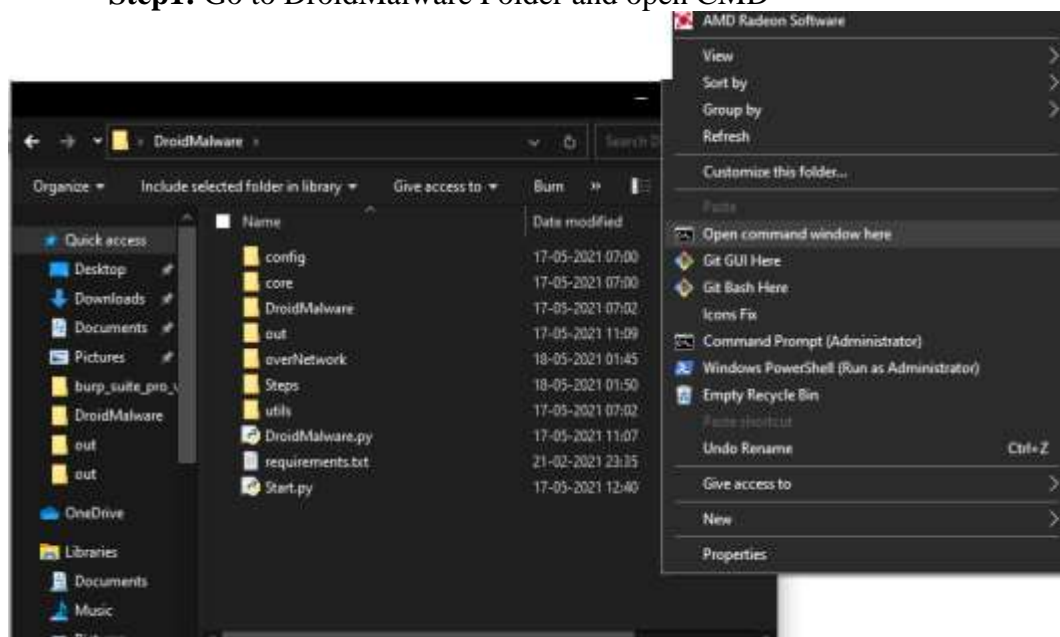


Figure No.6: RMD Tool root directory, and opening cmd

Step2: Type “python start.py” and hit enter , and Enter 1 as choice.

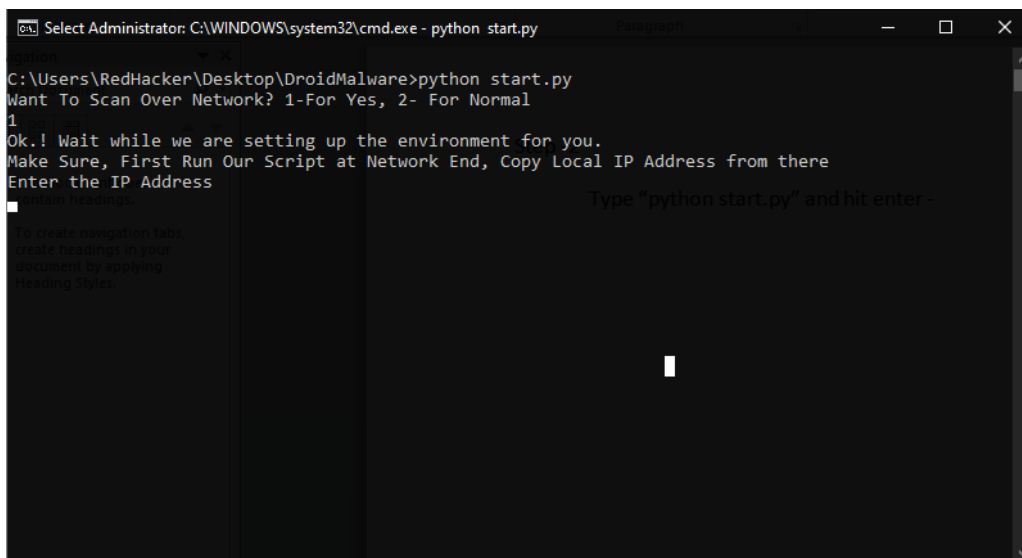


Figure No.7: After running start.py Output window

Step3: Type the IP address, you got from USER by overNetwork module. After entering it will show connected, and will open new CMD window.

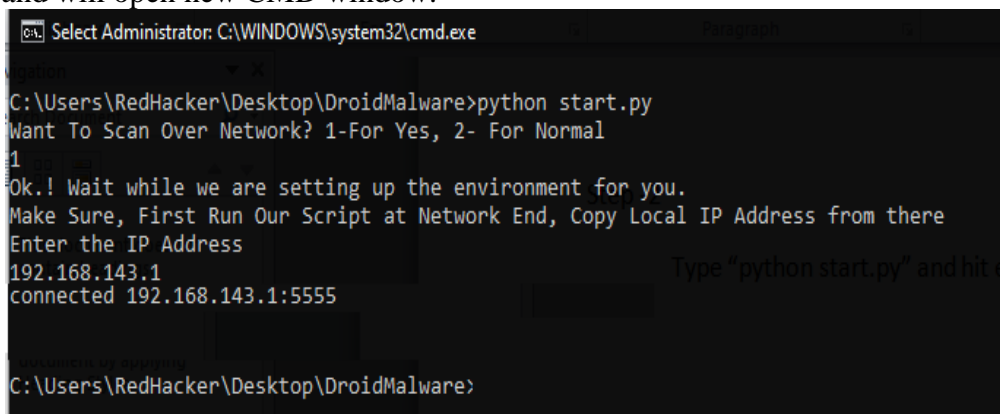


Figure No.8: After entering IP address, output showing as connected

Step4: Now you are at Scanner Portal. Type “python DroidMalware.py -h” for help and commands for how to scan.

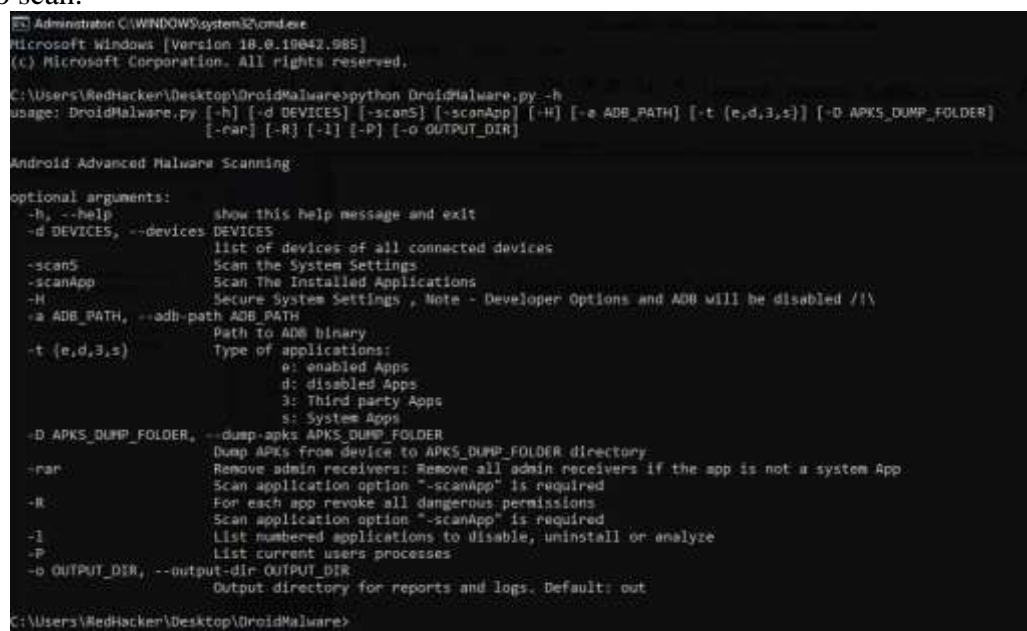
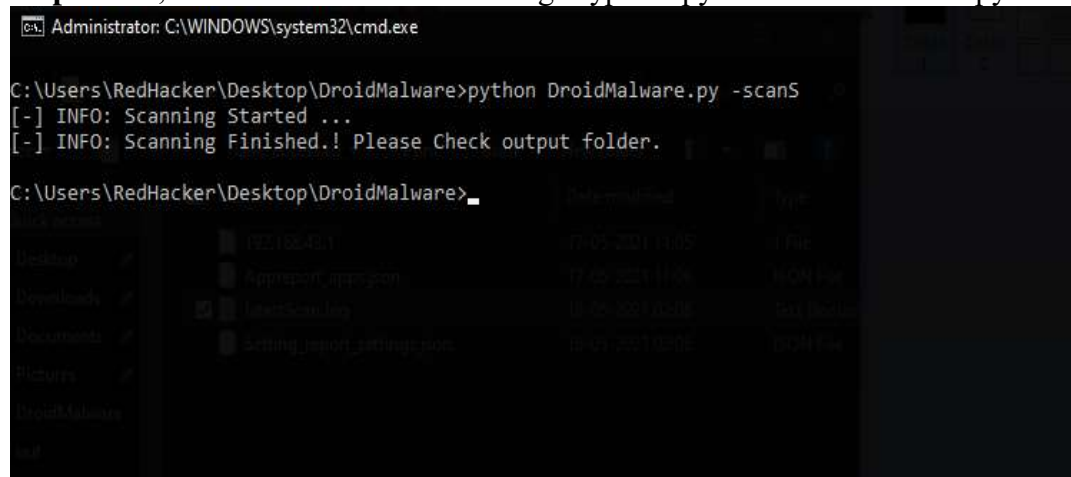


Figure No.9: Help commands of RMD Tool as output

Scan For Secure Settings

Step1:Now, Let's Scan the Device Setting. Type – “python DroidMalware.py -scanS” and hit Enter.



```
Administrator: C:\WINDOWS\system32\cmd.exe

C:\Users\RedHacker\Desktop\DroidMalware>python DroidMalware.py -scanS
[-] INFO: Scanning Started ...
[-] INFO: Scanning Finished.! Please Check output folder.

C:\Users\RedHacker\Desktop\DroidMalware>
```

Figure No.10:Settings Scan commands to scan

Step2: Go to Folder named “out” in RMD tool root directory, and checkout the results.

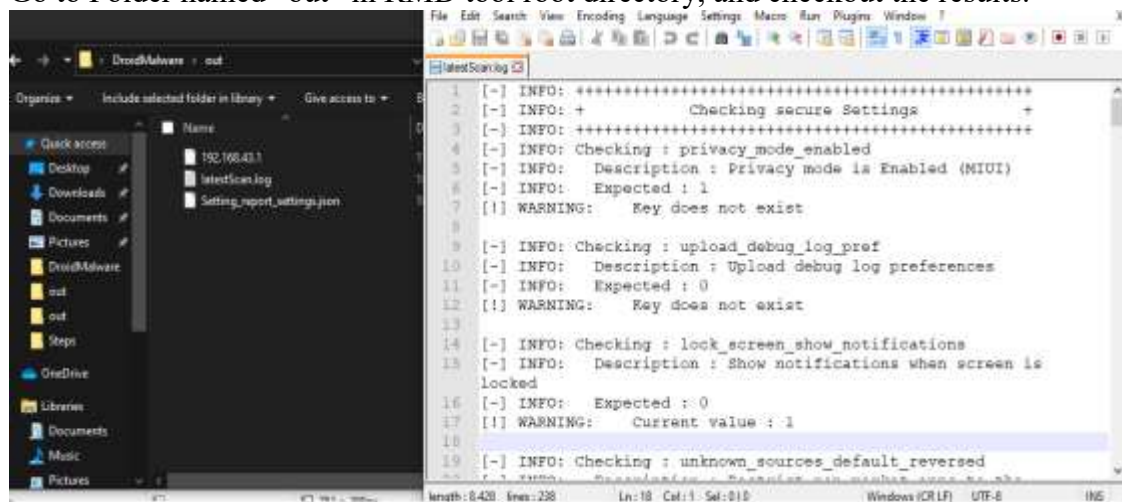


Figure No.11: Output of setting scan, as log file

It is checking secure setting.
Warning: Ke does not exist
Expected :1 Privacy mode is off.

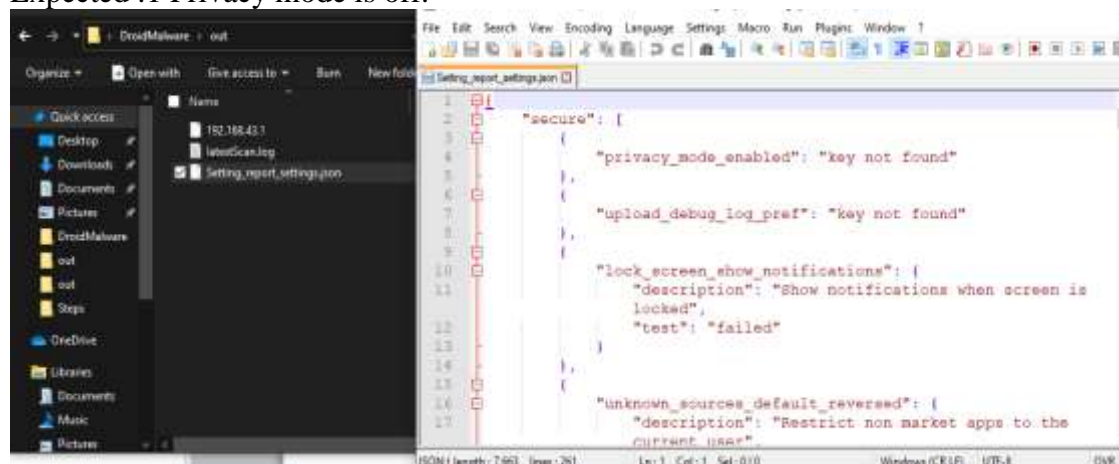
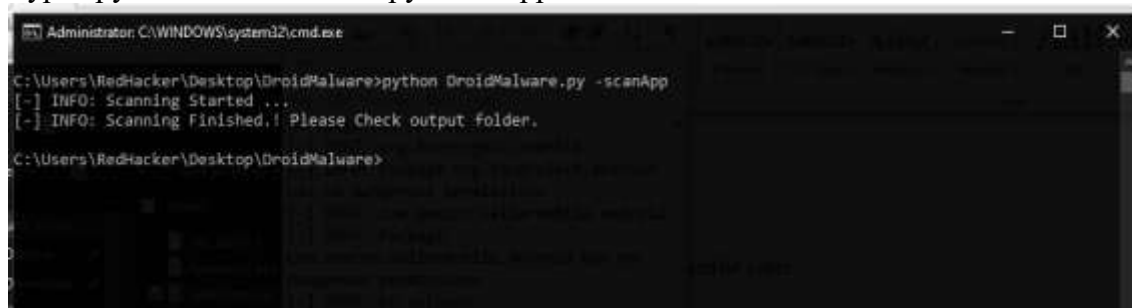


Figure No.12: Output of setting scan, as json file

A json file can also be used to export for a webapplication.

Now Scan For Application Malware Analysis

Step-1 Type “python DroidMalware.py -scanApp” and hit Enter



```
Administrator C:\WINDOWS\system32\cmd.exe

C:\Users\RedHacker\Desktop\DroidMalware>python DroidMalware.py -scanApp
[-] INFO: Scanning Started ...
[-] INFO: Scanning Finished. Please Check output folder.

C:\Users\RedHacker\Desktop\DroidMalware>
```

Figure No.13: Dynamic Application Scan Commands

Step- 2 Go to Folder named “out” in RMD Tool root directory, and checkout the results.

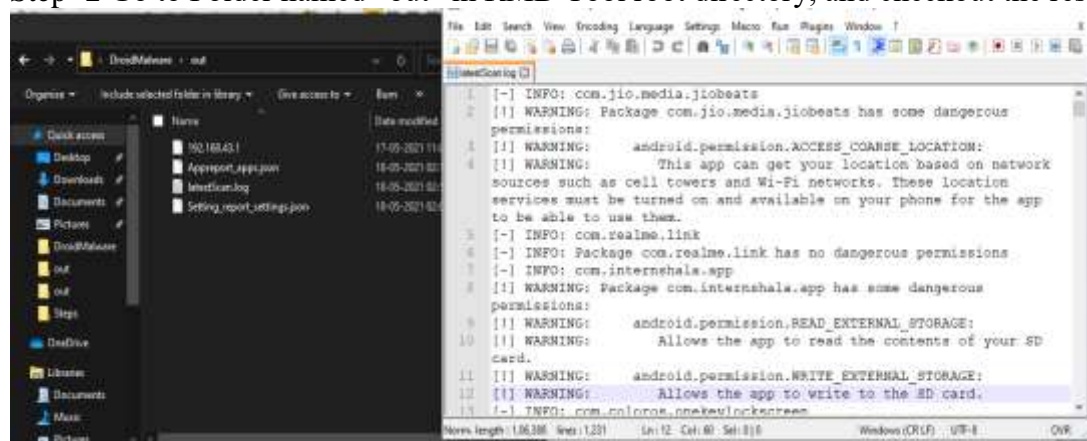


Figure No.14: Output of application scan, as log file

After scanning using RMD Tool, there were around 200 applications scanned and Twitter Moded application malware was ijected, it identified as malware,

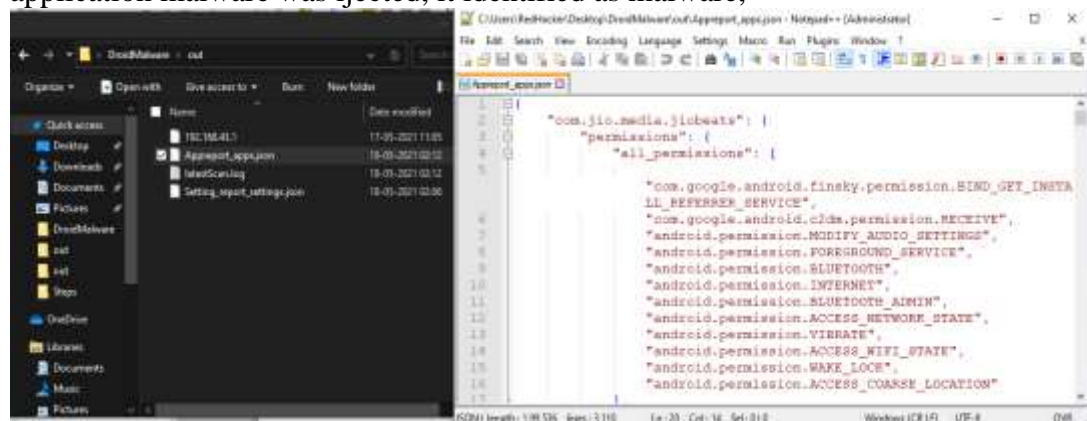


Figure No.15: Output of application scan, as json file

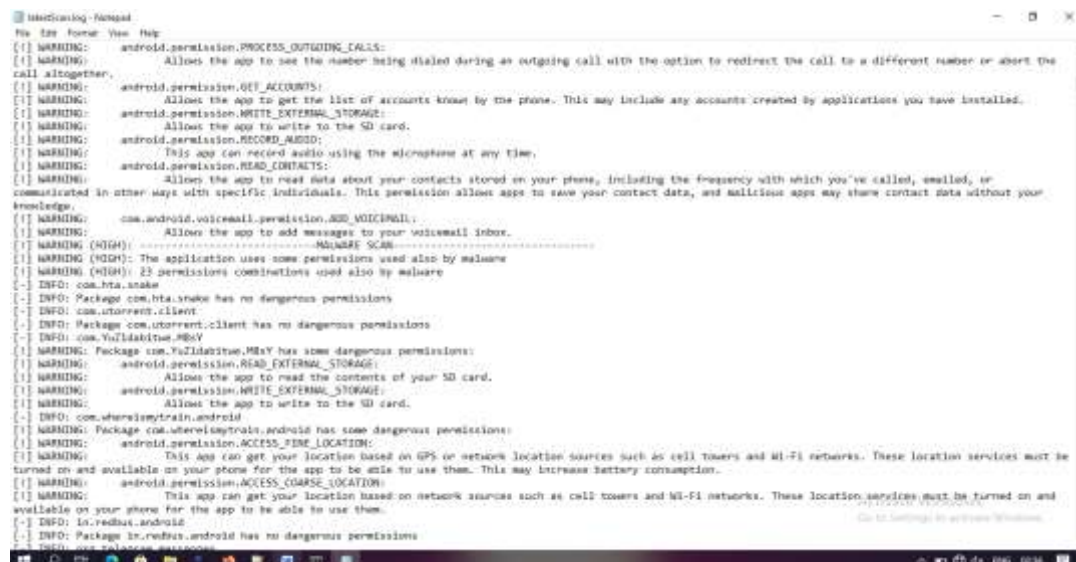


Figure No.16: Output of full application scan, as log file
Scan and list Apps, disable, and uninstall them

Type “python DroidMalware.py -l” and hit enter.



Figure No.17: Static Application Analysis Usage and commands

Here static analysis is being performed where in we list all the packages installed on the device and we can choose the package to be scanned.

Review of Literature

Conclusion

In this paper we have developed a Remote Malware detection (RMD) tool using ADB for Android mobile devices which aims at identifying the hidden malwares in an Android device. The experimental results demonstrate the effectiveness of this tool which shows that RMD can detect more hidden malwares which even an Malware detection tool cannot. Moreover, this RMD tool is efficient enough for practical implementation of local scanning of a device with in the same network and remote scanning of an Android device. With the increasing usage of smart technology in health care, which is easily prone to malwares, we aim to identify the malwares and security threats using machine learning algorithms and try to optimize these machine learning algorithms in our future work.

References

1. A.Razgllah,R.Khoury,S.Halle,KhanMohammadi, (2021) A survey of malware detection in Android apps: Recommendations and perspectives for future research.
2. O.Askan,R.Samet,O.Tanrioever, 2020Using a Subtractive Center Behavioral Model to Detect Malware, <https://doi.org/10.1155/2020/7501894>

3. J.H. Saltzer, M.D. Schroeder, The protection of information in computer systems, Commun. ACM 17 (1974).
4. S. Perez, Tap snake game in Android market is actually spy app (update), 2010, readwrite.com/2010/08/17/tap_snake_game_in_android_market_is_actually_spy_app .
5. E. Masabo, K. S. Kaawaase, J. Sansa-Otim, J. Ngubiri, and D. Hanyurwimfura, 2018 “A state of the art survey on polymorphic malware analysis and detection techniques,” ICTACT Journal of Soft Computing, vol. 8, no. 4.
6. S. Morgan, “Cybersecurity almanac 2019: 100 facts, figures, predictions and statistics. Cisco and cybersecurity ventures,”, <https://cybersecurityventures.com/cybersecurity-almanac-2019>.
7. A. Naway, Y. Li, 2018 A review on the use of deep learning in android malware detection, arXiv preprint arXiv:1812.10360
8. N. Alzahrani, D. Alghazzawi, 2019 A review on Android ransomware detection using deep learning techniques, in: Proceedings of the 11th International Conference on Management of Digital EcoSystems, pp. 330–335.
9. R. Sweetlin, A.S. Radhamani, 2016 Survey on detection of malware in Android, Int. J. Latest Trends Eng. Technol. 11 44–47, <http://dx.doi.org/10.21172/1.111.08>
10. P. Yan, Z. Yan, 2018 A survey on dynamic mobile malware detection, Softw. Qual. J. 26 (3) 891–919.
11. S. Arshad, M.A. Shah, A. Khan, M. Ahmed, 2016 Android malware detection & protection: A survey, Int. J. Adv. Comput. Sci. Appl. 7 (2) 463–475.
12. J. Chen, M.H. Alalfi, T.R. Dean, Y. Zou, 2015 Detecting Android malware using clone detection, J. Comput. Sci. Technol. 30 (5) 942–956, <http://dx.doi.org/10.1007/s11390-015-1573-7>
13. R. Potharaju, A. Newell, C. Nita-Rotaru, X. Zhang, 2012 Plagiarizing smartphone applications: Attack strategies and defense techniques, in: Engineering Secure Software and Systems - 4th International Symposium, ESSoS 2012, Eindhoven, the Netherlands, February, 16-17, Proceedings, 2012, pp. 106–120.
14. P. Liu, W. Wang, X. Luo, H. Wang, C. Liu, 2012, NSDroid: Efficient multiclassification of android malware using neighborhood signature in local function call graphs, Int. J. Inf. Secur. <http://dx.doi.org/10.1007/s10207-020-00489-5>
15. W. Zhou, Y. Zhou, X. Jiang, P. Ning, 2012, Detecting repackaged smartphone applications in third-party android marketplaces, in: Second ACM Conference on Data and Application Security and Privacy, CODASPY 2012, San Antonio, TX, USA, February 7-9, pp. 317–326, <http://dx.doi.org/10.1145/2133601.2133640>
16. G. Suarez-Tangil, S.K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, L. Cavallaro, 2017 Droidsieve: Fast and accurate classification of obfuscated Android malware, in: Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy, CODASPY 2017, Scottsdale, AZ, USA, March 22-24, pp. 309–320, <http://dx.doi.org/10.1145/3029806.3029825>.
17. M. Qiao, A.H. Sung, Q. Liu, 2016, Merging permission and api features for Android malware detection, in: 5th IIAI International Congress on Advanced Applied Informatics, IIAI-AAI 2016, Kumamoto, Japan, July 10-14, pp. 566–571, <http://dx.doi.org/10.1109/IIAI-AAI.2016.237>
18. D. Wu, C. Mao, T. Wei, H. Lee, K. Wu, 2012 DroidMat: Android malware detection through manifest and API calls tracing, in: Seventh Asia Joint Conference on Information Security, AsiaJCIS 2012, Kaohsiung, Taiwan, August 9-10, pp. 62–69, <http://dx.doi.org/10.1109/AsiaJCIS.2012.18>
19. K. Khanmohammadi, R. Khoury, A. Hamou-Lhadj, 2019 On the use of API calls to detect repackaged malware apps: Challenges and ideas, in: The 30th International Symposium on Software Reliability Engineering, ISSRE.
20. B.P. Sarma, N. Li, C. Gates, R. Potharaju, C. Nita-Rotaru, I. Molloy, 2012, Android permissions: A perspective combining risks and benefits, in: Proceedings of the 17th ACM Symposium on Access Control Models and Technologies, SACMAT '12, ACM, New York, NY, USA, pp. 13–22, <http://dx.doi.org/10.1145/2295136.2295141>.
21. H. Peng, C.S. Gates, B.P. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, I. Molloy, 2012, Using probabilistic generative models for ranking risks of Android apps, in: The ACM Conference on Computer and Communications Security, CCS'12, Raleigh, NC, USA, October 16-18, 2012, pp. 241–252, <http://dx.doi.org/10.1145/2382196.2382224>.