International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 11 | Nov -2020 ISSN: 2582-3930

Augmented Security Testing Platform for Android Mobile Applications

Shiwani Samel[1], Rushabh Shah[2]

[1]BTech. Mechatronics, SRM Institute of Science and Technology
[2]BTech. Information Technology, Vellore Institute of Technology

---

Abstract - This project proposes an automated virtual security testing platform for Android mobile apps. The method consist of three main components namely; mobile app trace information by customizing Android OS, using this customized OS to create a virtual testing platform and finally mobile malware detection by static and dynamic analyzing techniques. The proposed method is a malware detection solution based on comparing various malwares from server database which has a good compensation to the client-side mobile security software with both static and dynamic analysis.

Key Words: android, cybersecurity, malware, mobile app, mobile security, security testing, software, solution.

1. INTRODUCTION

These days protection of information security and personal data has become complex due to the malicious threats and application attacks[1][2]. Thus these various[3][4] attacks and threats has resulted in finding and providing various ways to defend against them, however improvisation is still required for the current detection technologies as they are ineffective to the new techniques of malware designers and thus can escape from anti-malwares[5][6][7]. So, it's important to provide a synergy of data mining processes to gain and improve malware detection methods and enable these systems to identify and detect malware with high precision, in less amount of time because in this case network security experts should react time efficiently to avoid mobile data theft[2][4]. It's essential to detect the unknown virus, trojans and this can be done by data mining, by focusing on malicious code analysis[8]. To achieve this, many classifiers have been implemented and show high precision rates[5][9]. Generation a feature set is one of the most common methods of data mining techniques for malware detection[8]. Signature-based detection is the most often used approach to detect malware[8]. Unfortunately, the results of the existing mobile security software are mediocre. Using a data set including 1260 malware samples, a research team conducted tests using four representative mobile security software: AVG, Lookout, Norton, and Trend Micro[10]. The margin of the experiments observes best case detects 80% of malware samples and the worst case detects about 20%. This approach can detect only the known malware but fails to stop new or unknown variants viruses. In this paper, we propose an automated virtual security testing platform for Android mobile apps. The proposed approach is a server-side malware detection solution which has a good compensation to the client-side mobile security software with both static and dynamic analysis.

2. LITERATURE SURVEY

In today's time the need of smart phones is increasing rapidly. Along with this, there is a trend of having certain android apps that are essential to socialize, however that isn't enough. New apps are being launched almost every day. As a result of popularity and functionality, mobile devices are a growing target for malicious activities. It is the need of the hour to protect our personal information.[11][12] In such context, mobile malwares have gained significant ground since the emergence and growth of smartphones and handheld devices, thus becoming a real threat. This also has the capacity of destroying a person's life by the theft of personal information. The various techniques such as installing lawful apps with hidden malware, updating recent apps that carry malicious variants or even a simple download can cause mobile malware malicious infections.[11] The effects of the infections can be single or multiple like accessing privileged data, remote control, financial charge and information collection. It is essential to be aware of the different malwares [11][12] and the basic framework on an android app.[13][14] Hence an effective model for malware detection is of great importance. There are different ways for the detection of these trojans, cloud-based malware detection is a promising approach towards mobile security[11], text classification method TMSVM[12], automated vulnerability tests of Android smart phones[13][14], supervised pattern recognition techniques.[15] The aspect that can be controlled by humans is granting permissions to the apps.[20] Sometimes unknowingly we give few applications permissions which are not really required and thus endangering our information, it is very important to understand the requirement of the permission.[12][16][17]

3. DESIGN OF ANDROID MOBILE APP

A. System Architecture

Existing system: Mobile devices such as smartphones and tablets have been widely adopted for personal and business purposes. However, the popularity of the mobile devices also raises many security issues and challenges. A mobile device is an easy target for cyber criminals due to its central data management. There have been many reports of threats and attacks on mobile device data. Among all the threats and attacks, malware (virus, Trojan, and spyware) is one of the greatest threats to mobile security. According to Trend Labs, the big data consists 718, 000 malicious Android apps in the second quarter of 2013.[11] Signature-based detection is the most often used approach to detect malware. However, the testing results from the existing mobile security software are not encouraging. The various disadvantages for this accuracy is low when compared to new algorithms, also requires some computational devices and implementation cost is high. However the proposed system is different. Proposed system:

An automated virtual security testing platform for Android mobile apps. The proposed approach is a server-side malware detection solution which has a good compensation to the client-side mobile security software with both static and dynamic analysis. The method consists of three main components namely; mobile app trace information by customizing Android OS, using this customized OS to create a virtual testing platform and finally mobile malware detection by static and dynamic analyzing techniques. We use of data mining methods for malware (malicious applications) detection and proposed a framework as an alternative to the traditional signature detection methods. As discussed before, the unknown malwares and changed codes cannot be detected by the traditional approaches which is using signatures. Thus for this process a well organized data mining framework is required. The classification of a particular program can be done into a malware or a clean class by the helps of collecting, analyzing and processing several corrupted and clean programs. Thus for the collection of various samples vector space model is used. The segregation of framework consists of includes two separate and distinct classes of experiments of data mining. This structure has advantages such as high accuracy, high computational processing and independent of ethnicity. Also there is no chance of missing any malicious code.

A1. System Framework

The relations between different components of the system are known. It becomes easier to deduce the logic when the connections are obtained.

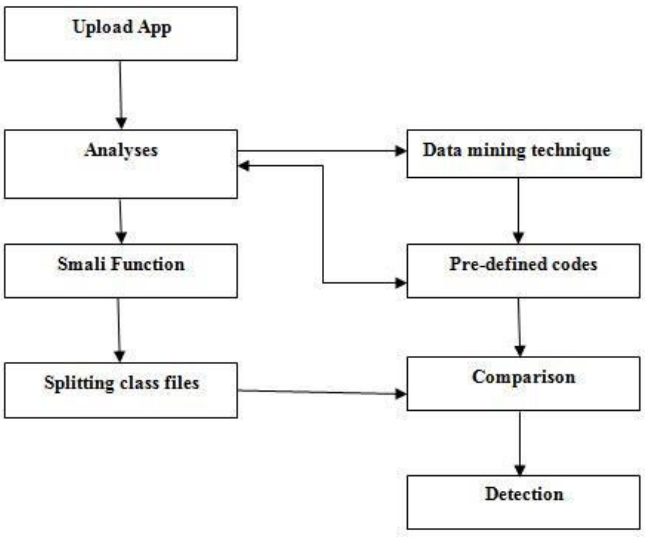

Figure1. System Framework

A2. Architecture diagram

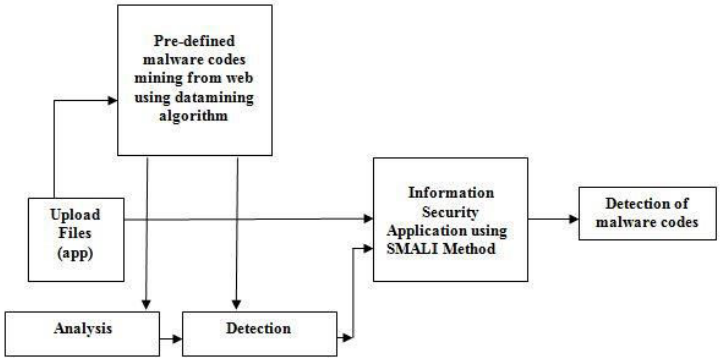

Figure2. Architecture Diagram

B. UML Diagrams

A UML diagram is a diagram supporting the UML (Unified Modelling Language) with the aim of visually representing a system along side its main actors, roles, actions, artifacts or classes, so as to raised understand, alter, maintain, or document information about the system.

B1. Activity diagram

The pictorial representation of the software is depicted so as to simply the flow of process for the end user.

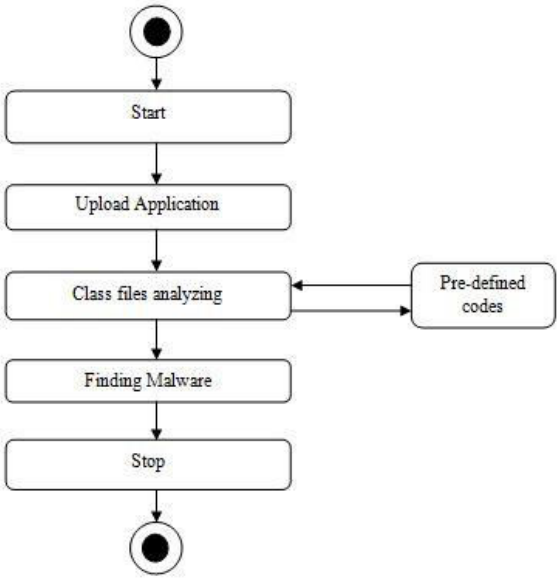

Figure3. Activity Diagram

B2. Use case diagram

The simplest representation of user interaction between different use cases is shown in this diagram.

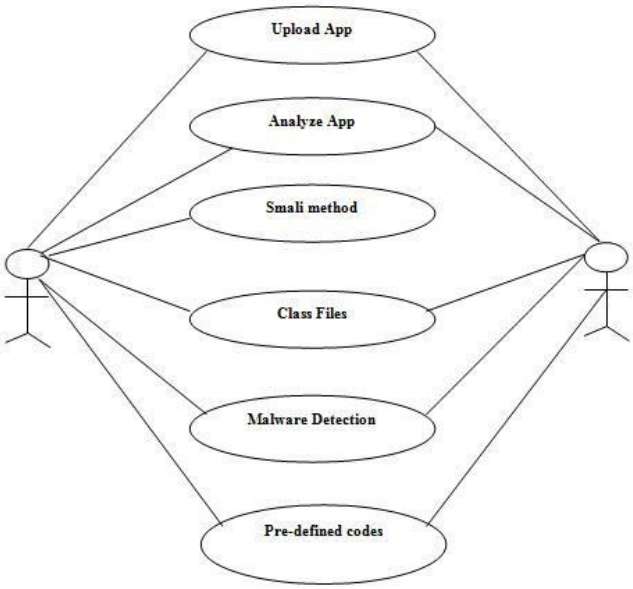

Figure4. Use Case Diagram

C. Module Description

C1. Language specification : Python

Python has bene a widely used general-purpose, high level programming language . it had been initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation. it had been mainly developed for emphasis on code readability, and its syntax allows programmers to precise concepts in fewer lines of code. Python is one of the programming languages that allows you to work quickly and integrate systems more efficiently.

Python graphical user interfaces (GUIs)

i) Tkinter − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python.

ii) wxPython − This is often an open-source Python interface for wxWindows.[21]

iii) JPython − JPython is considered a Python port for the Java language which gives Python scripts continuous smooth access to Java class libraries on the local machine.[22]

There are many other interfaces available, which you'll find on the internet.

C2. Module Description

i) Upload an APK

Transferring data from one remote system to a different under the control of an area system is remote uploading. Remote uploading is employed by some online file hosting services. it's also used when the local computer features a slow connection to the remote systems, but they need a quick connection between them. The basic steps are to first download the data to the local host without remote uploading functionality and then uploaded to the remote file hosting server. These are carried out both times over connections with comparison files.

ii) Data mining analysis

Android application package (APK) may be a format created by Google, which is employed to distribute and install an application onto the Android OS .One of the most famous free virus-sharing website with currently 27 million virus samples, also as a repository of malware samples to supply security researchers, incident responders, forensic analysts, and therefore the morbidly curious access to samples of malicious code. The platform provided a large collection dataset of 24317 Android malicious APK files. These APK files have already been renamed to their hash value, which made it impossible to retrieve the first name or category. This data set was generated.

iii) Code scanning and matching

Scanning is that the most generally used industry standard. Scanning involve checking out strings in files. The strings being searched are pre-defined virus signatures and support exact matching also as wildcards to seem for variants of an epidemic. Data processing involves the appliance of a full suite of statistical and machine learning algorithms on a group of features derived from malicious and clean programs.

iv) Detecting malware in android file

This project proposes employing a KNN algorithm on the Java code which has been retrieved by Smali. We have already got a dynamic and lavish data set provided by the open source. We logically analyzed the possible reasons behind the patterns of how permissions are requested along side other features. Finally it detects the malware code supported permission and data processing technique.

D. Algorithm Description

i) smali/baksmali

It is an assembler/disassembler for the dex format used by dalvik, Android's Java VM implementation. The syntax is loosely supported on Jasmin's/dedexer's syntax and supports the complete functionality of the dex format. The Icelandic equivalents of "assembler" and "disassembler" are named "smali" and "baksmali" respectively.

ii) APKiD

It gives you information about how an APK was made. Different aspects such as various compilers, packers, obfuscators are identified. It's PEiD for Android.

iii) KNN algorithm for classification

The k-nearest neighbors (KNN) algorithm, to say is a simple, easy-to-implement supervised machine learning algorithm which not only solves one but used to solve both classification and regression problems. Similar things existing in close proximity is one of the assumptions of KNN algorithm. In other words, similar things are almost one another

.The advantages of such an algorithm is easy to interpret output, low calculation time and high predictive power.

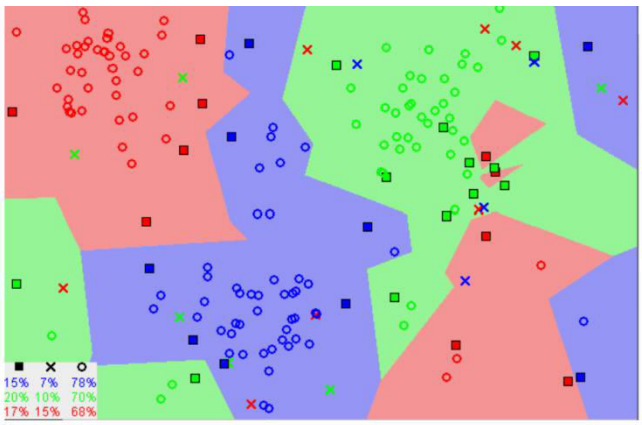

Figure5. KNN Model Diagram

D1.Step by step implementation of KNN model

i) Load the data
ii) Initialise the value of k
iii) For getting the predicted class, iterate from 1 to total number of training data points :
1. Calculate the length space between test data and every row of training data. Here we'll use Euclidean distance as our distance metric since it's the foremost popular method. The other metrics which can be used commonly are Chebyshev, cosine.
2. Sort the calculated distances in ascending order with respect to distance values
3. Achieve top k rows from the sorted array
4. Get foremost frequent class of these rows
5. Return the predicted class

E. Implementation

i) Identifying Database of around 27,000 Virus Files onVirusShare.com and using them dynamically for our Knn Algorithm
ii) Using the Smali code in Python to extract the Java Code from the Android Apk file.
iii)Started working on the user Interface with the help of Tkinter in Python.
iv) Using the Django Framework for uploading the apk files by the User.

Tkinter Widgets : These provides various controls, such as buttons, labels and text boxes used in a GUI application. These controls are commonly called widgets. Currently in Tkinter 15 types of widgets are present. We present these widgets as well as a brief description as following:

1. Button:

The Button widget is known to display buttons in your application.

2. Canvas:

The Canvas widget is employed to draw shapes, like lines, ovals, polygons and rectangles, in your application.

3. Check button

The Check button widget is well known to display variety of options as checkboxes. More than one option can be selected at the same time.

4. Entry

The Entry widget displays a single-line text field for accepting values from a user.

5. Frame

The Frame widget is known as a container widget to arrange other widgets.

6. Label

The Label widget is employed to get a single-line caption for other widgets. It can also contain images.

7. Listbox

The Listbox widget is well knonwn to provide various options to a user.

8. Menubutton

The Menubutton widget displays menus in your application.

9. Menu

The Menu widget is employed to carter the need of various commands to a user. These commands are contained inside Menubutton.

10. Message

The Message widget displays multiline text fields for accepting values from a user.

11. Radiobutton

The Radiobutton widget is known to display a large variety of options as radio buttons. The user can select just one option at a time.

12. Scale

The Scale widget is employed for the requirement of a slider widget.

13. Scrollbar

The Scrollbar widget is employed to add scrolling capability to various widgets, such as list boxes.

14. Text

The Text widget is displays text in multiple lines.

15. Toplevel

The Toplevel widget shows a separate window container.

16. Spinbox

The Spinbox widget helps to select from a fixed number of values. It is similar to and a variant of the standard Tkinter Entry widget.

17. PanedWindow

A PanedWindow is basically a container widget which contains any number of panes, arranged horizontally or vertically.

18. LabelFrame

A labelframe is a simple container widget. The basic function is to be a spacer or container for not simple window layouts.

19. tkMessageBox

This module is employed to display message boxes in your applications.

4. RESULT AND CONCLUSION

The testing results from the existing mobile security software are not encouraging. Using a data set including 1260 malware samples, a research team conducted tests using four representative mobile security software: AVG, Lookout, Norton, and Trend Micro. The experiments show that the best case detects 79.6% of malware samples and the worst case detects only 20.2%. The issue with signature-based detection is that apps could change through updated code or modified just enough to throw off the signature for the mobile security software to detect. The approach catches known malware, but fails to stop new or unknown variants in the wild.

Module1: Front End UI of our website with the help of Django

```
Mobile Security Framework v1.0.1 Beta

REST API Key: acf66070eda9db5905b24a423f1f58e1c8be05f149bfeb704c5efa8302ab56f6
OS: Windows
Platform: Windows-10-10.0.18362-SP0

[INFO] Finding JDK Location in Windows....

[INFO] Oracle Java JDK is installed!

[WARNING] Could not find VirtualBox path.
[INFO] MobSF Basic Environment Check
[INFO] Checking for Update.

[WARN] A new version of MobSF is available,
Please update from master branch or check for new releases.

System check identified no issues (0 silenced).
February 28, 2020 - 12:10:44
Django version 3.0.3, using settings 'MobSF.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Figure6. Django server started

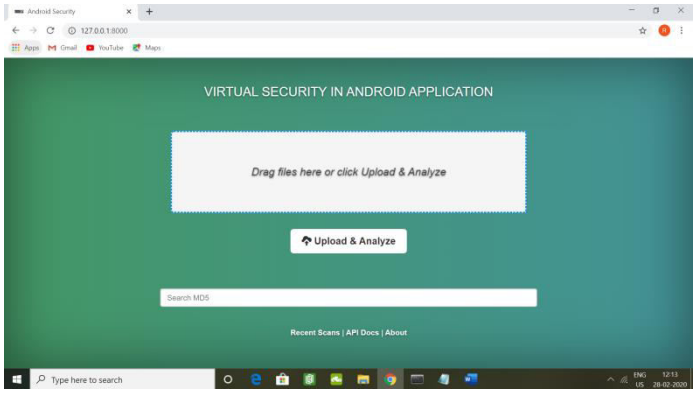

Figure7. Home Page of the Software for the user to upload his APK

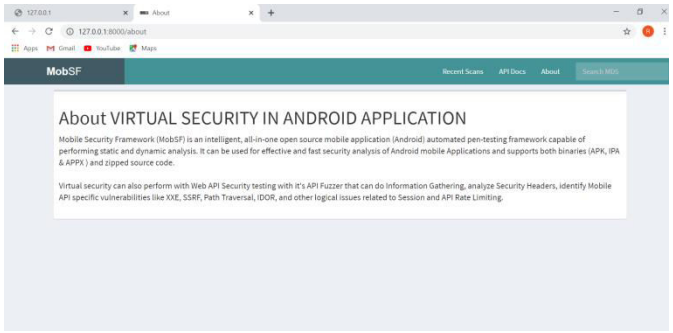

Figure8. A page about the software and how to use it

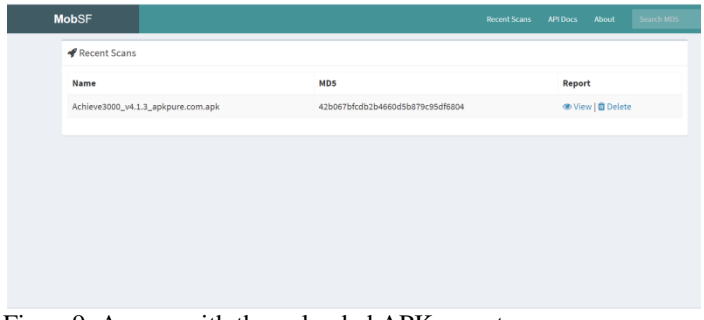

Figure9. A page with the uploaded APK report

The user can also view his recent uploads and the problems occurring with each APK he uploads.

Module 2: Uploading the User APK
We are using the Post method here for this purpose

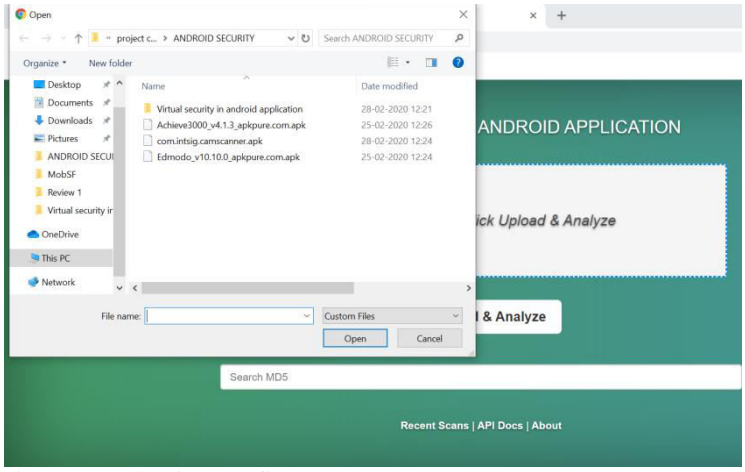

Figure10. Selecting the file to upload

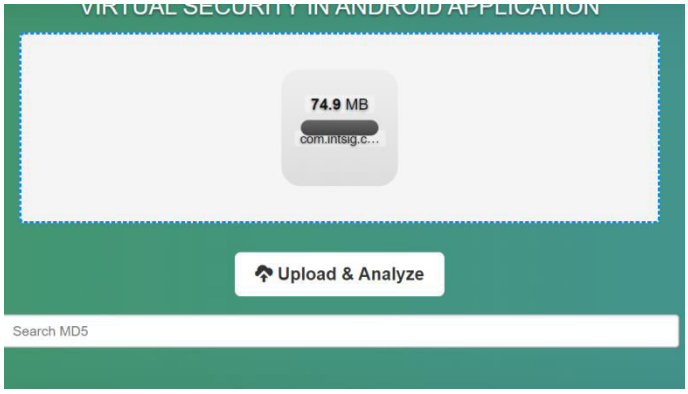

Figure11. Percentage File Uploaded

Module 3: Deassembling the APK into Smali Code and other sub codes

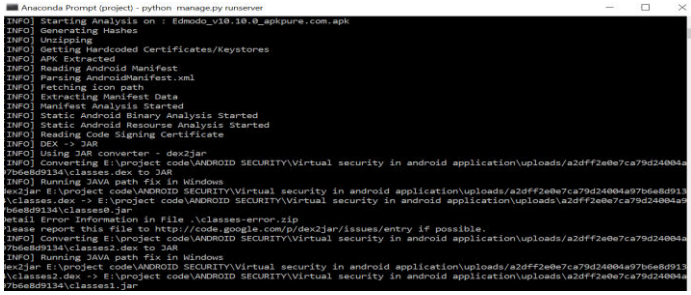

Figure12. Percentage File Uploaded

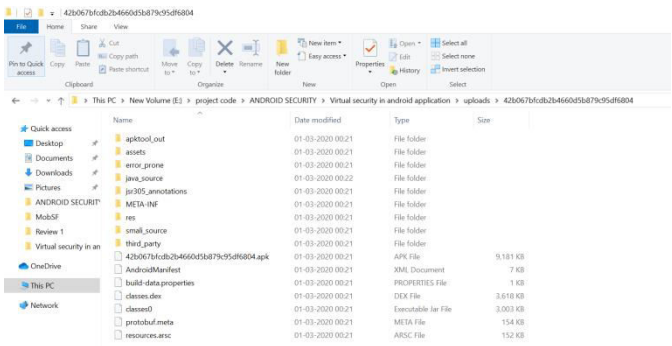

Figure13. Deassembled Code being stored in a separate folder for every new APK

Module 4 : Data Mining

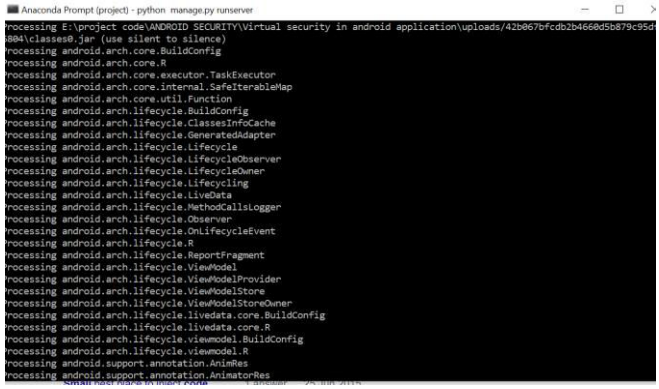

Figure14. Smali Code is being mined after applying appropriate filters

Module 5 : Comparison and Analysis of the Code to detect Trojans and Malwares

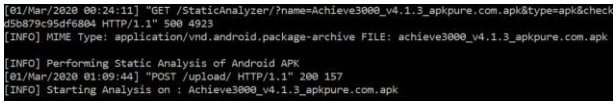

Figure15. Detection of malwares

Figure16. Issues and Malwares present along with their risks

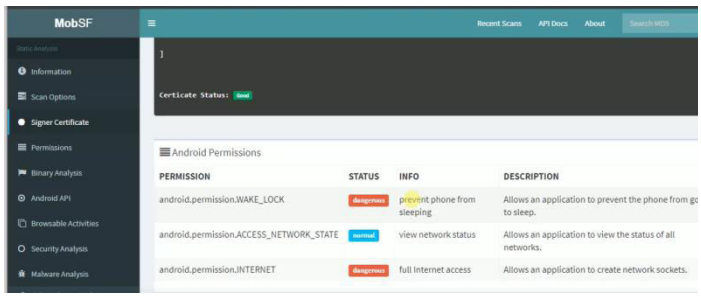

Figure17. Permissions accessed by the APK without the users knowledge

Android is the largest installed base of mobile platform for smartphones and tablets. It also has the largest malicious and high risk applications. In this paper, we propose an automated virtual testing platform for Android mobile apps. The proposed approach can be used to test mobile apps and requires nothing to be changed to a mobile app. Unlike a client-based mobile detection approach which must consider the constraints of a mobile device, the proposed approach is a server-side solution which can utilize more powerful and effective techniques to analyze data and detect malware. The proposed approach is also scalable. It is easy to clone the virtual testing platform in a cloud, launch multiple instances of the testing platform, and use them to test multiple mobile apps simultaneously. We have demonstrated that many mobile apps can be launched in the virtual testing platform using our Genymotion setup. However, restrictions may also exist due to using the virtual device and the ARM emulator. Emulators usually do not provide 100% of device functionality and it may block some apps to be launched in the virtual platform. In addition, a malicious app may also detect that it runs on a virtual device and reacts to this, e.g., by not acting malicious at all. Our future work includes developing and implementation of the proposed testing platform and testing mobile apps using the testing platform.

ACKNOWLEDGEMENT

It is our pleasure to express with deep sense of gratitude to Dr. Priya V, Professor, School of Information Technology and Engineering, Vellore Institute of Technology, for her constant guidance, continual encouragement, understanding; more than all, she taught us patience in this endeavour. The authors, Shiwani Samel and Rushabh Shah individually contributed for the completion of the project. No external funding was used. VirusShare.com has been used to verify the data collected and to check for the malware and virus files. All the other information has been researched from the world wide web.

REFERENCES

[1] Jang-Jaccard, Julian; Nepal, Surya. A survey of emerging threats in cybersecurity. Journal of Computer and System Sciences. 2014; 80(5):973–993. https://doi.org/10.1016/j.jcss.2014.02.005

[2] Yesilyurt, Murat & Yalman, Yıldıray. (2016). Security Threats on Mobile Devices and their Effects: Estimations for the Future. International Journal of Security and Its Applications. 10. 13-26. 10.14257/ijsia.2016.10.2.02.

[3] Tekade, Puja S., Chetan J. Shelke and P. R. Patil Coet. “A Survey on different Attacks on Mobile Devices and its Security.” (2014).

[4] Vasundhara Varade, “Research on Different Types of Attacks in Smart Phones”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5, Issue 6, June 2016

[5] https://cs.stanford.edu/people/eroberts/cs201/projects/2000-01/viruses/anti-virus.html

[6] Suarez-Tangil, Guillermo & Tapiador, Juan & Peris-Lopez, Pedro & Ribagorda, Arturo. (2013). Evolution, Detection and Analysis of Malware for Smart Devices. IEEE Communications Surveys &amp Tutorials. 16. 10.1109/SURV.2013.101613.00077.

[7] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic Malware Analysis in the Modern Era—A State of

the Art Survey. ACM Comput. Surv. 52, 5, Article 88 (October 2019), 48 pages. DOI:https://doi.org/10.1145/3329786

[8] Abhay pratap singh , Dr.S.S handa, "Malware detection using data mining techniques", International Journal of Advanced Research in Computer and Communication Engineering Vol. 4, Issue 5, May 2015

[9] Mathur, K., and Saroj H. A Survey on Techniques in Detection and Analyzing Malware Executables. International Journal of Advanced Research in Computer Science and Software Engineering, Vol. 44, No. 2, 2012.

[10] Zhou, Yajin & Jiang, Xuxian. (2012). Dissecting Android Malware: Characterization and Evolution. Proceedings - IEEE Symposium on Security and Privacy. 4. 95-109. 10.1109/SP.2012.16.

[11] Penning, Nicholas & Hoffman, Michael & Nikolai, Jason & Wang, Yong. (2014). Mobile malware security challeges and cloud-based detection. 2014 International Conference on Collaboration Technologies and Systems, CTS 2014. 181-188. 10.1109/CTS.2014.6867562.

[12] Xiao, Xi & Xiao, Xianni & Jiang, Yong & Li, Qing. (2015). Detecting mobile malware with TMSVM. 152. 507-516. 10.1007/978-3-319-23829-6_35.

[13] H. Lockheimer, "Android and Security," February 2012. http://googlemobile.blogspot.com/2012/02/androidandsecurity. html.

[14] S.Bagal, Navnath. (2013). Android open-source operating System for mobile devices.. IOSR Journal of Computer Engineering. 11. 25-29. 10.9790/0661-1152529.

[15] K. Costa, L. Silva, G. Martins, G. Rosa, R. Pires, J. Papa, "On the Evaluation of Restricted Boltzmann Machines for Malware Identification", International Journal of Information Security Science, vol. 5, no. 3, pp. 71-81, 2016.

[16] Sarma, B.P., Li, N., Gates, C., Potharaju, R., Nita-Rotaru, C., Molloy, I.: Android permissions: a perspective combining risks and benefits. In: Proceedings of the 17th ACM symposium on Access Control Models and Technologies, pp. 13–22, ACM Press, June 2012

[17] Moonsamy, V., Rong, J., Liu, S., Li, G., Batten, L.: Contrasting permission patterns between clean and malicious android applications. In: Zia, T., Zomaya, A., Varadharajan, V., Mao, M. (eds.) SecureComm 2013. LNICST, vol. 127, pp. 69–85. Springer, Heidelberg (2013)

[18] Cesare, S., Xiang, Y.: Classification of malware using structured control flow. In: Proceedings of the Eighth Australasian Symposium on Parallel and Distributed Computing-Volume 107, pp. 61–70. Australian Computer Society, Inc., January 2010

[19] Lin, Y.D., Lai, Y.C., Chen, C.H., Tsai, H.C.: Identifying android malicious repackaged applications by thread-grained system call sequences. Comput. Secur. 39, 340–350 (2013)

[20] Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and API calls. In: IEEE 25th International Conference on Tools with Artificial Intelligence (ICTAI), 2013, pp. 300–305. IEEE Press, November 2013

[21] http://wxpython.org.

[22] http://www.jython.org.