Android Security Analysis using Mobile Sandbox

Munishka Vijayvergiya, Abhignya Tayi

***Abstract:** Today, smartphones and Android devices are effectively in development like never before and have become the easiest cybercrime forum. It is necessary for security experts to investigate the vengeful programming composed for these frameworks if we closely observe the danger to security and defence. The main objective of this paper was to describe Mobile Sandbox, which is said to be a platform intended to periodically examine Android applications in new ways. First of all in the essence of the after-effects of static analysis that is used to handle the dynamic investigation, it incorporates static and dynamic examination and attempts to justify the introduction of executed code. On the other hand, to log calls to native APIs, it uses those techniques, and in the end, it combines the end results with machine learning techniques to collect the samples analysed into dangerous ones. We reviewed the platform for more than 69, 000 applications from multi-talented Asian international businesses sectors and found that about 21% of them officially use the local calls in their code.*

***Keywords:** The Main Objective of This Paper Was To Describe Mobile Sandbox*

References:

1. Mobile sandbox : using static and dynamic analysis.http://campar.in.tum.de/pub/spreitzenbarth2015mobilesandbox/spreitzenbarth2015mobilesandbox.pdf
2. Android Developers.: Android platform versions http://developer.android.com/about/dashboards/index.html

I. INTRODUCTION

Smartphone sales have impressively hit the next stage in recent years. The attention of cybercriminals who try to manoeuvre the user into installing harmful software on the computer was drawn to this phenomenal growth. We tried to understand about 6,100 harmful applications in previous work and assembled them with the aid of the VirusTotal API into 51 malware institutions. In addition, 45 percent of our malware institutions sent text messages. Interestingly, these messages were sent to affect individuals such as industrialists, the numbers of politicians to make money immediately.

Until recently, the research was performed manually with the aid of manpower using instruments such as compilers and debuggers with their own intelligence, which takes a lot of time and, depending on the expert's ability, often results in errors. Tools were therefore built.

Manuscript received on 21 May 2021 | Revised Manuscript received on 03 June 2021 | Manuscript Accepted on 15 June 2021 | Manuscript published on 30 June 2021.

***** Correspondence Author
Munishka Vijayvergiya*, SCOPE, Vellore Institute of Technology, Chennai, India. Email: munishka.vijayvergiya2019@vitstudent.ac.in
Abhignya Tayi*, SCOPE, Vellore Institute of Technology, Chennai, India. Email: tayi.abhignya2019@vitstudent.ac.in

A static analytical approach, focused on signature detection of applications, serves as a popular antivirus technology approach.

The back or trap door uses the approach in use where the unintelligible methods are made more difficult for static study.

By using the read elf utility, they worked to take out the function calls from an Android application and made the difference in producing the list of familiar malware info. Malware writers created the behaviour of rendering ambiguous methods that have been shown to be successful against changeless research, parallel to the environment of desktop PCs.

References:

1. *Dimjasevic, M., e. a. (2015). Evaluation of android malware* detection based on system calls. https://dl.acm.org/doi/abs/10.1145/2875475.2875487
2. Guptil, B. (2013). Examining application components to reveal android malware. https://scholar.afit.edu/etd/868/

II. BACKGROUND

With the notable usage of smartphones and the distribution model of applications, criminals are making their work easy with the help of smartphones as a potential target for malware to steal private information, misuse it for premium SMS services, or try to manoeuvre necessary banking information (mTAN) on these devices. Sometimes, we can even find these threats within one hostile app. In this area of paper, we give a short overview of current mobile threats and describe why and how the Android platform is the most targeted mobile platform. Mobile threats are categorized into two classes: web-based and application-based threats.

These threats depend on the rigorous usage of mobile browsers and their numerous options of usage of implementations.

- malware is the software it is designed to cause damage to the mobile sandbox.it has the potential to wipe out all the data in the mobiles.

- personal spyware gathers personal information and it relays on data firms, advertisers etc.normally it tracks and sells your internal usage data, and captures the credit card and bank account information.

- Grayware operates similarly to malware, but it is not transmitted to harm users directly. It won't affect the functionality of the system . Above all, information about us patterns is collected for the purpose of selling any of this data or to make advertisements in a manner.

References:

1. Static and dynamic analysis of Mobile malware . https://www.researchgate.net/publication/314521542_Static_and_Dynamic_Analysis_of_Android_Malware
2. https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1488&context=etd_projects

III. OBJECTIVES

The main aim of our project is to explore the static and efficient ways of using software tools to detect malware on Android, compare and decide the best method. As we all know, cyber crime is rising day by day, so Android is increasingly focusing on it. For researchers to evaluate hateful applications written for programmes, knowing the safety and privacy risks is important. Mobile-Sandbox is a software designed to test Android apps in several ways automatically. It includes static and dynamic analysis, i.e. the static analysis outcomes used to direct dynamic analysis and increase the generated code coverage.

IV. SYSTEM ARCHITECTURE AND IMPLEMENTATION

To search for malicious software, both static and dynamic analysis have been deployed. In order to identify applications as malicious or benevolent, static analysis uses machine learning and these results can be used in dynamic analysis. The programme is performed in an emulator in dynamic analysis and every process is logged in. The system uses some methods to log calls to native APIs, which is an essential process since certain malicious applications can call certain functions outside their native codes, and it is crucial to track all these activities. To give final results, all these outcomes are then combined.

A. *Static Analysis:*

Static analysis initially checks the app downloaded and its source code. Many antivirus technologies use signature based-detection of apps and a similar approach is used by static analysis. Many malware systems have begun to use certain obfuscation techniques. These techniques make static analysis trickier and can often surpass static analysis without being caught as malicious. One such method used by many android applications is concealing certain dangerous activities. They do this by calling functions outside the Dalvik/Java runtime library. These risky functions may be written in C/C++. This is where Dynamic analysis comes handy.

Firstly, the hash value of the database is checked with a database called VirusTotal. This tells us how many tools have marked this application as malicious with regards to the total number of tools that inspected this application. Then, all the files of the application are unzipped using WinZip and all the permissions required by the particular application are checked. Here, reading the SDK version becomes important because only if the application is compatible with our android system, it is sent for dynamic analysis. Otherwise, its discarded. After this process, parsing of certain files is done. This is automated. There are certain functions which are known to be dangerous for android security like sendTextMessage(), getPackageInfo(), getSimCountryIso() and all these functions are checked for.

Another check is performed to evaluate any encryption techniques used in the application. Over and underprivileged applications are very important to check for here because when combined, they can greatly threaten the security of a device. Thus, this is also monitored in the static analysis. Once the entire process of static analysis completes, an XML file is generated which contains all the data that has been collected in the previous steps.

B. *Dynamic Analysis:*

Dynamic analysis looks at the run-time behavior of an application. This helps to figure out many threats/ dangerous activities which may have been missed during static analysis. Dynamic analysis executes the application being inspected in a controlled environment called sandbox. Every relevant operation of the execution of the application such as sending SMS messages, reading data from storage, and connecting to remote servers, is monitored and a report is generated. Though Dynamic analysis handles obfuscation techniques, it is circumvented by runtime detection methods. This is why, it's important to combine static and dynamic analysis.

For dynamic analysis, the Android emulator provided by Google is used. This emulator can also be set to its previous state once the application has been tested. This is an advantage. The latest version of Droidbox has been used as the basis for dynamic analyzer to make it compatible with the recent android versions. Droidbox however lacks the support needed to track native API calls which is why Android NDK has also been used. ltrace command has also been used to track the code. All this information including the native calls required to check shared objects is documented in separate files. Wireshark is a network simulation tool and it has also been used to monitor all the data sent over a network. Most applications require user interaction for them to be assessed properly and Ranorex Studio has been used for the same. It generates a number of interaction elements which are sufficient to track all possible activities of the application.

References:

1. Tamara, H., e. a. (2007). Design and evaluation of dynamic software birthmarks based on api calls. Nara Institute of Science and Technology, Technical Report. https://www.academia.edu/4149097/Design_and_Evaluation_of_Dynamic_Software_Birthmarks_Based_on_API_Calls
2. Zhou, Y., e. a. (2012a). Detecting malicious apps in official and alternative android markets. Proceedings of the Second ACM Conference on Data and Application Security and Privacy. https://www.csd.uoc.gr/~hy558/papers/mal_apps.pdf
3. Saudi, M., e. a. (2015). Android mobile malware surveillance exploitation via call logs: Proof of concept. 17th UKSIM-AMSS International Conference on Modelling and Simulation. https://uksim.info/uksim2015/data/8713a176.pdf
4. .Wang, X., e. a. (2009). Detecting software theft via system call based birthmarks. Proceedings of 25th Annual Computer Security Applications Conference. http://www.cse.psu.edu/~sxz16/papers/acsac09.pdf
5. Fuchs, P., e. a. (2009). Scandroid: Automated security certification of android applications. Technical Report CSTR-4991, Department of Computer Science, University of Maryland,College Park. https://www.scitepress.org/Papers/2017/62567/
6. Hand, J., e. a. (2001). A simple generalisation of the area under the roc curve for multiple class classification problems. http://www.defaultrisk.com/pa_test_05.htm
7. Aung, Z., e. a. (2013). Permission-based android malware detection. International Journal of Scientific Technology Research. https://www.researchgate.net/publication/288267309_Permission-Based_Android_Malware_Detection

V. USE CASE DIAGRAM

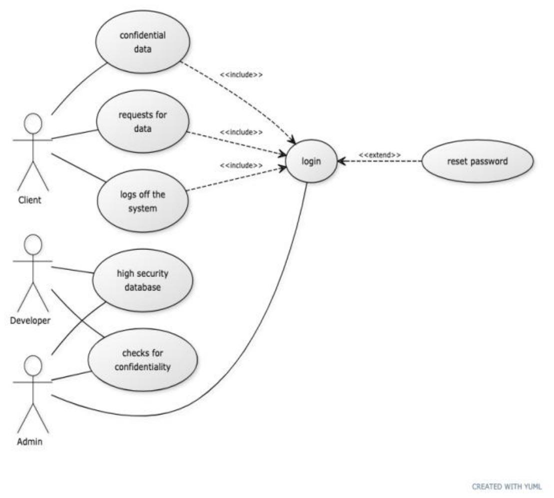


Figure 1

This figure shows the use case diagram of android security analysis using mobile sandbox

VI. DFD DIAGRAM

Level 0

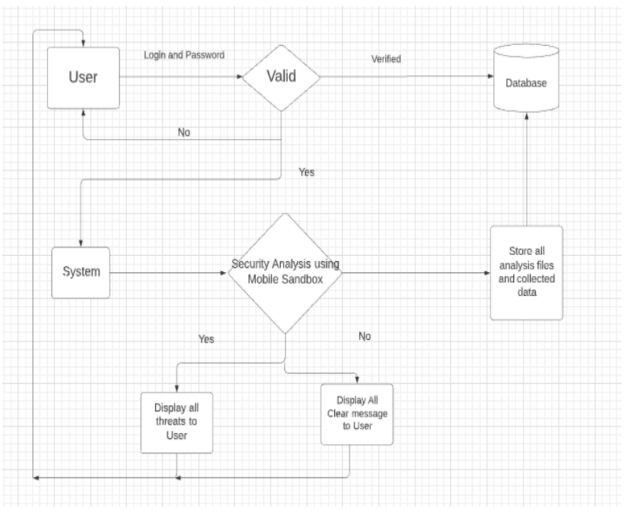

User
Login and Password
Valid
Verified
Database
No
Yes
System
Security Analysis using Mobile Sandbox
Store all analysis files and collected data
Yes
No
Display all threats to User
Display All Clear message to User

Figure 2

This figure shows the DFD level 0 diagram of android security analysis using mobile sandbox

Level 1

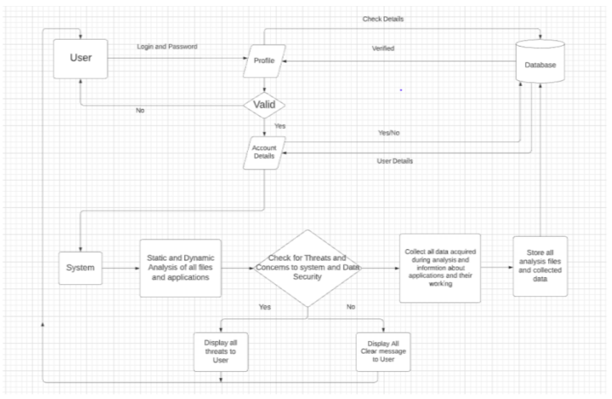

Check Details
User
Login and Password
Profile
Verified
Database
No
Valid
Yes
Yes/No
Account Details
User Details
System
Static and Dynamic Analysis of all files and applications
Check for Threats and Concerns to system and Data Security
Collect all data acquired during analysis and informtion about applications and their working
Store all analysis files and collected data
Yes
No
Display all threats to User
Display All Clear message to User

Figure 3

This figure shows DFD level 1 diagram of android security analysis using mobile sandbox

Level 2

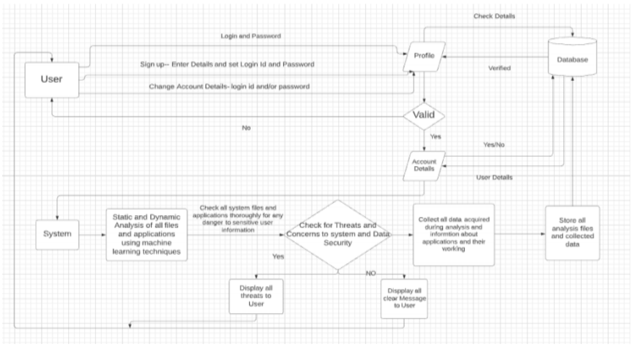

Check Details
Login and Password
User
Sign up-- Enter Details and set Login Id and Password
Change Account Details- login id and/or password
Profile
Verified
Database
Valid
No
Yes
Yes/No
Account Details
User Details
System
Static and Dynamic Analysis of all files and applications using machine learning techniques
Check all system files and applications thoroughly for any danger to sensitive user information
Check for Threats and Concerns to system and Data Security
Collect all data acquired during analysis and informtion about applications and their working
Store all analysis files and collected data
Yes
NO
Display all threats to User
Dispplay all clear Message to User

Figure 4

This figure shows DFD level 1 diagram of android security analysis using mobile sandbox

VII. EVALUATION

For mobile sandbox system the following aspects are being evaluated:

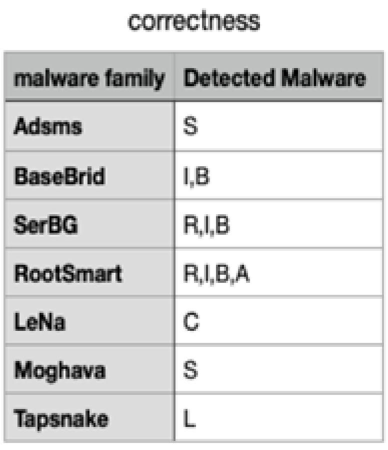
1. Correctness
2. Performance
3. Detectability

A. Correctness:

Using correctness , the entry to the application i.e Mobile sandbox can only be done if the proper actions are being performed by the analysed app .If these actions satisfy one can access the log file of the mobile sandbox .
Methods to check correctness:

1. We can use data visualisation techniques ie Machine learning to find the most sophisticated malware using plots such as scatter plot, heat maps and line plot.
2. The sample we chose consists of various families of android malware which is meant to assure the coverage of malicious actions. The sample data is given below.

Table 1



correctness

malware family	Detected Malware
Adsms	S
BaseBrid	I,B
SerBG	R,I,B
RootSmart	R,I,B,A
LeNa	C
Moghava	S
Tapsnake	L

The following describes different android malware families that are intended to ensure malicious activities are protected.

A -> installs additional apps
B -> Botnet characters
C -> compromises local storage
I -> Steals private related data
L -> steals location data
R -> gains root access
S -> sends SMS messages

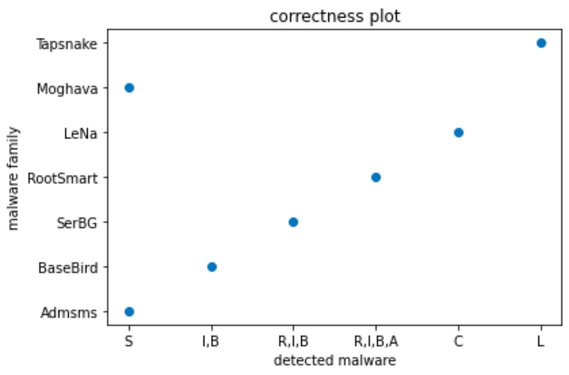


Figure 5

Figure5 describes the correctness scatter plot of the various families of android malware using machine learning techniques.

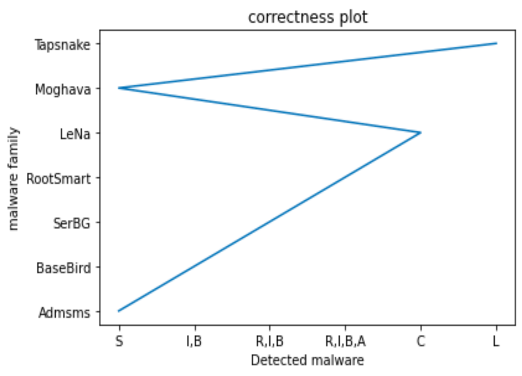


Figure 6

Figure 6 describes the correctness linear plot of the various families of android malware using machine learning techniques.

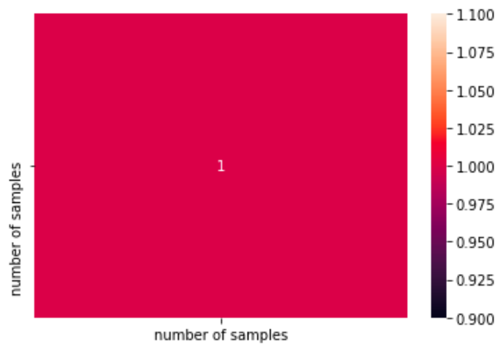


Figure 7

Figure 7 describes the correctness heat map of the various families of android malware using machine learning techniques.

These two methods are used to find the most sophisticated and least sophisticated data .

From the considered data the most sophisticated malware sample is "RootSmart" and the least dangerous Malware family is "TapSmart".

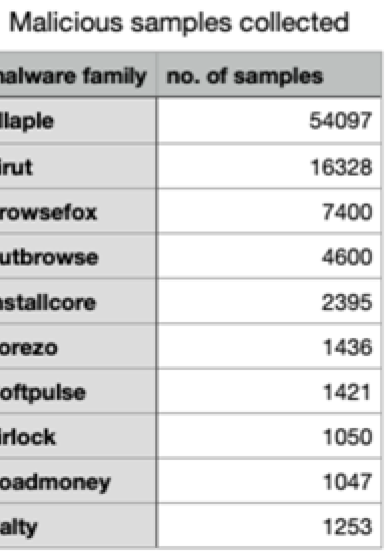
A. Performance :

This system, i.e., The mobile sandbox is powered by Intel Xeon2 . In general, it takes 3s for a mobile sandbox to check viruses in the system and additional 10s - 15s would be required for subsequent static analysis. Cleaning and rebooting a new version takes the machine roughly 3 minutes. It takes a further 4-6 minutes to instal and download the application after rebooting. It takes another 8-10min for the programme execution and monkey runner scripts to run. The application's execution time is solely dependent on the application's duration. It takes another 2s to record all the files after shutting down the emulator. This is how success takes place and it is revised from time to time. For performance enhancement or to reduce the time taken, dynamic analysis may be used for improvement.

B. Detectability :

There are mechanisms in the Windows environment to detect sandbox environments and virtualized ones to transform this method into a malicious programme. Detectability plays a major role in the research platform for safety purposes . The following table describes the malware family and the number of samples of malicious data.

We can use machine learning techniques to plot data and understand which malware family is harmful.



Malicious samples collected

malware family	no. of samples
allaple	54097
virut	16328
browsefox	7400
outbrowse	4600
installcore	2395
eorezo	1436
Softpulse	1421
virlock	1050
Loadmoney	1047
Salty	1253

Table 2

The following table describes the malware family and the number of samples of malicious data.

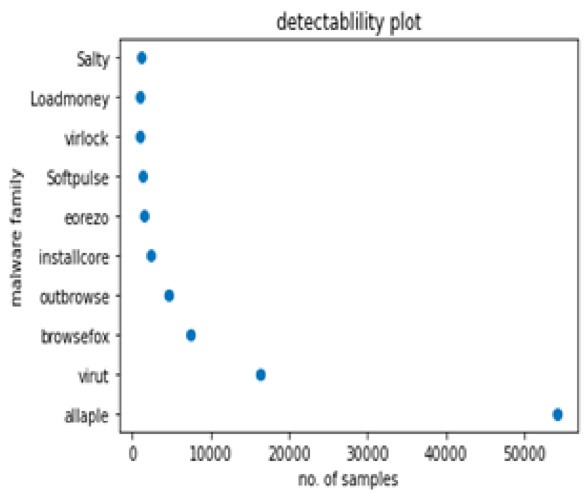


Figure 8

Figure 8 describes the detectability scatter plot of the various families of android malware using machine learning techniques.

```
Text(0.5, 1.0, 'detectablility plot')
```

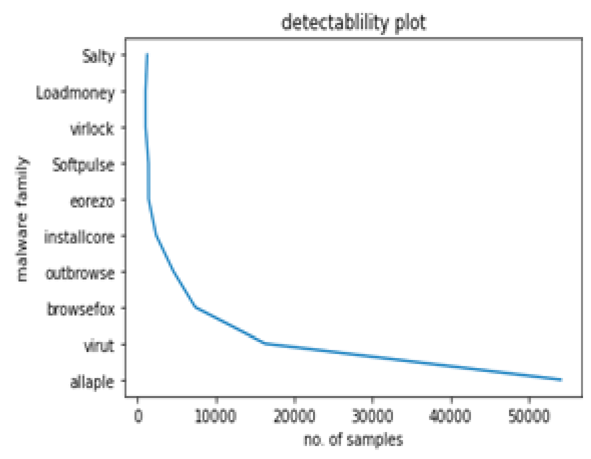


Figure 9

Figure 9 describes the detectability linear plot of the various families of android malware using machine learning techniques.

```
27]: <matplotlib.axes._subplots.AxesSubplot at 0x7fa6f8c38ca0>
```

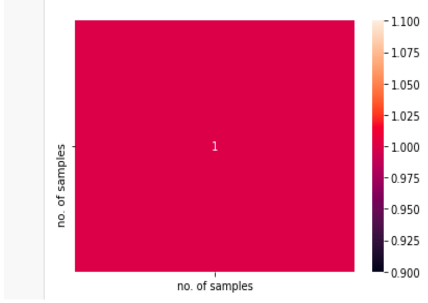


Figure 10

Figure 10 describes the detectability heat map of the various families of android malware using machine learning techniques.

References :
1. Abah, J., e. (2015). A machine learning approach to anomaly-based detection on android platforms. International Journal of Network Security and Its Applications https://www.researchgate.net/publication/286637377
2. A_Machine_Learning_Approach_to_Anomaly-Based_Detection_on_Android_PlatformsShalizi, C. (2016). Logistic regression. Advanced Data Analysis from an Elementary Point of view. https://www.stat.cmu.edu/~cshalizi/ADAfaEPoV/ADAfaEPoV.pdf
3. Feng, Y., e. a. (2014). Apposcopy: semantics-based detection of android malware through static analysis. Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. https://dl.acm.org/doi/10.1145/2635868.2635869
4. Vemparala, S. (2016). Malware detection using dynamibirthmarks. 2nd International Workshop on Security & Privacy Analytics (IWSPA 2016), co-located with ACM CODASPY 2016. https://arxiv.org/abs/1901.07312
5. Ruggieri, S. (2000). Efficient 4.5. Static and Dynamic Analysis of Android Malware https://www.researchgate.net/publication/314521542_Static_and_Dynamic_Analysis_of_Android_Malware
6. Arp, D., e. (2014). Drebin: Efficient and explainable detection of android malware in your pocket. 21stAnnual Network and Distributed System Security Symposium (NDSS). https://www.sec.cs.tu-bs.de/pubs/2014-ndss.pdf
7. Breiman, L., e. (2013). Random forests. Burguera, I., e. (2011). Crowdroid: behavior - based malware detection system for android. Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Device. https://www.ida.liu.se/labs/rtslab/publications/2011/spsm11-burguera.pdf

VIII. CONCLUSION

In this research paper, we have proposed an updated version of Mobile Sandbox which uses both static and dynamic analysis to check for threats in mobile applications.Each application is thoroughly monitored and every activity of the application in inspection is checked to expose all possible security threats to the data and the user.. Although static analysis focuses primarily on the application's source code, dynamic analysis runs the application in a structured environment to show all of its operations. In order to ensure no malicious application is left off-guard, even native API calls have been reviewed. As long as the application is on the computer, users can build an account to sign in and from there, it continuously tracks all the apps added during its time span, reviews app permissions and their functions to provide the user and his/her data with a safe and stable mobile environment.

ACKNOWLEDGEMENT

It is our pleasure to express a deep sense of gratitude to our professor Vani V. for her constant guidance and understanding and also having faith in our capabilities to finish the project.

We would also thank our parents for believing in our abilities and also providing an emotional cushion whenever we stumbled. Cheerful due to a number of our classmates without their support this task wouldn't have been completed. Especially we thank God for providing me with new opportunities each time.

REFERENCES

1. Mobilesandbox : using static and dynamic analysis.http://campar.in.tum.de/pub/spreitzenbarth2015mobilesandbox/spreitzenbarth2015mobilesandbox.pdf

2. Android Developers.: Using the Android emulator https://developer.android.com/guide/developing/devices/emulator.html.
3. Android Developers.: Android platform versions http://developer.android.com/about/dashboards/index.html
4. Static and malware detection for Android malware detection https://scholarworks.sjsu.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1488&context=etd_projects
5. Echtler, F.: ltrace for Android. https://github.com/floe/ltrace

AUTHORS PROFILE

Abhignya Tayi, is currently a 2nd year student pursuing B.Tech in Computer Science at Vellore Institute of Technology, Chennai. She is self directed and self motivated person .She believes in team work.She is a hardworking individual who takes interest in sports ,python development , app development, machine learning, Artificial intelligence , photography and video editing. She takes special interest in cloud computing and have done many labs using quiklab platform .Her research interests include Machine Learning, Network security , Software development , Mobile application and Computer Vision. She aims at working as a Software professional in the IT field at top product based companies .

Munishka Vijayvergiya, is currently a student pursuing B.Tech in Computer Science at Vellore Institute of Technology, Chennai. Her research interests include Android Security, Machine Learning and Blockchain. She is a highly optimistic individual who also takes interest in Public Speaking, Dance and Reading. She firmly believes in the power of hard-work and sincerity at work. She also spends considerable time doing web-development and thoroughly enjoys it. Being an avid reader since her childhood, she also took interest in writing and gained recognition through CBSE expression series during her school time. She aims at working as a Software professional in the IT field while continuing her research-work.