

**ACKNOWLEDGEMENT**

We would like to place on record my deep sense of gratitude to Dr. Hariom Gupta, Director, Jaypee Institute of Information Technology, Noida for his generous guidance, help and useful suggestions.

We express our sincere gratitude to Mr. Anurag Goel, Department of CSE, JIIT, for his stimulating guidance, continuous encouragement and supervision throughout the course of present work.

We also wish to extend our thanks to our batchmates for their insightful comments and constructive suggestions to improve the quality of this project work.

**Signature(s) of Students**

Akshita (9915103108)

Anubhav Johri (9915103111)

Gautam Sharma (9915103118)

Shivangi (9915103131)

**Signature of Mentor**

Anurag Goel

*i*

**ABSTRACT**

Android mobile devices are not immune to ransomware attacks; rather they have less security than windows and IOS. The world has been terrorized by this cyber-threat as it is intractable and only the prevention and detection can give a solution to this crime. Ransomware is a type of malicious software which encrypts or locks the victim’s data and threatens the victim's data or perpetually blocks access to it unless a ransom is paid. Our aim is to detect the presence of Ransomware in the Android devices and stop the malware from taking any unwanted action as well as protect the data. In our project we propose four modules namely, Offline Permission Verification, Threatening Text Detector, Threatening Image Detector, Lock Detector and a feature of Database Storage which works for android as well as desktop application . Threatening Text Detector module has already been made in our Minor I work. These modules focus on a methodology based on static methods. Our application checks for 4 main features of ransomware i.e. permissions requested by the android application, any threatening message in the code of the mobile application in the form of text or image that can be later used to threaten the user to pay the ransom and also detects the locking functions used by the ransomware. With these we will be able to discuss whether an android app is suspicious of containing ransomware or not. These modules are applied on uninstalled apps. Further, we have provided an external feature i.e. to store the results of apps which were found suspicious in RansomwareElite desktop and android application, on an online server containing a database which maintains the records of all the apps, for future use. We have tested our RansomwareElite App on 9 Test Apps to improve its efficiency. Moreover, the detection features are parametric and thus adaptable to future families.

*ii*

**Table of Contents**

Page No.

*Acknowledgement i*

*Abstract* *ii*

*List of Tables* *iv*

*List of Figures* *iv*

**Chapter 1: INTRODUCTION 1**

1.1 What is Ransomware? 1

1.2 Problem Statement 1

1.3 Our approach 1

**Chapter 2: LITERATUREREVIEW** **2**

2.1 Techniques 2

2.1.1 Static Methods 2

2.1.2 Dynamic Methods 3

2.2Background Study 3

**Chapter 3: DESCRIPTION OF PROJECT** **5**

3.1 Requirement Analysis 5

3.1.1 Hardware 5

3.1.2 Software 5

3.1.3 Functional Requirements 5

3.1.4 Non-Functional Requirements 5

3.1.5 Use Case Diagram 6

3.2 Detailed Design 7

3.3 Implementation 9

**Chapter 4: EXPERIMENTAL RESULTS AND ANALYSIS** **12**

4.1 Testing 12

4.1.1 Valid Test Cases 12

4.1.2 Invalid Test Cases 14

4.2 Images of Result Samples 15

**Chapter 5: CONCLUSION AND FUTURE SCOPE** **18**

**References 19**

**Power Point Presentation 21**

**List of Tables**

**Table Title Page**

* 1. Samples and detection of Ransomware family……………………………………1

3.1 Permissions demanded by android applications…………………………………..9

4.1 Valid test cases for the modules…………….……………………………………12

4.2 Valid test cases for database storage of desktop RansomwareElite……………...13

4.3 Valid test cases for database storage of android RansomwareElite…...………....13

4.4 Invalid test cases for the modules…………….…………………………………..14

4.5 Valid test cases for database storage of desktop RansomwareElite...……………14

4.6 Valid test cases for database storage of desktop RansomwareElite...……………14

**List of Figures**

**Figure Title Page**

2.1 Growth in ransomware…………………………………………………………….2

3.1 Use case diagram for RansomwareElite desktop applications………………….....6

3.2 Use case diagram for RansomwareElite android applications…………………….7

3.3 The detailed model of our project…………………………………………………8

4.1 TestApp 1…………………………………………………………………………15

4.2 TestApp 2…………………………………………………………………………15

4.3 TestApp 3…………………………………………………………………………15

4.4 TestApp 4…………………………………………………………………………15

4.5 TestApp 5…………………………………………………………………………16

4.6 TestApp 6…………………………………………………………………………16

4.7 TestApp 7…………………………………………………………………………16

4.8 TestApp 8…………………………………………………………………………16

4.9 TestApp 9…………………………………………………………………………17

*iv*

**1. INTRODUCTION**

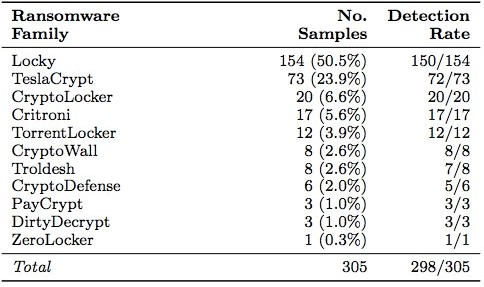
* 1. **What is Ransomware?**

It is any type of malware that demands a sum of money from the infected user while promising to “release” a hijacked resource in exchange of digital currency called ransom. There are two general categories of malware that fall under the “ransomware” label:

• **Lock-screen ransomware**: In lock-screen types of ransomware, the hijacked resource is access to the compromised system.

• **Crypto-ransomware**: In file-encrypting “crypto-ransomware” that hijacked resource is the user’s files. In encryption the victim's files are encrypted, making them inaccessible, and demands a ransom payment to decrypt them.

**Table 1.1**



* 1. **Problem Statement**

Our problem is to identify the presence of the ransomware for uninstalled android applications and hence before their execution on the android device and alert the user and suggest the preventive measures.

* 1. **Our Approach**

We propose an approach based on static methods those are able to detect the ransomware and to identify in the malware’s code the instructions that implement the characteristic of the ransomware on the static level. The static method inspects an android app for its security vulnerability, malicious code, and security threats with respect to data structure and states in a non-runtime environment. Approach is to determine whether an android application attempts to threaten the user, to lock the device, wants to access private areas or a combination of these actions. We have worked on 4 features of ransomware family and proposed 4 modules i.e. Threatening Text Detector, Lock Detector, Offline Permission Verification & Threatening Image Detector. Further we have provided a feature named Database Storage to store all the results of apps which were found suspicious in RansomewareElite Desktop and Android application, in a database on an online server for future use.

*1*

**2. LITERATURE REVIEW**

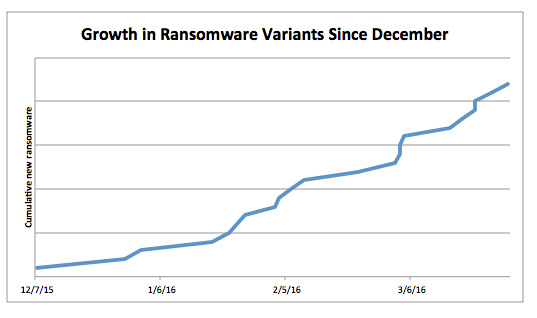


Fig 2.1

**2.1 Techniques**

**2.1.1 Static Method:**

Static analysis inspects app for its security vulnerability, malicious code, and security threats with respect to data structure and states in a non-runtime environment.

**1. Text classification**: This performs linguistic analysis on strings and thus reveals extortion behavior of ransomware. It classifies text into categories.

**2. Image classification**: For this purpose, a collection of logos has been gathered. Then, ICM compares application images with this collection using the Structural Similarity Index Measure algorithm (SSIM) and reports the number of detected images as a feature.

**3. API calls and permissions (APM)**: The APM extracts thelist of permissions from the AndroidManifest.xmlfile and by decompiling an APK, we obtain a list of API methods.

**4. Encryption detection**: We check whether the (disassembled) code of the sample under analysis contains traces of unsolicited file-encryption operations.

**5. Locking detection**: We check if the application under analysis is able to lock the device (i.e., to prevent navigation among activities).

**6. Resources Checking**: Some Android applications have executable files that are encrypted to disguise into become resources files. We can find a ratc file belongs to asserts directory in the extraction APK file, and this ratc file is encrypted but will be decrypted at runtime to get the administrator privileges.

**7. Device Administration APIs Misuse Detector**: Detector parses the Android Manifest file, looking for the declaration of the appropriate Receiver. The detector checks if there exists at least a call to one of the potentially harmful methods of DevicePolicyManager. Important methods are wipeData () and resetPassword ().

*2*

**2.1.2 Dynamic Method:**

Dynamic analysis is usually executed in a runtime sandbox environment to dynamically inspect subtle security vulnerability or part of attack path.

**1. Critical Path and Data Flow**: In android system there are sensitive paths through malicious app could invoke system functions under the directory and also insert gaps into the API call sequences to align them and extract the common sub-sequences which represent the injected malicious behavior.

**2. Malicious Domain Access**: Stolen data by malware would send user’s information to a specific domain. We can collect those malicious domain names to build a blacklist as a factor to distinguish malwares.

**3. Malicious Charges**: In dynamic analysis, we can track the destination of SMS and Call. If there is an unknown destination, the software would be considered as a malware.

**4. Bypassing the Android Permission**: If an application does not announce some permission, but the application execute some related work with those permissions. It’s called bypassing permission.

**5. Text Classification**: We use a Naïve Bayes Classifier in natural language processing (NLP). We train the classifier using phrases labeled by us as threat, law, copyright, porn, and money, which typically appear in scareware or ransomware campaigns. It tells whether a given sentence is “threatening” in any of the languages on which it has been trained on.

* 1. **Background Study**

There have been a few research papers and techniques available on Ransomware detection for android devices as Ransomware has been introduced to about 7-8 years ago.

1. **Automated detection and analysis for Android Ransomware, 2015** [9]

***Summary:*** It detectsthe malicious actions of an app. Static analysis inspects an app for its security vulnerability, malicious code, and security threats with respect to data structure and states in a non-runtime environment. It performs the inspection in a more conservative way with data source.

1. **HELDROID: Dissecting and Detecting Mobile Ransomware, 2015** [10]

***Summary:*** It set a benchmark for all the researches coming up in ransomware detection. It uses 3 detectors, Encryption detector,Lock Detector, Threatening Text Detector and Static-taint analysis which reads the function calls and processes running in the background of the OS.

1. **Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform, 2016**[12]

***Summary:*** It proposed a method that monitors file activities while ransomware accesses and copies them. So the technique detects and eradicates the ransomware by using CPU, I/O usage as well as the information stored in the database. This approach was propped to detect the ransomware in its early stages of malicious activities.

*3*

1. **Ransomware Steals Your Phone. Formal Methods Rescue It, 2016**[11]

***Summary:*** It localizes the peculiar instructions that implement the stages of infection, and the activation of the payload in the code. The approach is structured in three main sub-processes: Temporal Logic Properties Construction, Formal Model Construction, Ransomware family detection.

1. **R-PackDroid, 2017** [13]

***Summary:*** It developed a system that categorized API packages in three phases- Preprocessing- analyzing apps to determine their API packages, Feature Extraction- counting the number of occurrences of each API package in the app and Classification- the above number goes to the statistical classifier and gives the application label. By understanding the packages used in specific applications, we have a better view of the type of actions that an app might execute.

1. **DnaDroid, 2017**[14]

***Summary:*** It overcame the shortcomings of HelDroid used two modules, static and dynamic. This module used text classification, image classification and permission module to categorize the malware as ransomware or not. Once the malware is suspicious then the dynamic module checks for the API calls and then identifies the ransomware.

1. **GreatEatlon, 2017** [15]

***Summary:*** It used same detectors more efficiently by adding a pre-filtering module which reduced processing time and increased the efficiency of the dataset used.

1. **Text Extraction from Text Based Image Using Android, 2018**[16]

***Summary:*** They have integrated the TesseractOptical Character Recognition (OCR) engine and the Google Vision library to develop an android application to capture the images using camera and extract the corresponding text. They have used Optical Character Recognition.

1. **Machine Learning Approach to Detect the Presence of Ransomware in Android Devices, 2018** (to appear)[17]

***Summary:*** Uses two static modules namely, Permission Verification module, which checks the various permissions demanded by the applications and compare them to those of suspicious apps, and Threatening Text Detector module, which scans the code and all the files for texts that can be threatening, to detect the ransomware families in android devices.

*4*

**3. DESCRIPTION OF PROJECT**

**3.1 Requirement Analysis**

**3.1.1. Hardware**

* I3 processor
* 4GB RAM

**3.1.2 Software**

* Android Studio
* NLTK Library
* PIL Liberary
* Windows Operating System
* Android Operating System
* Python version 2.7.14
* Jdk 7
* Requests & Firebase Liberary

**3.1.3 Functional Requirements**

* Apk file of android applications
* Extracting of images
* Extracting of text files
* Extracting the permissions
* Extracting the source code
* Checking for threatening text in the code and images
* Checking for suspicious permissions
* Checking for locking functions and classes
* Giving the details of the detected corrupted app
* Storing the results on the online server

**3.1.4 Non-Functional Requirements**

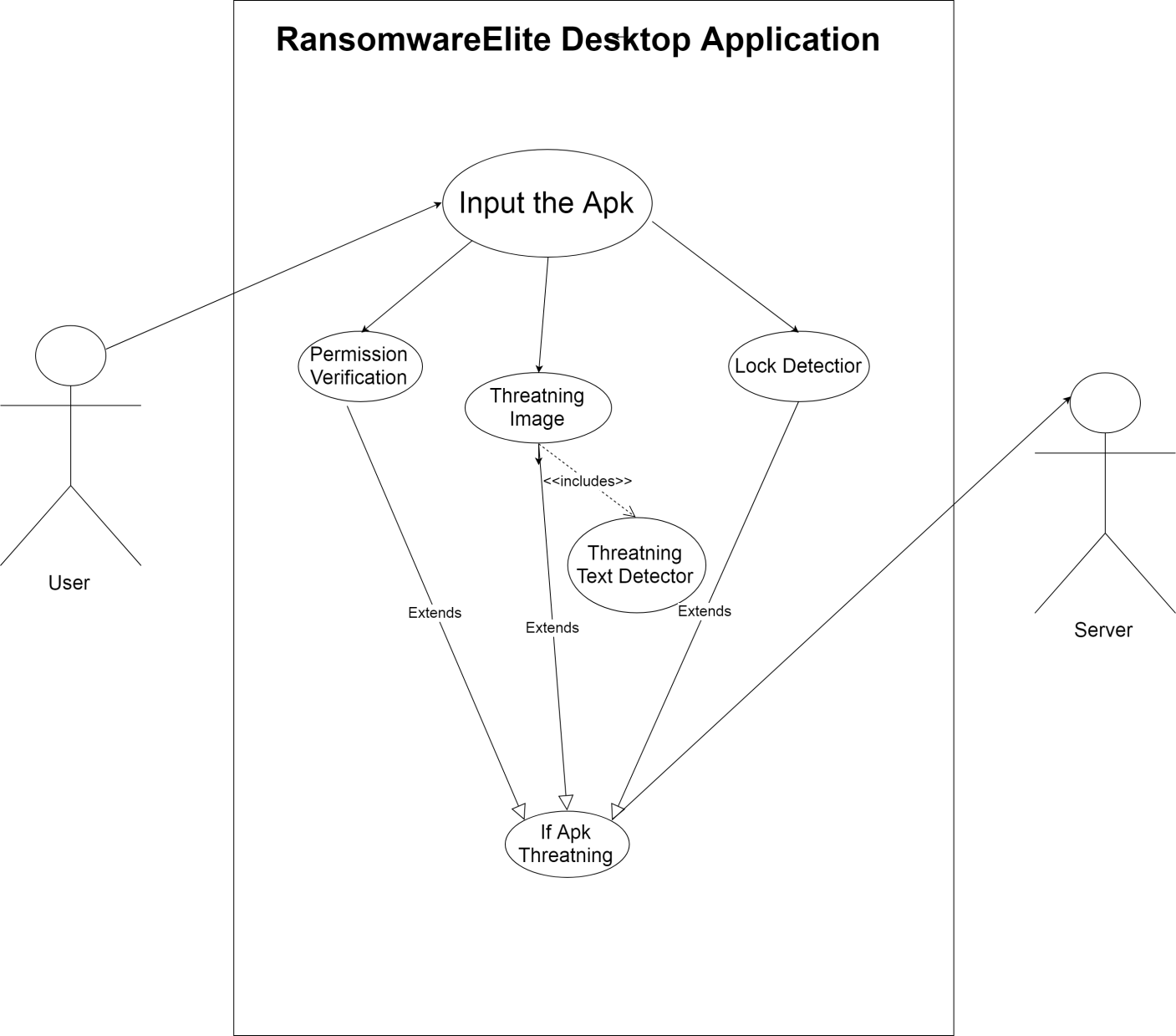
* Validation Process.
* Adaptability
* Checking last scan of a particular application.

*5*

* Cost
* Flexibility of Application

**3.1.5 Use Case Diagrams:**

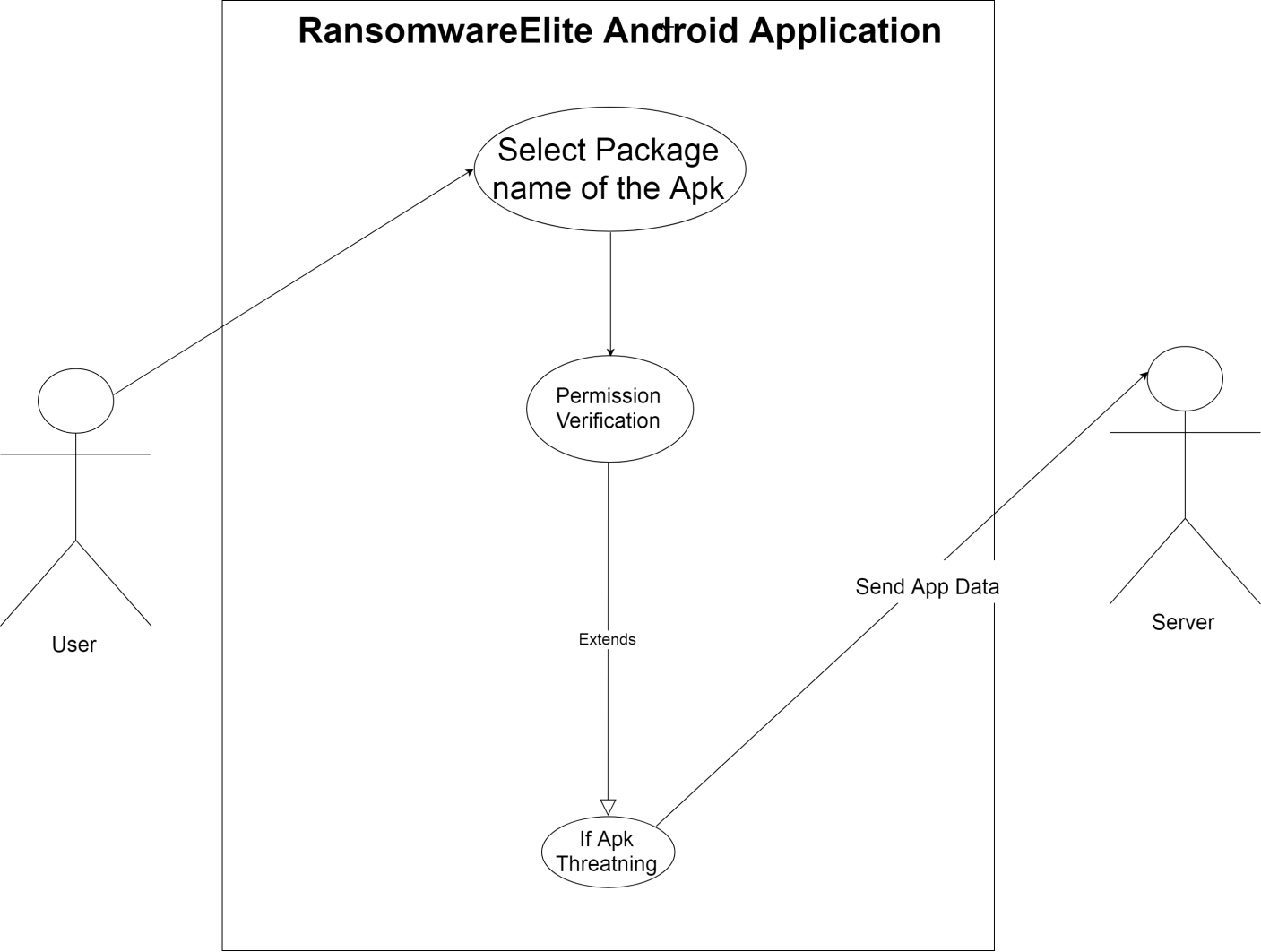
*For RansomwareElite Desktop Application:*



*Fig 3.1*

*6*

*For RansomwareElite Android Applications:*



*Fig 3.2*

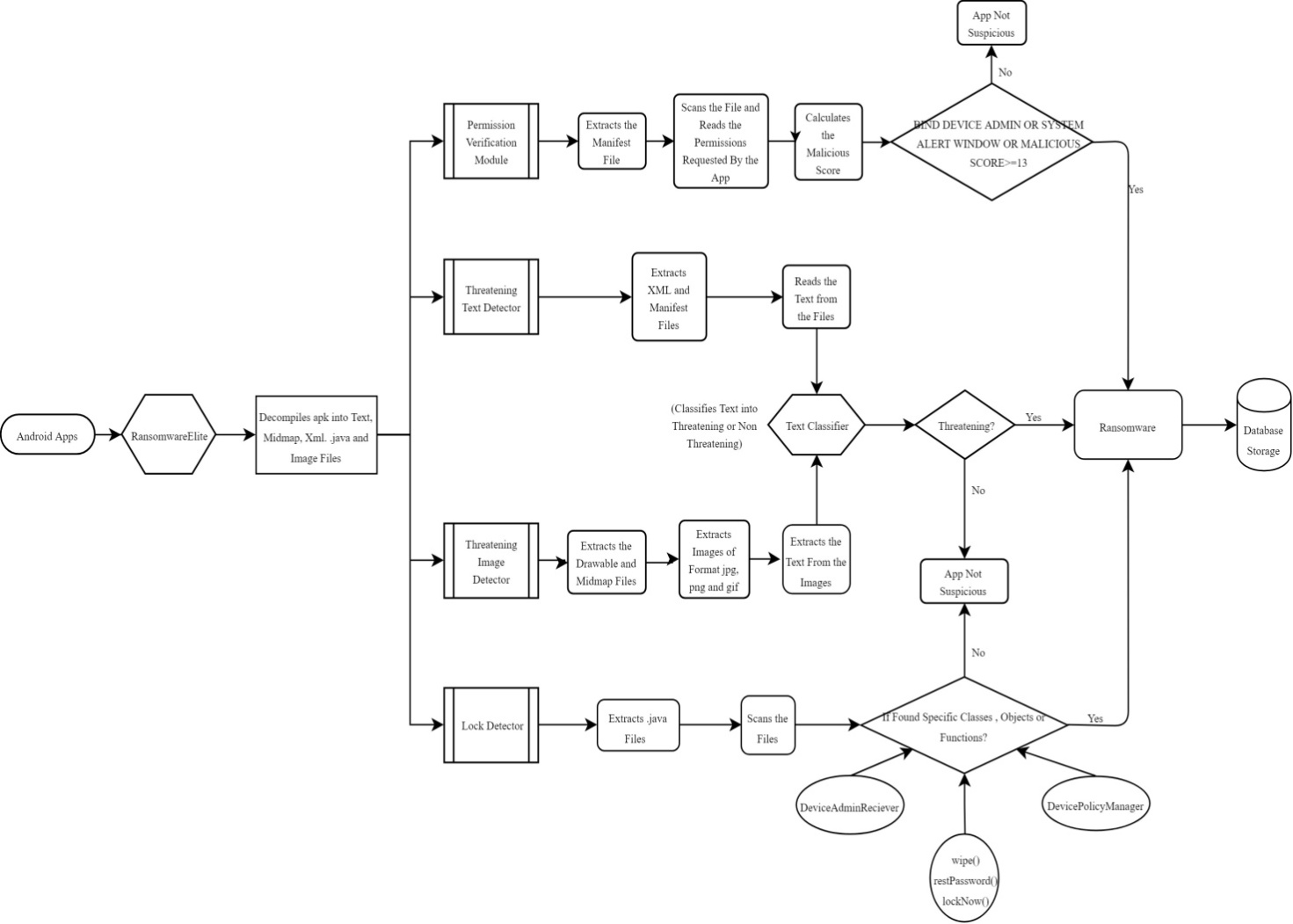
**3.2 Detailed Design**

Our approach is based on static methods which is able to identify in the malware’s code the instructions that implement the characteristic of the ransomware and hence detect its presence on the static level for uninstalled android apps. Purpose is to determine whether an android app attempts to threaten the user either in form of text or image, to lock the device or wants to access private areas or a combination of these actions before it starts working on the android device. We have proposed four modules i.e. Threatening Text Detector, Lock Detector, Permission Verification, Threatening Image Detector and a feature named Database Storage working for both android and desktop RansomareElite application.We have created an app which detects these features of ransomware by verifying the code of the app to find suspicious behavior or any of Ransomware features.

* ***Threatening Text Detector : (Minor 1)***

It detects the threatening text present in the code of a ransomware application which is used to demand ransom from the victim. We have made some changes to this module and reflected the updations.

*7*



*Fig 3.3*

* ***Threatening Image Detector :***

This module detects the threatening text embedded in an image called threatening image, which is used by the ransomware application to threaten the victim later on. Images are used so to not to be detected by various detectors.

* ***Lock detector:***

Ransomware overwrites some functions that are used for locking the device. We have created a module which checks these methods, classes and interfaces used by the Ransomware to lock the user’s device while the attack.

* ***Offline Permission Verification:*** Ransomware requires certain specific permissions to get control on any android device. Those permissions are termed as suspicious permissions. This module checks these permissions demanded by the uninstalled android applications. The list of suspicious permissions is shown in Table 3.1.

*8*

**Table 3.1**

|  |  |  |
| --- | --- | --- |
| **Permission Classification** | **CLASSIFICATION** | |
| TYPE | *PERMISSION* | *SUSPICIOUS* |
| SYSTEM | GET\_TASK |  |
| WRITE\_SETTING |  |
| SYSTEM\_ALERT\_WINDOW |  |
| RECEIVE\_BOOT\_COMPLETED |  |
| READ\_PHONE\_STATE |  |
| READ\_EXTERNAL\_STORAGE |  |
| WRITE\_EXTERNAL\_STORE |  |
| WAKE\_LOCK |  |
| GET\_ACCOUNTS |  |
| BIND\_DEVICE\_ADMIN |  |
| DISABLE\_KEYGUARD |  |
| CAMERA |  |
| INSTALL\_SHORTCUT |  |
| SMS | RECEIVE\_SMS |  |
| SEND\_SMS |  |
| READ\_SMS |  |
| CONTACT | READ\_CONTACTS |  |
| READ\_CALL\_LOG |  |
| CALL\_PHONE |  |
| NETWORK | INTERNET |  |
| ACCESS\_NETWORK\_STATE |  |
| READ\_HISTORY\_\_BOOKMARKS |  |
| ACCESS\_WIFI\_STATE |  |
| LOCATION | ACCESS\_COARSE\_LOCATION |  |
| ACCESS\_FINE\_LOCATION |  |

* ***Database Storage:*** All the results of Desktop RansomwareElite app, whether suspicious due to any reason, are stored in a database created on an online server, with the name of the modules on the basis of which it is found suspicious. Also the results of suspicious apps from the android RansomewareElite app are also stored in this database. This database is being created for the future use. In future, RansomwareElite will check this database first before executing. This will save time and RansomwareElite will work more efficiently.

**3.3 Implementation**

We are implementing static approach in our application to detect the ransomware and alerting the user. There are many features of ransomware families in which we have selected the most important and appropriate. Till now we have implemented four features.

**1. Threatening Text Detector:** We input the app name and apply reverse engineering on it. We first, extract all the xml files and manifest file of the app provided and stores all these files in a folder in a drive in our desktop. Our module then extracts all the text from all the extracted xml files of an app. All

*9*

the extracted texts are stored in a list which is then classified one by one using the text classifier made in python. The module then gives the output if the app has threatening text or not.

**2. Threatening Image Detector:** We first apply reverse engineering on the app provided then we extracted all the layout files i.e. mipmap files and drawable files. The image files are then extracted from the above files with the use of PIL Libraries of python followed by the extraction of texts from the image files. This text is classified by the text classifier**.** The text in the text file is taken as an input by the python script, which classifies the text as threatening or nonthreatening with the implementation of Natural Language Processing which uses NLTK and Textblob libraries in python. Naive Bayes Classifier is used to implement NLP classification. This classifier decides whether the text in the image is threatening or not and hence decides whether the image is threatening or not. If YES, then our applicationshows suspicious app.

**3. Lock Detector:** Ransomware tries to lock the victim’s device and hence threatens the victim to pay the ransom otherwise it won’t provide the key to unlock it. There are some functions, classes and objects which are used to lock the device. For example, the functions which controls the working of our main keys of device like back and home as disabled by overwriting the function as blank and in this way the device is locked in some extent. If such type of functions are overwritten in the code of an app or the objects of some classes are created then that app is concluded as suspicious by the module.

Firstly, the app name will be entered as input to this module. Next, this module applies reverse engineering on that app and hence extracts all the .java files from the apk. Then it will read all the files one by one and the whole code line by line and stores the whole content in the form of a string using a python code. Next it will check whether the string (.java file of the app) contains a

* Class which extends “DeviceAdminReciever”.
* Object of type “DevicePolicyManager”.
* Function like wipeNow(), restPassword() or lockNow().

It will maintain a count which increases by 1 only if it matches the string with the names of above classes and functions. It uses python string libraries and functions for it. At the end the count is checked and if count equals or greater than 1 then the app is concluded as suspicious.

**4. Offline Permission Verification:**First, it will apply the reverse engineering mechanism and extracts the manifest file of the app which has been an input to the module. It will scan the whole file and list all the permissions demanded by the app and store it with the help of python libraries. Then it will compare these permissions with the suspicious permissions mentioned in the table 3.1. If the app demands “BIND\_DEVICE\_ADMIN” or “SYSTEM\_ALERT\_WINDOW” or if it has malicious score equals to

*10*

or greater than 13, then the app is concluded as suspicious and corresponding actions are taken place.

We will combine the results of all modules, hence increase the efficiency of our application, we come to a conclusion whether the app is a Ransomware. Suspicious or Non-Suspicious. If either of the modules gives the app is non-suspicious then we can never claim that the app is a Ransomware.

**5.Database Storage:**

Along with detecting Apk(s) to be Ransomware or not, our RansomwareElite Android & Desktop applications are used to create a database to store the details of malicious apps found by both. The Android app will classify apps as suspicious on the basis of Permission Verification module, whereas Desktop Application uses all the four modules explained above. This module records the threatning apk(s) package names and condition, on which the apk was classified as suspicious, in the online server helping us create a vast database of such threatening apps. The results of both the applications will be sent to Firebase for storage. Firebase is an online cloud Database provided by Google. We have used Requests and Firebase libraries in Python to send data from our desktop application to Firebase database,whereas Android is using FirebaseAuth,FirebaseDatabase libraries. Both the applications are sending only threatning apks(s) data to the server.Based on this dataset we can further extend our research to, understanding the nature of the Ransomware embedded apps,predicting, and classifying various parameters which are commonly found in Ransomware apps. We aim at creating a dataset of more than half a million users with this module for our future work. It’s to apply the 4 modules as we can directly conclude it as Ransomware. It is an external feature which we have provided and can be very useful in future predictions of Ransomware. In future, first we will scan the database and check, if the app name entered is found in the database or not. If it is found, then we won’t need to perform the whole process and results will be concluded immediately from the database.

*11*

**4. EXPERIMENTAL RESULTS AND ANAYSIS**

**4.1 Testing**

To verify RansomwareElite we have created 9 Test Apps on which we have applied our four modules and analyzed the results. These test-apps have been developed keeping in mind basic features possessed by Ransomware family such as asking for malicious permission, threatening image etc. Some test-apps contain suspicious permissions and malicious methods and classes while some contains threatening images or other features. The results have been verified in every case. These were made to ensure that the device on which the test is to be carried out remains safe and protected from the ransomware attack.

**4.1.1 Valid Test Cases:**

*Apps which contains:*

1. Threatening Text Only
2. Threatening Image Only
3. Suspicious Permissions Only
4. Locking Functions Only
5. Both Threatening Image & Threatening Text
6. Both Suspicious Permission & Threatening Text
7. Both Suspicious Permission & Locking Functions
8. All
9. None

Table 4.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | TestApp1 | Threatening Text Present | Threatening Text Present | Pass |
| 2. | TestApp2 | Threatening Image Present | Threatening Image Present | Pass |
| 3. | TestApp3 | Suspicious Permissions Present | Suspicious Permissions Present | Pass |
| 4. | TestApp4 | Locking Functions Present | Locking Functions Present | Pass |
| 5. | TestApp5 | Threatening Image & Text Present | Threatening Image & Text Present | Pass |
| 6. | TestApp6 | Suspicious Permission & Threatening Text Present | Suspicious Permission & Threatening Text Presnt | Pass |
| *12* | | | | |
| 7. | TestApp7 | Suspicious Permission & Locking Functions Present | Suspicious Permission & Locking Functions Presnt | Pass |
| 8. | TestApp8 | Ransomware Present | Ransomware Present | Pass |
| 9. | TestApp9 | This App Is Safe | This App Is Safe | Pass |

**For Database Storage of Desktop RansomwareElite :**

Table 4.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | TestApp1 | Entry Has been Added | Entry Has been Added | Pass |
| 2. | TestApp2 | Entry Has been Added | Entry Has been Added | Pass |
| 3. | TestApp3 | Entry Has been Added | Entry Has been Added | Pass |
| 4. | TestApp4 | Entry Has been Added | Entry Has been Added | Pass |
| 5. | TestApp5 | Entry Has been Added | Entry Has been Added | Pass |
| 6. | TestApp6 | Entry Has been Added | Entry Has been Added | Pass |
| 7. | TestApp7 | Entry Has been Added | Entry Has been Added | Pass |
| 8. | TestApp8 | Entry Has been Added | Entry Has been Added | Pass |
| 9. | TestApp9 | Entry Has been Not Added | Entry Has been Not Added | Pass |

**For Database Storage of Android RansomwareElite :**

Table 4.3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | TestApp1 | Entry Has been Not Added | Entry Has been Not Added | Pass |
| 2. | TestApp2 | Entry Has been Not Added | Entry Has been Not Added | Pass |
| 3. | TestApp3 | Entry Has been Added | Entry Has been Added | Pass |
| 4. | TestApp4 | Entry Has been Not Added | Entry Has been Not Added | Pass |
| 5. | TestApp5 | Entry Has been Not Added | Entry Has been Not Added | Pass |
| 6. | TestApp6 | Entry Has been Added | Entry Has been Added | Pass |
| *13* | | | | |
| 7. | TestApp7 | Entry Has been Added | Entry Has been Added | Pass |
| 8. | TestApp8 | Entry Has been Added | Entry Has been Added | Pass |
| 9. | TestApp9 | Entry Has been Not Added | Entry Has been Not Added | Pass |

**4.1.2 Invalid Test Cases:**

1. App which doesn’t exist
2. Wrong app name

Table 4.4

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | Xyz | App Not Present | App Not Present | Pass |
| 2. | App7 | App Not Present | App Not Present | Pass |
| 3. | Hello | App Not Present | App Not Present | Pass |

**For Database Storage for Desktop RansomwareElite :**

Table 4.5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | Xyz | Entry Has Not been Added | Entry Has Not been Added | Pass |
| 2. | App7 | Entry Has Not been Added | Entry Has Not been Added | Pass |
| 3. | Hello | Entry Has Not been Added | Entry Has Not been Added | Pass |

**For Database Storage for android ransomwareElite :**

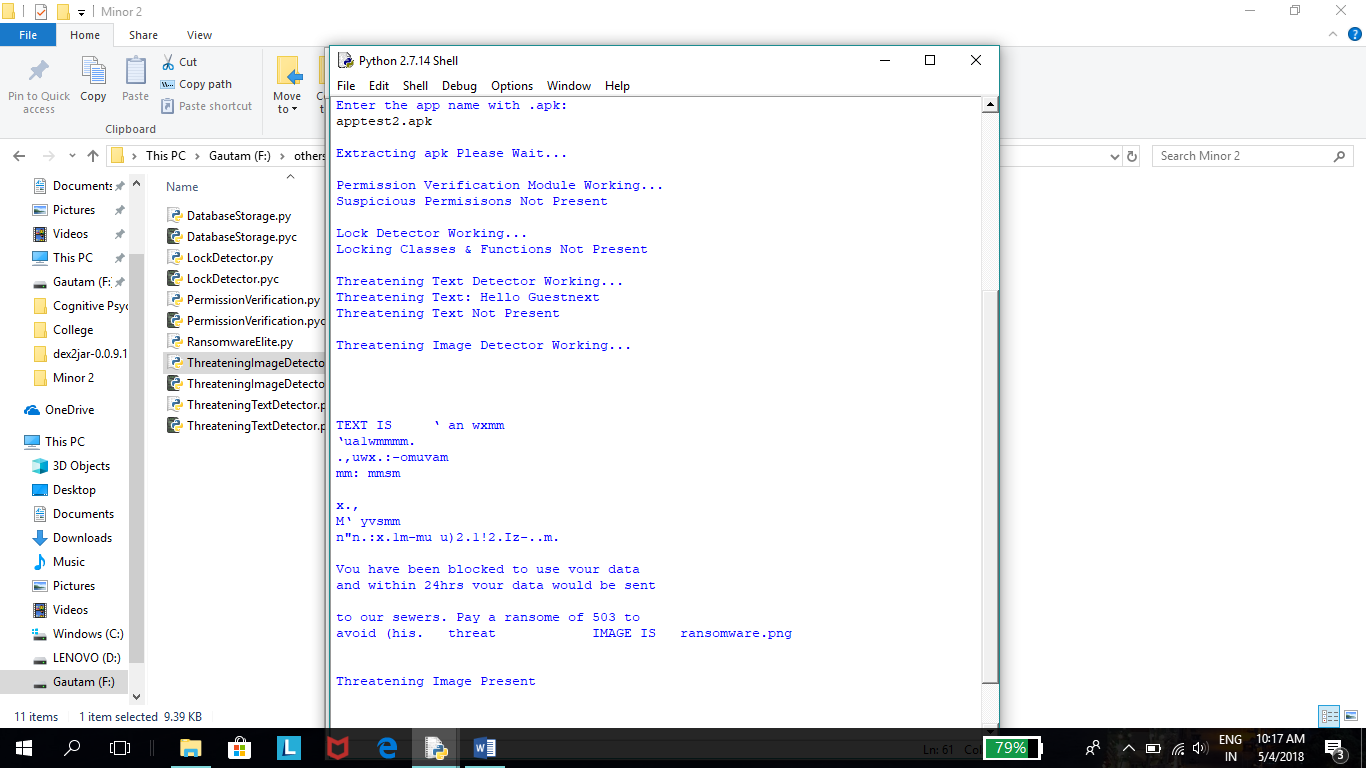
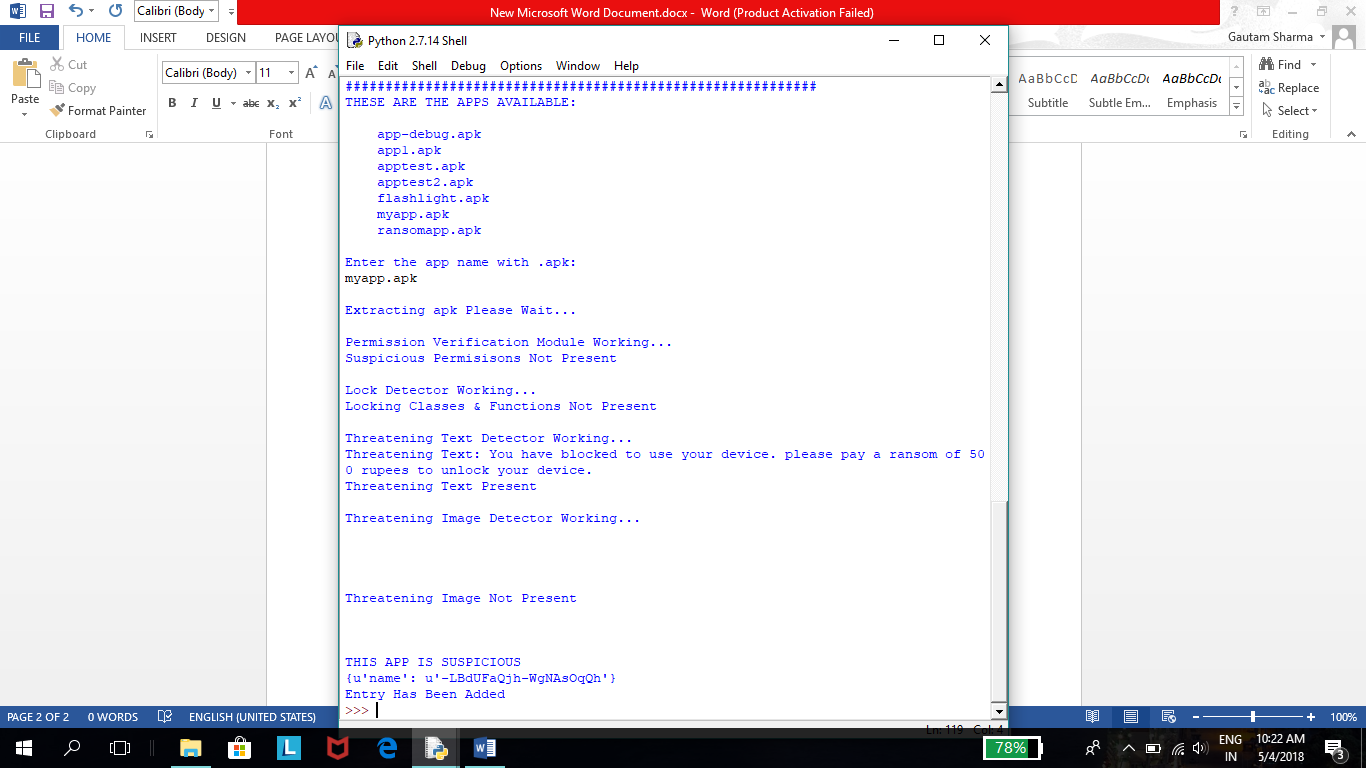
Table 4.6

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** | **Input** | **Expected Output** | **Output** | **Status** |
| 1. | Xyz | Entry Has Not been Added | Entry Has Not been Added | Pass |
| 2. | App7 | Entry Has Not been Added | Entry Has Not been Added | Pass |
| 3. | Hello | Entry Has Not been Added | Entry Has Not been Added | Pass |

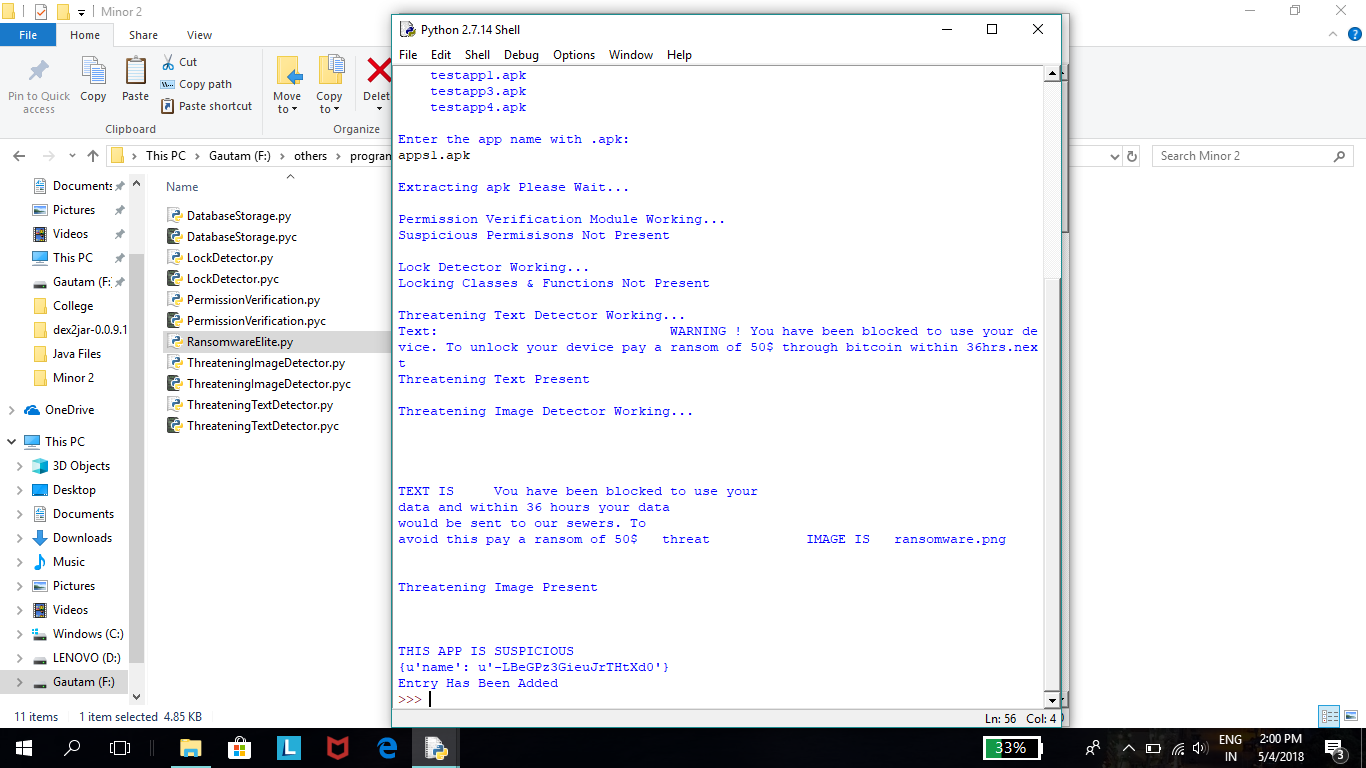
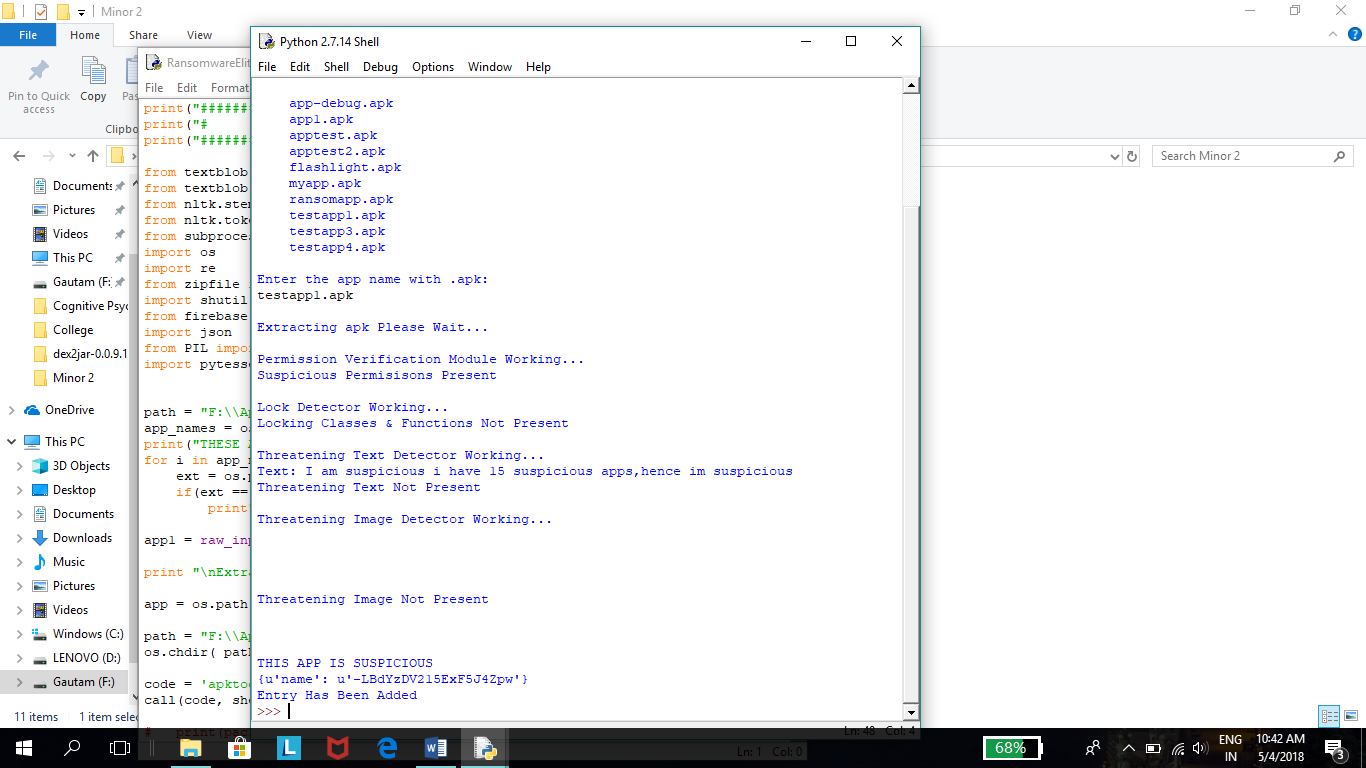
*14*

**4.2 Images of the Results**

*TestApp 1 TestApp 2*

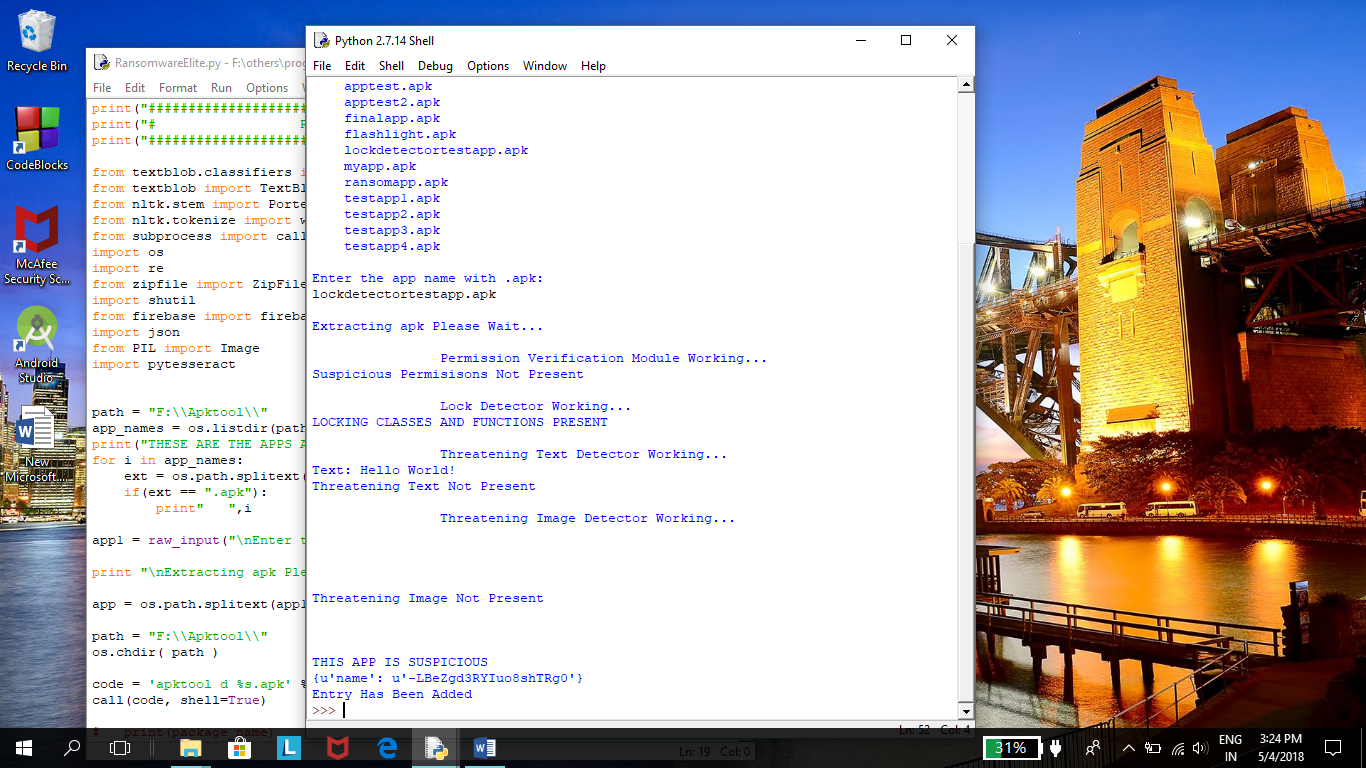
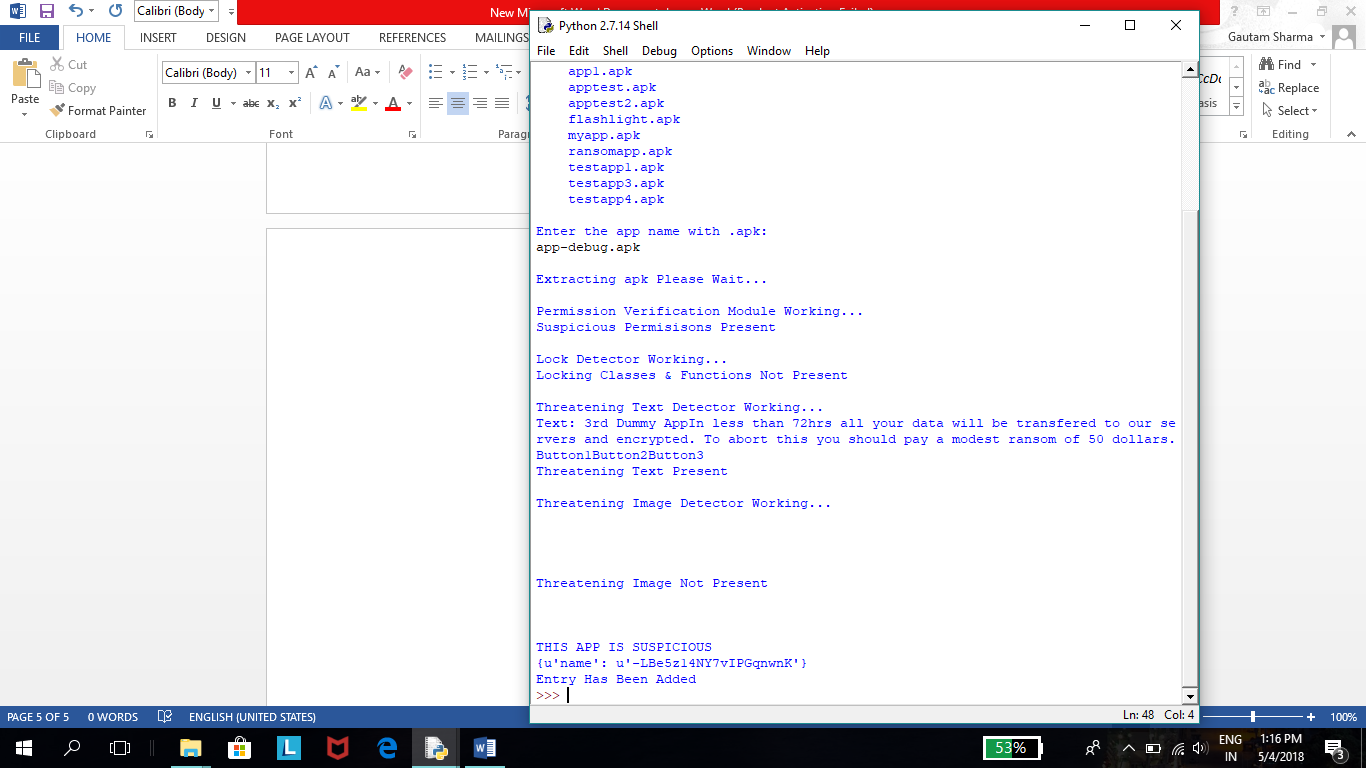


*TestApp 3 TestApp 4*

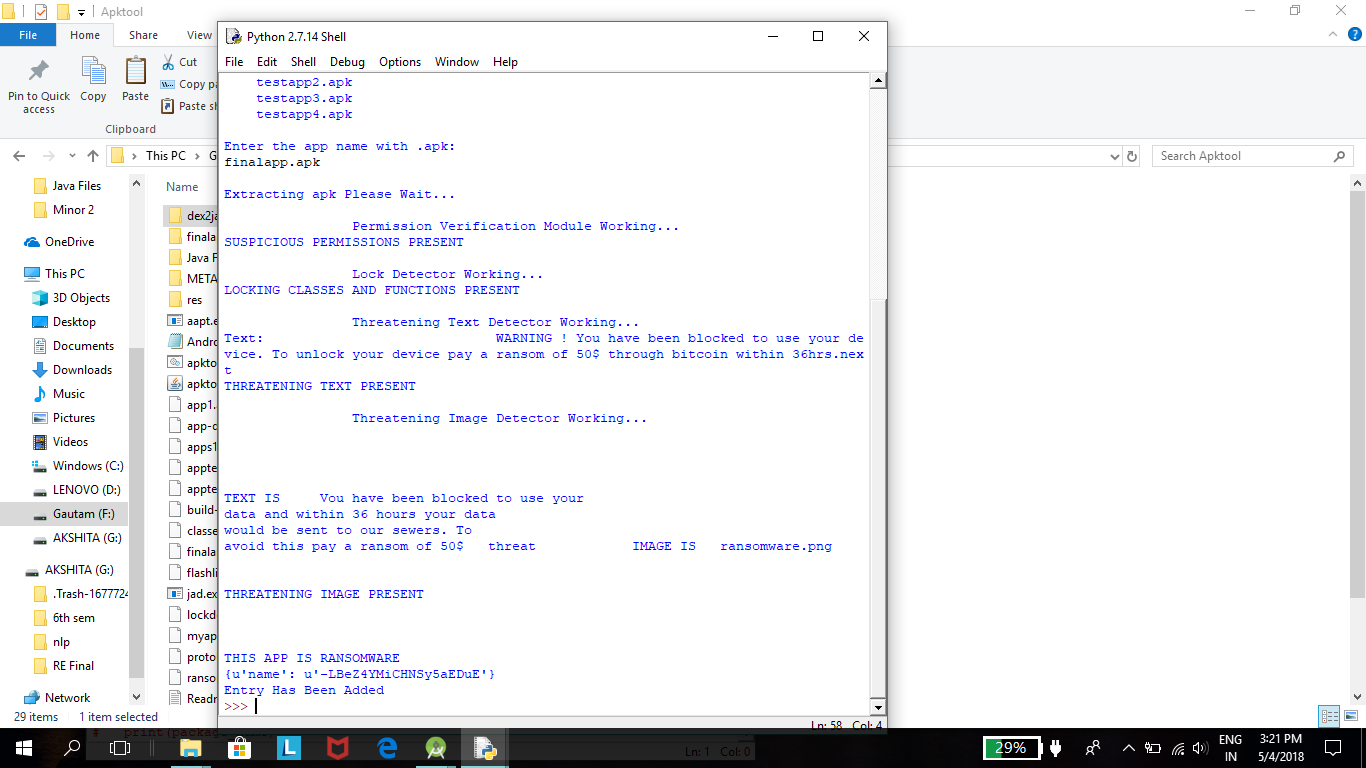
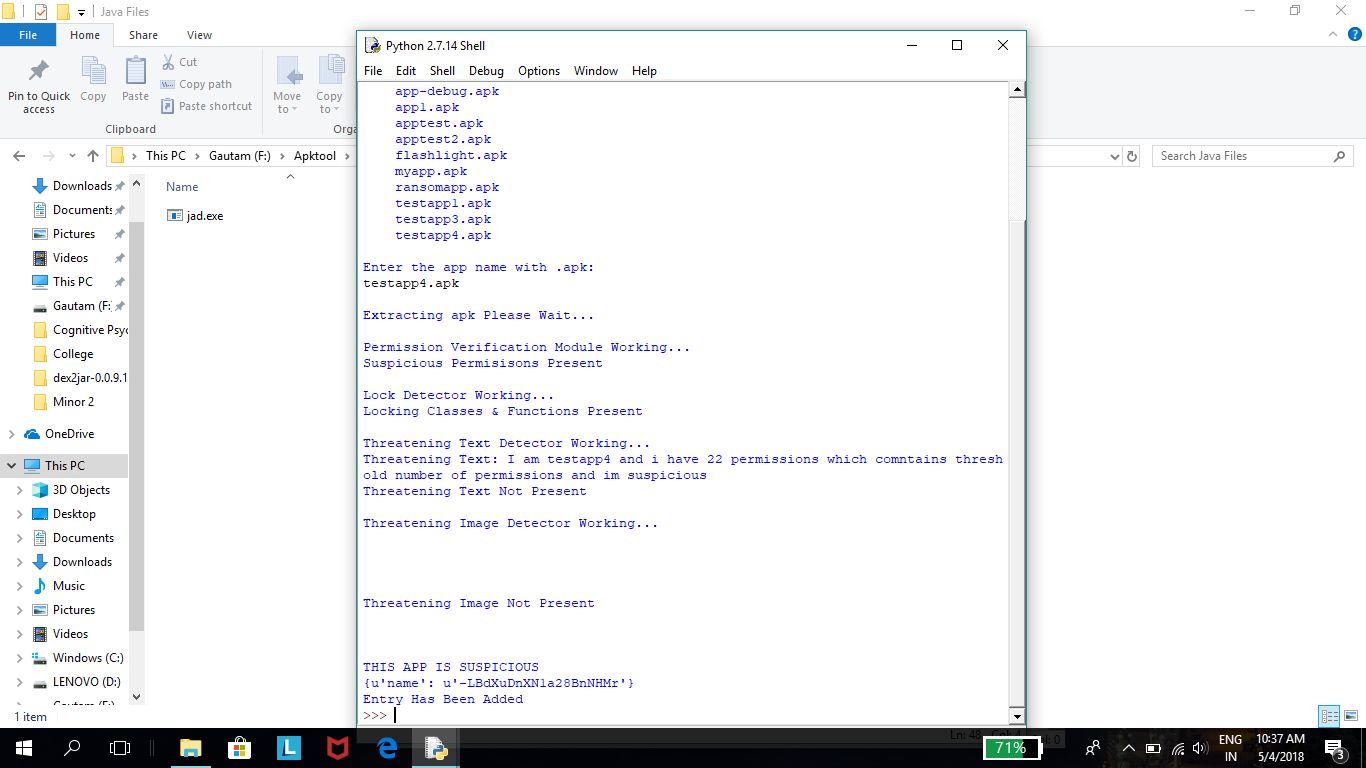


*15*

*TestApp 5 TestApp 6*

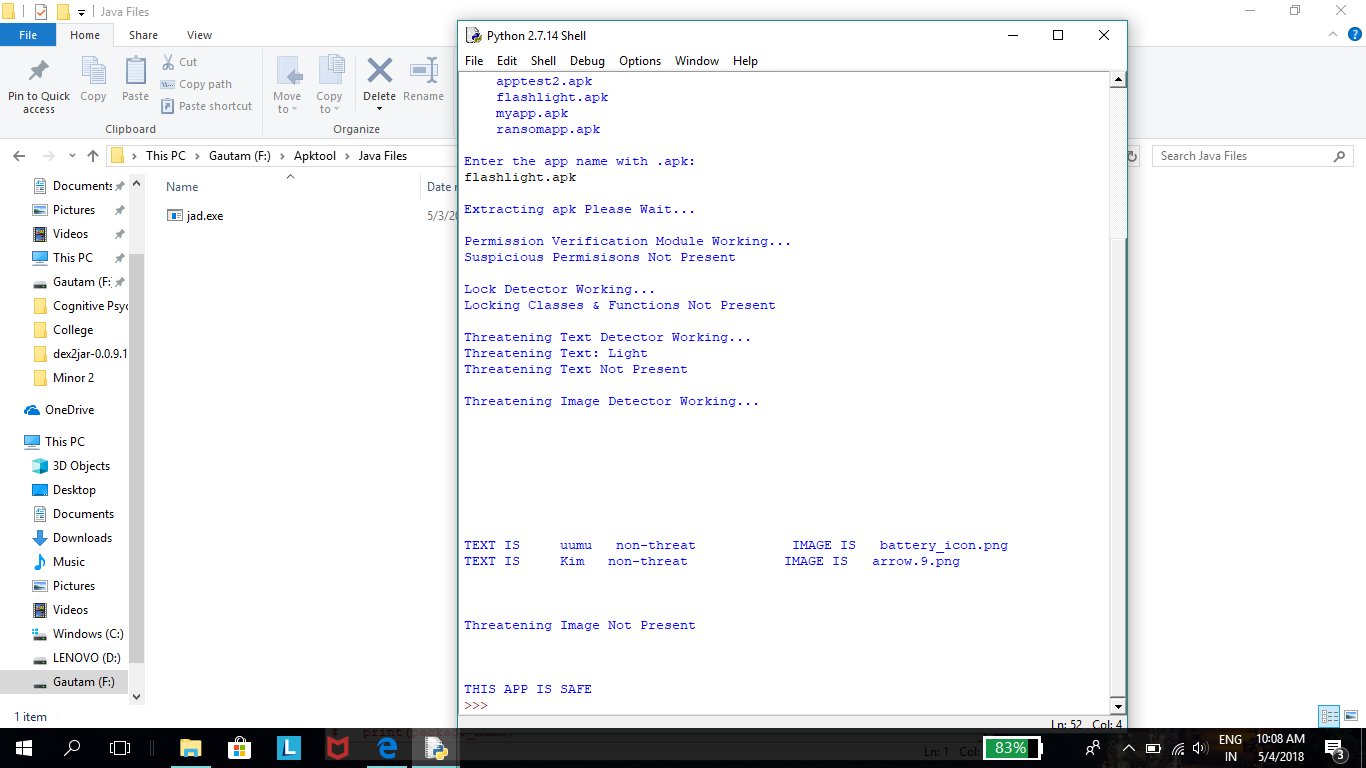
 

*TestApp 7 TestApp 8*



*16*

*TestApp 9*



*17*

**5. CONCLUSION AND FUTURE SCOPE**

Ransomware is a new type of malware that restricts access to the infected smart phone and it demands the user to pay a ransom to the attacker in order to remove the restriction. Ransomware samples are able to encrypt files on the infected device, which become difficult or impossible to decrypt without paying the ransom for the encryption key. And hence a crucial step must be taken in this regard. Static methods can be used to detect such ransomware. This project penetrated and analyzed the characteristics of ransomware and introduced different static methods.

We have created “RansomwareElite” which detects the presence of ransomware for the android applications. It consists of 4 modules: Threatening Text Detector which detects threatening text in an app, Threatening Image Detector which detects threatening images in an app, Offline Permission Verification which checks for suspicious permissions and Lock Detector which checks for locking functions & classes. The result obtained by our four models gave us some idea if the application is malicious or not and can lead to ransomware attack or not. This detection helps to stop the ransomware attack on our android device. RansomwareElite consists of a feature called Database Storage in which all the results of suspicious apps found are further stored in a database created on an online server. It works for android and desktop RansomwareElite app both. We have also tested RansomwareElite with 9 Test App to increase its efficiency.

As a future work, we are creating a database which contains the records of all suspicious apps found, so that to make this app more efficient. The future research also could focus on the combination of static patterns and the APIs collection for detection and can be used for prevention methods also. Not only this, as a future work our solution can be extended using a ransomware dataset for different environment, like Windows Mobile and IOS in order to experiment the portability of our method.

*18*

**References**

**Online**

[1] Ransomware on the rise, January 2015. <http://www.fbi.gov/news/stories/2015/january/ransomware-on-the-rise>

[2]Avast Software. Avast ransomware removal, June 2014. <https://play.google.com/store/apps/details?id=com.avast.android.malwareremoval>

[3]Jarvis, K.: CryptoLocker ransomware, December 2013. <http://www.secureworks.com/cyber/threatintelligence/threats/cryptolockerransomware/>

[4] Android “FBI Lock” malware how to avoid paying the ransom. https://goo.gl/bSgNGz. Accessed 02 Jan 2017

[5]Android ransomware variant uses clickjacking to become device administrator.[https://goo.gl/C1bBEJ. Accessed 02 Jan 2017](https://goo.gl/C1bBEJ.%20Accessed%2002%20Jan%202017)

[6]<http://developer.android.com/about/index.html>

[7] <https://www.lookout.com/resources/reports/mobile-threatreport>

[8] https://cybersecurityventures.com/ransomware-damage-report-2017-5billion/

**Proceedings paper**

[9] T. Yang, Y. Yang, K. Qian, D Chia-Tien, “Automated Detection and Analysis for Android Ransomware”, 1338-1343. 10.1109/HPCC-CSSICESS.2015.39

[10] N. Andronio, S. Zanero, and F. Maggi (B), “HELDROID: Dissecting and Detecting Mobile Ransomware”, IEEE Transactions on Image Processing, Springer International Publishing Switzerland, 2015

[11] F. Mercaldo (B), V. Nardone, A. Santone, and C. A. Visaggio, “Ransomware Steals Your Phone. Formal Methods Rescue It”, IEEE Transactions on Image Processing, IFIP International Federation for Information Processing 2016

[12] S. Song, B. Kim, and S. Lee, “Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform”, IEEE Transactions on Image Processing, March, 2016

[13] D. Maiorca, F. Mercaldo, G. Giacinto, C. A. Visaggio, F. Martinelli, “RPackDroid”, IEEE Transactions on Image Processing, April 03-07, 2017

[14] A. Gharib and A. Ghorbani, “DnaDroid”, IEEE Transactions on Image Processing, Springer International Publishing AG 2017

[15] C. Zheng (B), N. Dellarocca, N. Andronio, S. Zanero, and F. Maggi, “GreatEatlon”, IEEE Transactions on Image Processing, ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017

*19*

[16] Amar G. Waghade, Anuja V. Zopate, Ankita G. Titare and Suraj A. Shelke, “Text Extraction from Text Based Image Using Android,” IEEE International Research Journal of Engineering and Technology (IRJET), vol. 5, issue 3,March 2018.

[17] Anubhav Johri, Shivangi, Gautam Sharma, Akshita, Anuradha Gupta “Machine learning approach to detect presence of Ransomware in android device,” INDIACom 2017, to appear.

*20*