**DETECTION OF ANDROID RANSOMWARE**

**MINOR PROJECT I**

Submitted By:

**Anubhav Johri (9915103111)**

**Akshita (9915103108)**

**Shivangi (9915103131)**

**Gautam Sharma (9915103118)**

Under the Supervision of:

**Mrs. Anuradha Gupta**



**Department of CSE/IT**

**Jaypee Institute of Information Technology University, Noida**

**November 2017**

**ACKNOWLEDGEMENT**

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

We express our sincere gratitude to Mrs. Anuradha Gupta, Department of CSE, JITT, India, for her 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**

Shivangi (9915103131)

Anubhav Johri (9915103111)

Gautam Sharma (9915103118)

Akshita (9915103108)

***i***

***ABSTRACT:***

Along with the rapid growth of new science and technology, the functions of smart phones become more and more powerful. Smart phones bring so much convenience for people and also bring the security risks at the same time. Malicious application has become a big threat to the mobile security. One of these malware is Android Ransomware. “Ransomware” , as the name suggests, 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. In our project we propose a research paper and 3 other modules (Permissions Verification App, Text classification Desktop and Mobile Application) for the justification of our work. The paper focuseson a methodology 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 instructions of the ransomware on the static level. To detect the presence of ransomware application, we have also made 3 modules which detect the permission requested by the android application before executing by reading the manifest file and detect any threatening text in the code of the mobile application that can be later used to threaten the user to pay the ransom. If any android app is requesting for malicious permissions in its manifest file, and found some threatening text to pay ransom requests will be displayed on the screen. With these we will be able to discuss whether an app is suspicious of containing ransomware or not. We will also discuss all the features related to ransomware as understanding its anatomy can only help us realize the solution.Our Research is specific to the ransomware schemes, but it does not rely on any family in particular. Moreover, the detection features are parametric and thus adaptable to future families.

***ii***

**TABLE OF CONTENTS:**

No. Page

*Acknowledgement i*

*Abstract ii*

*List of Tables v*

*List of Figures vi*

**Chapter 1: Introduction 1**

* 1. What is Ransomware? 1
  2. Importance 1

1.3 Approach 2

**Chapter 2: Present Work 3**

2.1 Techniques 3

2.1.1 Static Methods 3

2.1.2 Dynamic Methods 4

2.2 Models Used 5

**Chapter 3: Description of Our Project 7**

3.1 Detailed Design 7

3.1.1 Permission Verification Module 7

3.1.2 Text Classification Module 8

3.2 Suspicious & Non Suspicious Permissions 9

***iii***

**Chapter 4: Requirement Analysis 12**

4.1 Functional Requirements 12

4.2 Non Functional Requirements 12

4.3 User Requirements 12

4.4 Software and Hardware 12

4.5 Use Case Diagram 13

4.5.1 Text Classification Application 13

4.5.2 Permission Verification Application 14

**Chapter 5: Literature Review 15**

5.1 Background Study 15

5.2 Research Papers 15

**Chapter 6: Results & Discussions 17**

6.1 Implementations 17

6.2 Testing Reports 18

**Chapter 7: Conclusion and Future Scope 26**

**References 28**

***iv***

**List of Tables**

**Table Title Page**

1.1 Ransomware Families…………………………………………………1

3.1 Suspicious & Non-Suspicious Permissions……………………………9

6.1 Test Table……………………………………………………………..18

6.2 Test Cases…………………………………………………………….18

***v***

**LIST OF FIGURES**

**Figure Title Page**

2.1 Growth in Ransomware…………………………………………………..3

3.1 Detailed Diagram of Our Application……………………………………7

4.1 Use Case Diagram of Text Classifier…………………………………….13

4.2 Use Case Diagram of Permission Verification…………………………..14

***vi***

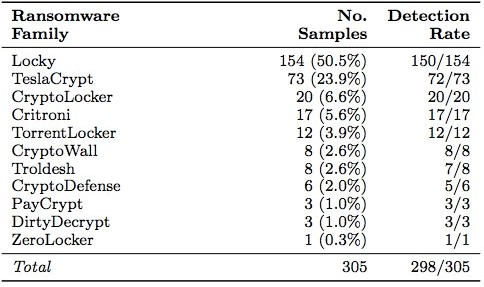
1. ***INTRODUCTION***
   1. ***What is Ransomware?***

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.

The payment is made through bit-coins, iTunes, vouchers, amazon gift cards etc.However paying ransom doesn’t ensure 100% recovery of data in most cases[4]



***Table 1.1***

* 1. ***Importance***

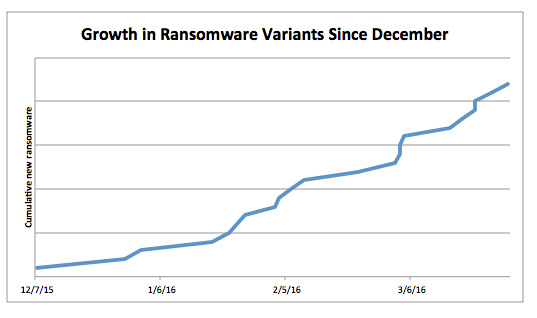
Android operating system is the most extensively used operating system by the smartphone industry. The number of android users has increased manifolds in past few years [1]. Due to this, android became major attraction for the hackers and web criminals to target android devices. Security threats on Android are reportedly growing exponentially [2], however, Google engineers have argued that the malware and virus threat on Android is being [exaggerated](https://en.wikipedia.org/wiki/Fear,_uncertainty_and_doubt) by security companies for commercial reasons, and have accused the security industry of playing on fears to sell virus protection software to users. Google maintains that dangerous malware is actually extremely rare, and a survey conducted by [F-Secure](https://en.wikipedia.org/wiki/F-Secure) showed that only 0.5% of Android malware reported had come from the Google Play store [3]. Android apps run in a [sandbox](https://en.wikipedia.org/wiki/Sandbox_(computer_security)), an isolated area of the system that does not have access to the rest of the system's resources, unless access permissions are explicitly granted by the user when the app is installed. The first ransomware attacking android was found in the middle of 2014[5].It spread immensely in the three years. According to Cyber security researchers at ESET, Android ransomware attacks have risen by over 50% in just a year, peaking in the first half of 2016, and many unfortunately fell for it. In 2016 ransomware exploded about a billion dollars [6]. And it is predicted that the damage caused by android ransomware globally could exceed $5 billion in 2017[7]. This infection has caused a huge economic loss especially to US which is most likely to be followed by India because of extreme usage of android mobile phones by the Indians.

* 1. ***Approach***

The two major approaches for the detection of the ransomware are Static and Dynamic. Static-based analysis means analyzing an app’s code prior to its execution to determine if it is capable of any malicious activities. If the static analysis finds any malicious code, the executable will be stopped from launching. Dynamic-based analysis detection entails the live monitoring of processes, in order to determine if any are behaving with any malicious intent. Any maliciously behaving process will be flagged as dangerous and terminated [8].In our work, we would like to discuss mainly the static method for the detection of the ransomware on android device and provide a solution in order to alert the user regarding the presence of ransomware. 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-runtimeenvironment. Some of the techniques are Text classification, Image classification, API calls and permissions, encryption technique, locking detection, etc. However our main focus will be on the Text classification and Manifest Permissions.

***2***

1. ***PRESENT WORK***



***Fig 2.1***

* 1. ***Techniques***

The two techniques used are Static and Dynamic:

***2.1.1 Static Methods:***

Static analysis is a thorough white-box type of analysis which inspects 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 conservation.

**1. Text classification**: This performs linguistic analysis on strings and thus reveals extortion behavior of ransomware. Notification messages delivered to users commonly have encryption, locking, threatening, pornography, ormoneyrelated content. This module works and can suspect about a file being a ransomware or not.

**2. Image classification**: For this purpose, a collection of logos has been gathered including those of banks, police, government, and famous brands. 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 Android system has a specific permission policy; permissions are granted by the user upon app installation. The APM extracts the list 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. Unsolicited file-encryption operations are usually implemented by reading the storage (e.g., external storage), looping over the files, invoking encryption routines on each of them, and deleting the original files.

**5. Locking detection**: We check if the application under analysis is able to lock the device (i.e., to prevent navigation among activities). This can be achieved in many ways in Android, including the use of the native screen locking functionality, dimming, immortal dialogs, and so forth

**6. Resources Checking**: Some Android applications have executable files that are encrypted to disguise into become resources files and the encryptionexecutable file will be executed when the malware is running. 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. If found, and if the related meta-data contains dangerous policies, then it proceeds to analyze the source code. Whenever the manifest and policy meta-data file analyses are completed and return a positive result, 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 ().

***2.1.2 Dynamic Methods:***

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

***4***

**1. Critical Path and Data Flow**:Android system is based on Linux kernel, and there are sensitive paths such as system executable directory /system/xbin, 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**: An App is charged for a service without clear notification is called chargeware. Chargeware may use charge SMS or Call to pay for services. 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. If an application has administration privileges, it could execute sensitive operation without any related permissions.

**5. Text Classification**: To estimate whether a string contains threatening sentences, we use a natural language processing (NLP) supervised classifier. We train it on generic threatening phrases, similar to (and including) those that typically appear in ransomware or scareware samples.We train the classifier using phrases labeled by us as threat, law, copyright, porn, and money, which typically appear in scareware or ransomware campaigns.Localization. Our NLP classifier supports localization transparently: It tells whether a given sentence is “threatening” in any of the languages on which it has been trained on.

***2.2 Models used***

Till now we have studied a lot of research papers. We have encountered many techniques to detect ransomware:

1. Automated Detection

***5***

2. Model Checking

3. API packages

4. Monitoring Processes

5. Function Calls

6. Multiple Sequence Alignment (MSA)

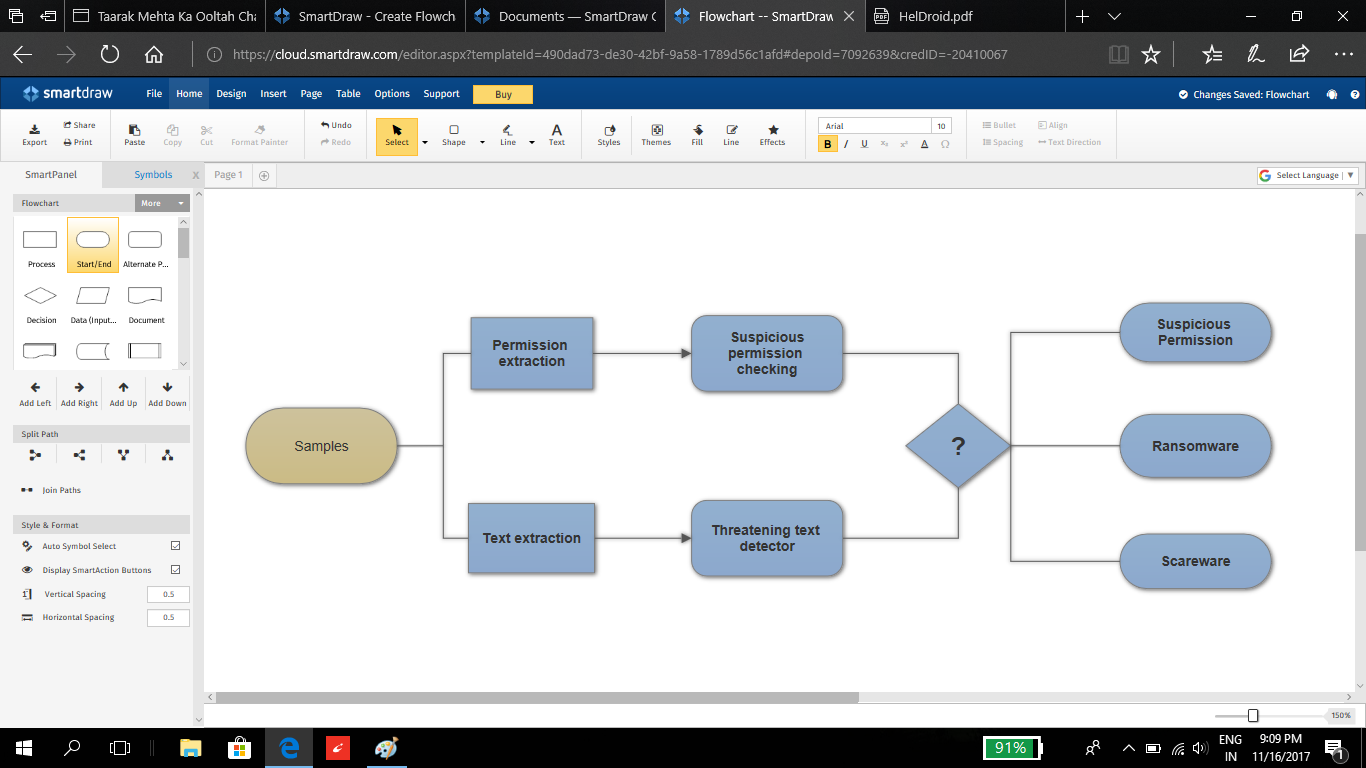
7. NLP Text Classification

8. Static Taint Analysis

***6***

1. ***DESCRIPTION OF OUR PROJECT***
   1. ***Detailed Design***

Our design consists with an approach based on static methods that is able to detect the presence of ransomware on android devices with the help of two different modules, permission verification and text classification.



***Fig3.1***

***3.1.1 Permission verification module***

The purpose of this module is to check and verify whether an application wants to access some particular permission(s). Ransomware is a malware which requires certain specific permissions to hack our android device and lock it. Those permissions are termed as suspicious permissions. We have gathered information about such permissions and created a table (Table 1) which distinguishes suspicious and non-suspicious permissions.

The app “Permission Verification” works in some basic steps. It has two major parts: To check a specific app and to verify the whole system. In the first part, it will show the list of packages of all applications installed in our device. Then it takes input from the user and shows the corresponding result. In the other part the click of the button will show the result containing the list of those applications which are suspicious. In the code behind it uses a “PackageManager” class to extract a list of all those packages. Then it will access the manifest file of the packages and extracts all the permissions with the help of a function getPackageInfo() and stores it in a list. Then it compares all the permissions in that list with some specific permission(s), mentioned in the table. If that list contains 80% of those suspicious permissions or if it contains “BIND\_DEVICE\_ADMIN” then it will conclude that application to be suspicious otherwise not and corresponding message will be shown on the screen.

***3.1.2 Text classification module***

This module is written in Python and classifies the provided texts into threatening and non-threatening texts. We have implemented Natural Language Processing by using NLTK and Textblob libraries in python. We have used Naive Bayes Classifier to implement this NLP classification. This module consists of two phase explained in detail:-

**Desktop App:**

It inputs an apk of an app uses apktool that extracts the all the xml files and manifest file of a particular apk file and then uses the text classifier to classify the texts in the main activity of that app as mostly all the threatening texts in an app are found in the main activity itself. The module then gives the output if the app has threatening text or not.

**Mobile App:**

We, have made an App which is detecting whether the text file in the app package is threatening or non-threatening. For classifying the text file we have used python code implementing NLP Text Classifier to classify the text as threatening or non-threatening. We have used our PC as a Server and have done server side programming in java to receive and send the text file. Our PC is running our python code. Firstly, when we login with our app rawextract() method is called which extracts the text file from the raw folder in the app package, then Volley.onResponse() method sends the text

Machine learning approach to detect the presence of ransomware in android device file to the PC over same network and creates a file in any drive in our PC. Then, the text in the text file is taken as an input bythe python code and the result is extracted and sent back to our app as a response.

***8***

* 1. ***Suspicious And Non-Suspicious Permissions***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***TYPE*** | ***PERMISSION*** | ***BEHAVIOUR*** | ***NON-SUSPICIOUS***  ***PERMISSIONS*** | ***SUSPICIOUS***  ***PERMISSIONS*** |
| SYSTEM | GET\_TASK  WRITE\_SETTINGS  SYSTEM\_ALERT\_WINDOW  RECEIVE\_BOOT\_COMPLETED  READ\_PHONE\_STATE  READ\_EXTERNAL\_STORAGE  WRITE\_EXTERNAL\_STORAGE  WAKE\_LOCK  GET\_ACCOUNTS  BIND\_DEVICE\_ADMIN  DISABLE\_KEYGUARD  CAMERA  INSTALL\_SHORTCUT | Allows an application to get information about currently or recently running tasks.  Allows an application to read or write system settings  Allows an application to alert system  Allows an application to receive the ACTION BOOT COMPLETED that is broadcasted after the system finishes booting  Allows read only access to phone state  Allows an application to read from e0ternal storage  Allows an application to write to e0ternal storage  Allows using PowerManagerWakeLocks to keep processor from sleeping or screen from dimming  Allows access to the list of accounts in Accounts service  Must be required by device administration receiver to ensure that only the system can interact with it  Allows applications to disable the keyguard if it is not secure  Required to be able to access the camera device.  Allows an application to install a shortcut in Launcher | 0  0  0  0 | 0  0  0  0  0  0  0  0  0 |
| SMS | RECEIVE\_SMS  SEND\_SMS  READ\_SMS | Allows an application to receive SMS messages  Allows an application to send SMS messages  Allows an application to read SMS messages | 0  0 | 0 |
| CONTACT | READ\_CONTACTS  READ\_CALL\_LOG  CALL\_PHONE | Allows an application to read user’s contacts data  Allows an application to read the user’s call log  Allows an application to initiate a phone call without going through the Dialer user interface for the user to confirm the call | 0  0  0 |  |
| NETWORK | INTERNET  ACCESS\_NETWORK\_STATE  READ\_HISTORY\_BOOKMARKS  ACCESS\_WIFI\_STATE | Allows applications to open network sockets  Allows applications to access information about  networks  Allows an application to read the user’s browsing history and bookmarks  Allows applications to access information about Wi-Fi networks. |  | 0  0  0  0 |
| LOCATION | ACCESS\_COARSE\_LOCATION  ACCESS\_FINE\_LOCATION | Allows an app to access appro0imate location.  Allows an app to access precise location. |  | 0  0 |

***Table 3.1***

***11***

1. ***Requirement Analysis***
   1. ***Functional Requirements***

The requirements those are essential for the working of a device. In our project those are:

1. Scanning of the doubtful application.

2. Checking of permissions asked by the new applications.

3. Checking status of the app.

4. Raising an alarm and notifying the user.

5. Giving the details of the detected corrupted app.

* 1. ***Non Functional Requirements***

1. Validation Process.

2. Checking last scan of a particular application.

3. Cost

4. Flexibility of app

5. Storage of data

* 1. ***User Requirements***

1. False negative rate should be less

2. Easy to operate

3. Providing simple user interface

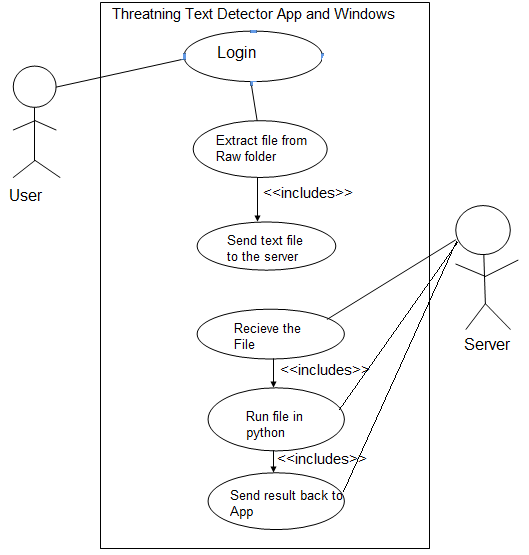
4. Quick and response

* 1. ***Software and Hardware***

Hardware: I3 processor, DDR3 Memory, 4GB RAM

Software:Java, Python2.7, ApkTool, MyEclipse8.6, Android Studio

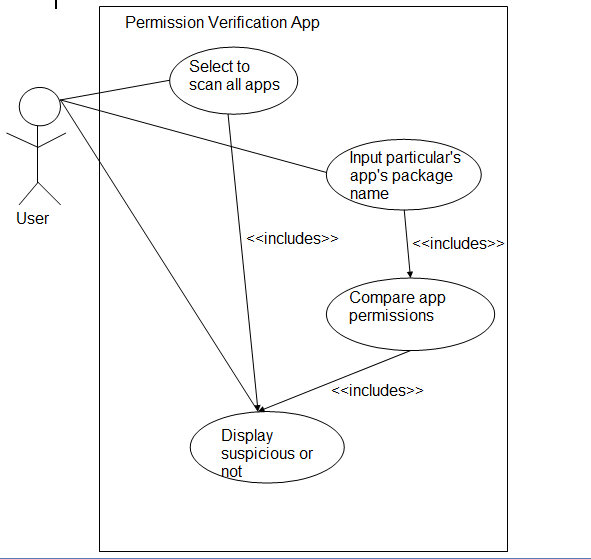
* 1. ***Use Case Diagram***
     1. ***Text Classification Application***

******

***Fig 4.1***

***13***

* + 1. ***Permissions Verification Application***



***Fig 4.2***

***14***

1. ***LITERATURE REVIEW***
   1. ***Background Study***

Android, Core java,Python, Machine Learning, Ransomware attacks and their techniques, Research papers of different institutions, Books, Prevention and detection methods used till now.

* 1. ***Research Papers***

1. DNA-DROID

Authors - Amirhossein Gharib(B) and Ali Ghorbani

1. Experimental Analysis of Ransomware on Windows and Android Platforms: Evolution and Characterization

Authors - Monika, Pavol Zavarsky, Dale Lindskog

1. HELDROID: Dissecting and Detecting Mobile Ransomware

Author:-Nicol´o Andronio, Stefano Zanero, and Federico Maggi(B)

1. GreatEatlon: Fast, Static Detection of Mobile Ransomware

Author-Chengyu Zheng(B), Nicola Dellarocca, Niccol`o Andronio, Stefano Zanero, and Federico Maggi

1. The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform

Author-Sanggeun Song, Bongjoon Kim, and Sangjun Lee

1. R-PackDroid

Authors-Davide Maiorca, Francesco Mercaldo, Corrado Aoron Visaggio, Fabio Martinelli, Giorgio Giacinto

1. Automated Detection and Analysis for Android Ransomware

Authors- Tianda Yang, Yu Yang, Kai Qian, Dan Chia-Tien, Ying Qian, Lixin Tao

1. Ransomware Steals Your Phone. Formal Methods Rescue It

Authors- Francesco Mercaldo (B), Vittoria Nardone, Antonella Santone, and Corrado Aaron Visagg

***16***

1. ***Result and discussions***
   1. ***Implementation***

Firstly, the permission verification app will read all the permissions sought by the apps to be tested and check if they match with malicious column of the permission table then the testing app is considered suspicious. Secondly, we will extract text statements from the application to be tested and feed them to the text classifier that identifies whether the text is ‘threatening’ or not with the help of machine learning. If the text is labeled as ‘threatening’ then the app is considered to be suspicious. If both the methods suggest the testing app to be suspicious then there are 60% chances that the app is malicious and belongs to the Ransomware family.

To carry out the above test we have created some dummy apps on which we have applied our two methods and analyzed the result identifying whether the result declares these apps “suspicious” or not. These dummy applications have been developed on the features possessed by any malware belonging to the ransomware family. These apps are designed in such a way that it experiments every test case for our project. Some apps contain suspicious permissions or some contains threatening text or both. 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.

Other than this we have also tried and experimented our methodology on 28 other android mobile devices to scan the whole device as well as specific applications. And to our results we have found one device unsafe. The suspicious application found is “Mini Militia” a gaming application. Hence, we have verified our work on every aspect.

This result will help the user in realizing the contributor of ransomware before it completely attack the device and hence the corresponding app will be stopped or uninstalled to prevent damage and loss of data.

* 1. ***Testing Reports***

***Table 6.1-Test Table***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Test Id** |  | **Modules** |  | **Type Of Testing** |
|  |  |  |  |  |
| 1. |  | Permission Verification |  | White Box |
| 2. |  | Mobile Text Classifier |  | Black Box |
| 3. |  | Desktop Text Classifier |  | Black Box |

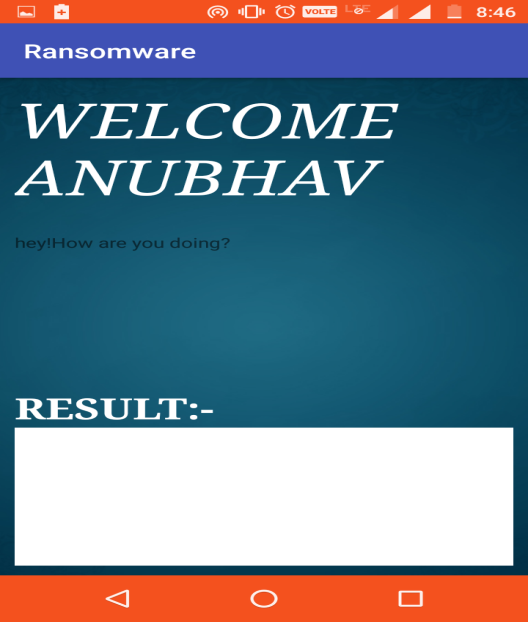
***Table 6.2-Test Cases***

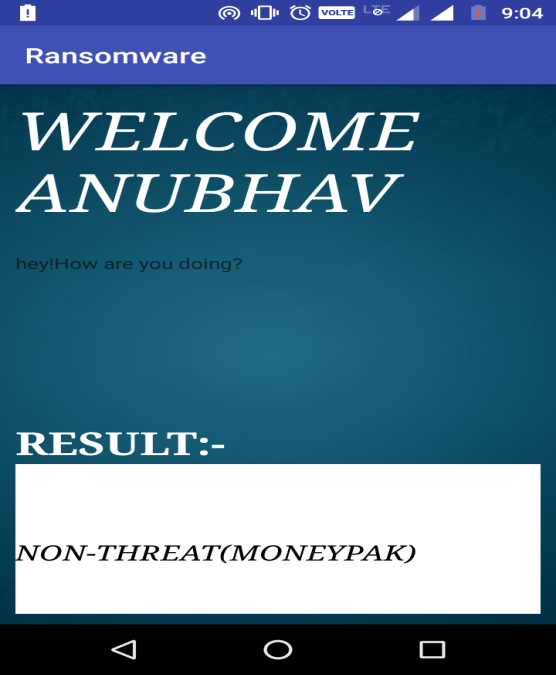
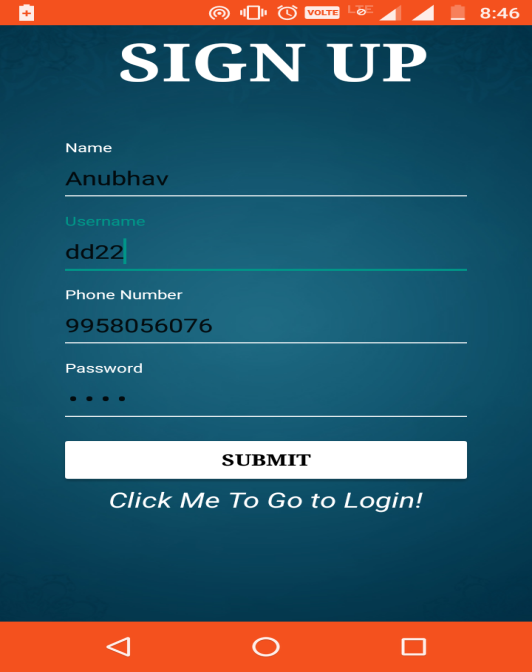
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Id** |  | **Input** |  | **Expected Output** |  | **Status** |
|  |  |  |  |  |  |  |
| 1 |  | Dummy Apps Input |  | Suspicious/ Non-Suspicious |  | Pass |
| 2 |  | 40 Android Devices |  | 1 Suspicious Device |  | Pass |
| 3 |  | Threshold : 0.95 |  | 23 Attributes |  | Pass |
| 4 |  | Threshold : 0.7 |  | 13 Attributes |  | Pass |

**Text Classification Mobile Application:**

First sign up and then login to the app. It will take the raw file and pass it to server tells whether it has threatening messages or not and shows the result on app.

***18***

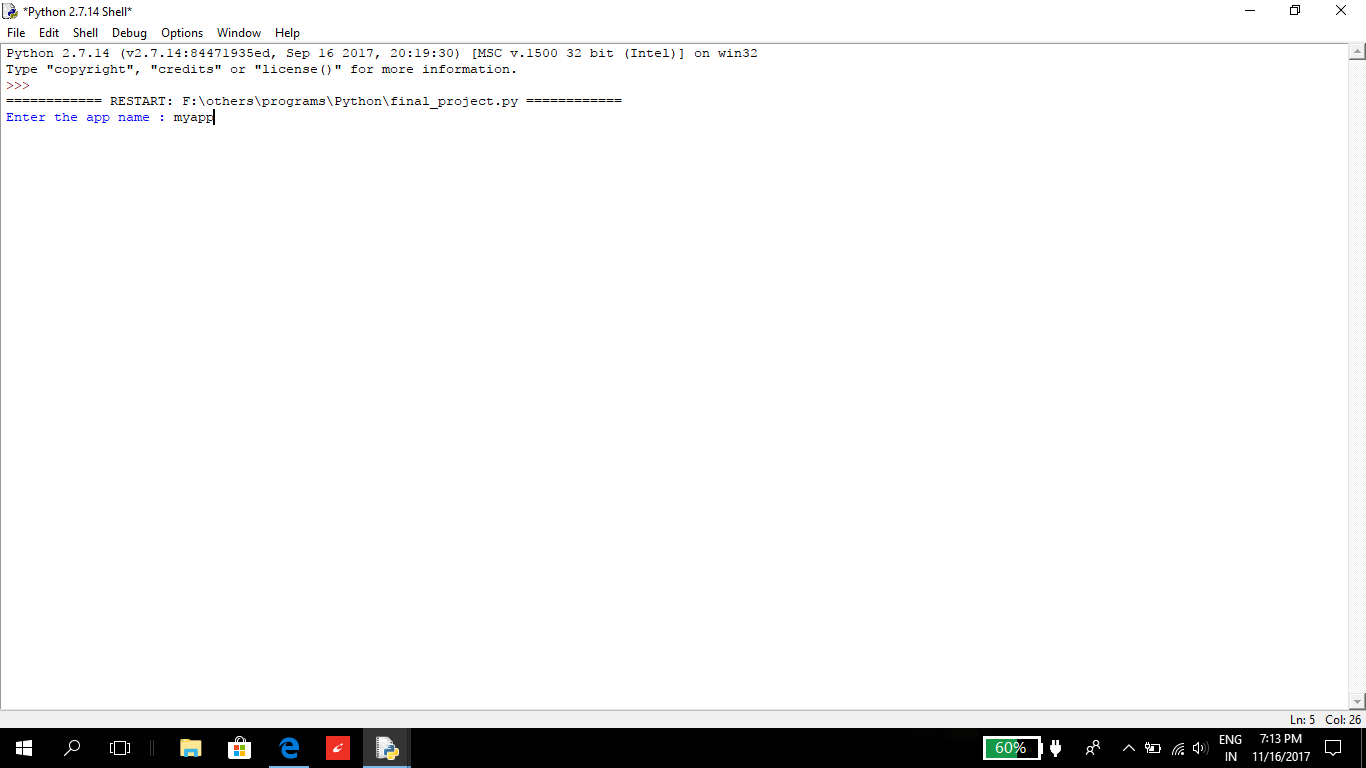
******

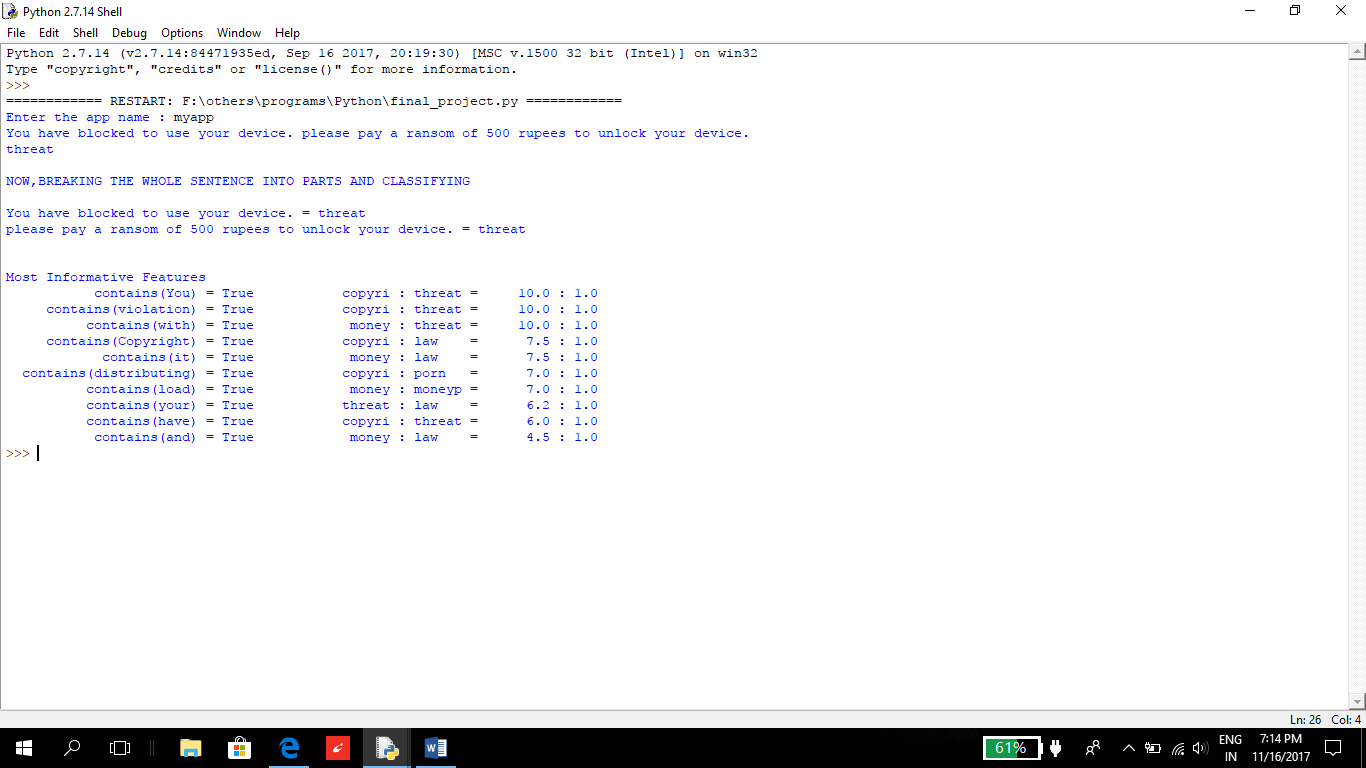
**

**Text Classification Desktop Application**

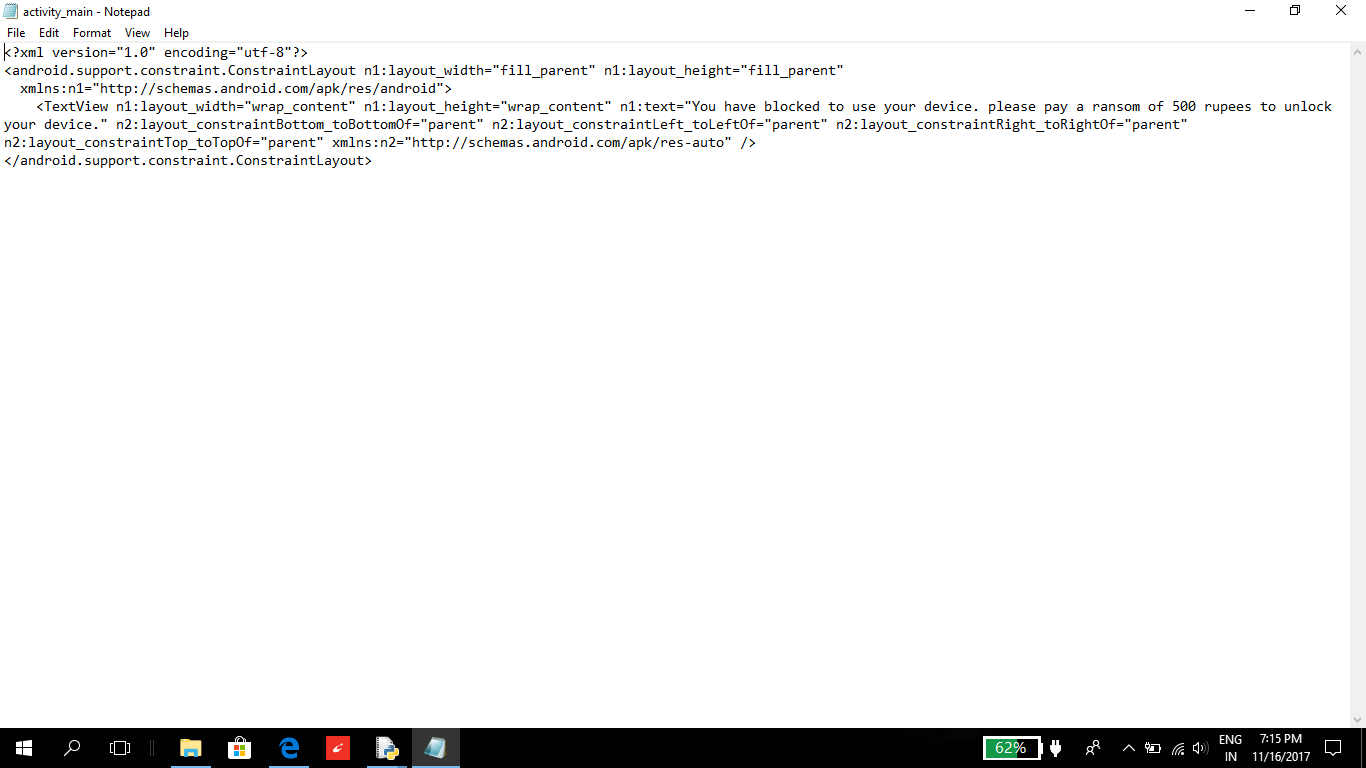
Enter the app name and it will show if it has threatening messages or not

***19***





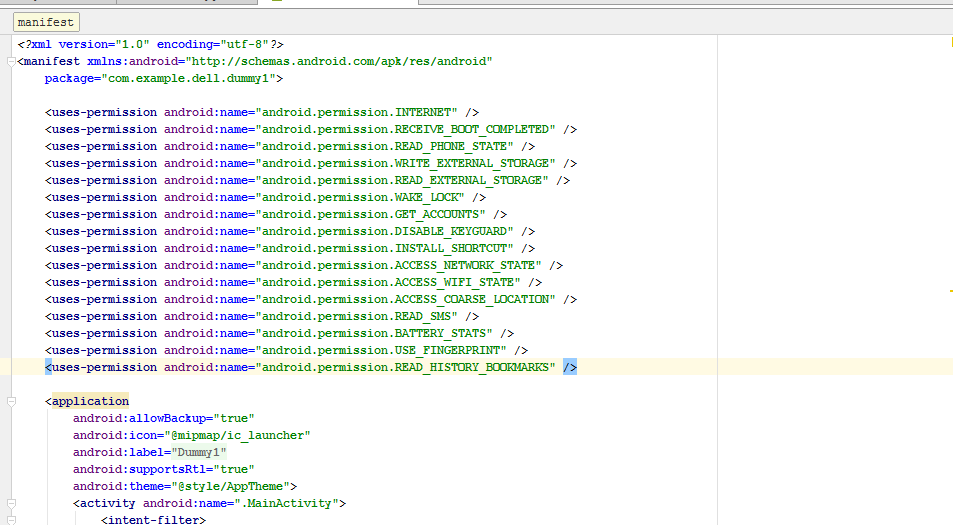
***20***



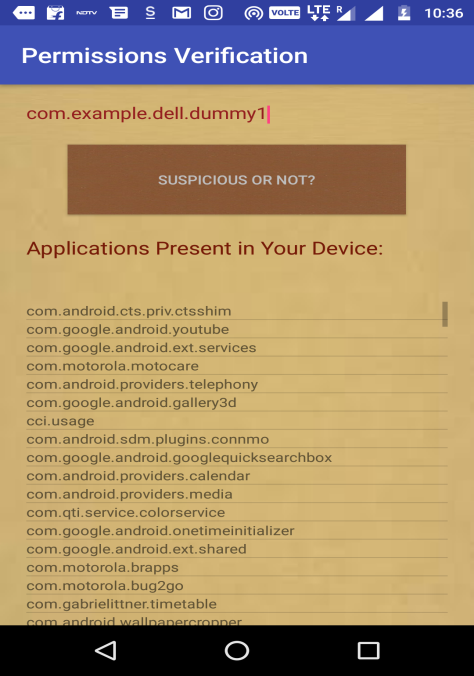
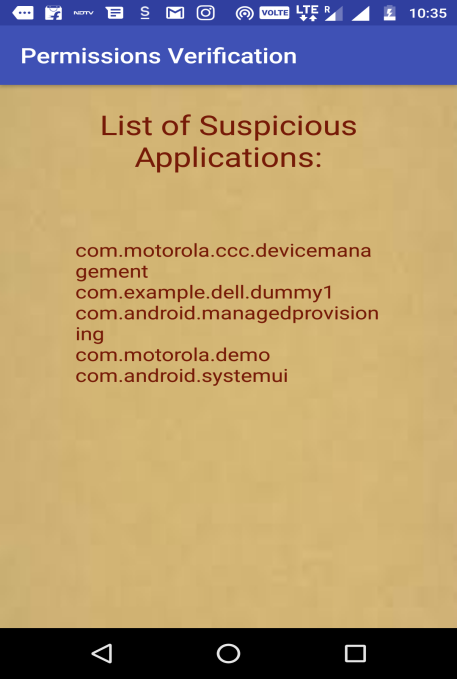
**Permission Verification Mobile Application:**

*Dummy App 1: 80% Suspicious Permissions*

*Manifest File:*

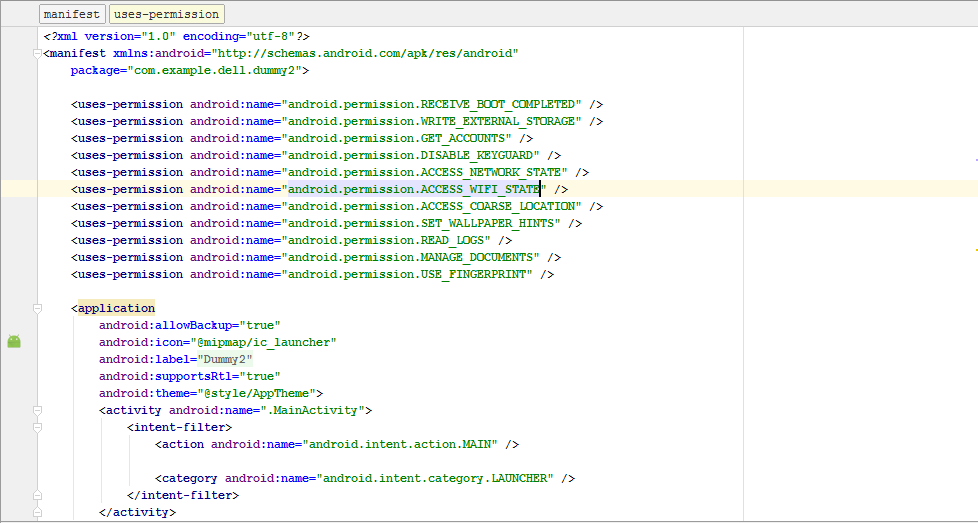
******

*Application layouts:*

**

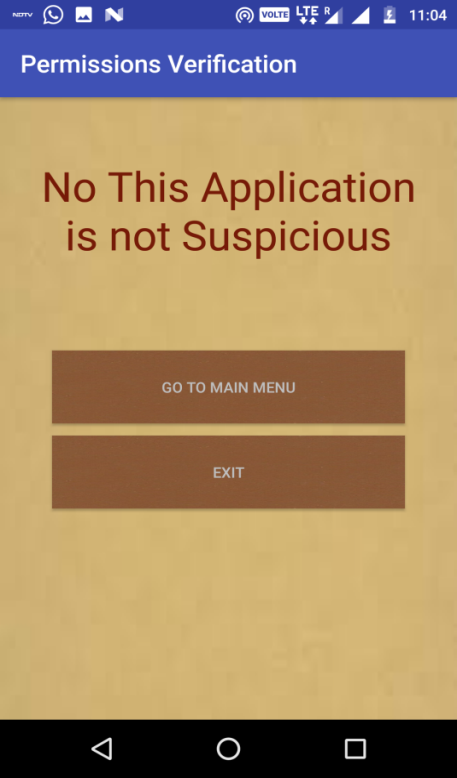
*Dummy App 2: Less than 80% Permissions*

*Manifest File:*

**

***22***

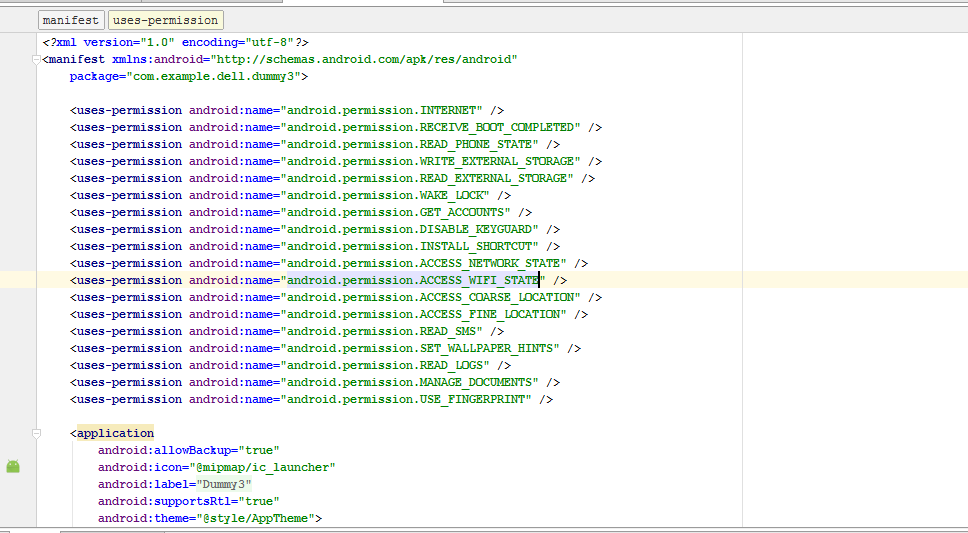
*Application Layouts:*

********

**A test case in which the dummy app contains both threatening text as well as suspicious permissions:**

*Dummy App 3:*

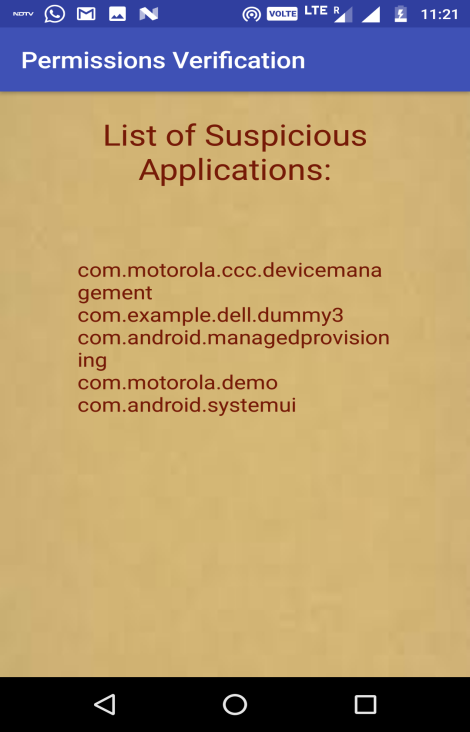
*Manifest file:*

******

*Activity containing threatening text:*

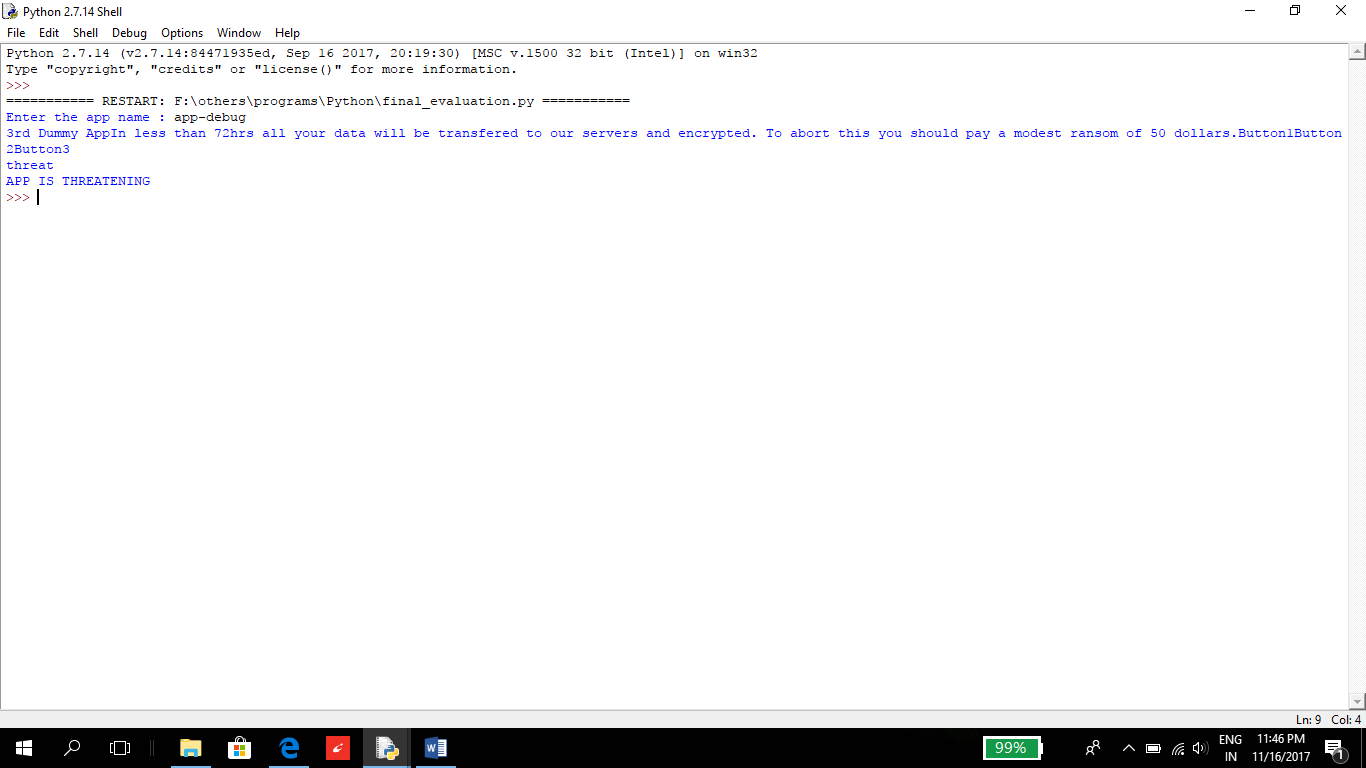
******

*Application layouts:*

******

*Text classification of threatening text:*

***24***

******

***25***

1. ***CONCLUSION AND FUTURE SCOPE***

The result obtained by our two models namely, Text Classifier and Permission Verification Tool, gave us the 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.

Since the android market is extremely vast and as mentioned above ransomware is an ever increasing and dangerous threat worldwide, there is an urgent need for a proper detection and preventing method for this problem and will be needed with more work explored in the long run.

This project has a great scope in future. We have considered 2 features of ransomware. As future work, our methodology can be extended to detect more features of ransomware and software can be developed which will detect he presence of ransomware in android devices with more accurate results.

***References***

**Journal References:**

[9] Tianda Yang, Yu Yang, Kai Qian, Dan Chia-Tien (Kennesaw State University , Marietta, GA, USA ) , Ying Qian (East China Normal University , Shanghai, China) , Lixin Tao (Pace University , New York, USA), “Automated Detection and Analysis for Android Ransomware”, *IEEE Transactions on Image Processing*, 2015

[10] Nicol´o Andronio, Stefano Zanero, and Federico Maggi (B), “HELDROID: Dissecting and Detecting Mobile Ransomware”, *IEEE Transactions on Image Processing*,Springer International Publishing Switzerland 2015

[11] Francesco Mercaldo (B), Vittoria Nardone, Antonella Santone, and Corrado Aaron Visaggio (Department of Engineering, University of Sannio, Benevento, Italy), “Ransomware Steals Your Phone. Formal Methods Rescue It”, *IEEE Transactions on Image Processing*, IFIP International Federation for Information Processing 2016

[12] Sanggeun Song, Bongjoon Kim, and Sangjun Lee from School of Computing, Soongsil University, Sangdo-ro, Dongjak-gu, Seoul06978, Republic of Korea, “Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform”, *IEEE Transactions on Image Processing*, 10 March, 2016

[13] Davide Maiorca ( Department of Electrical and Electronic Engineering University of Cagliari Cagliari, Italy), Francesco Mercaldo (Institute for Informatics and Telematics National Research Council Pisa, Italy) , Giorgio Giacinto (Department of Electrical and Electronic Engineering University of Cagliari Cagliari, Italy) , Corrado Aaron Visaggio (Department of Engineering University of Sannio Benevento, Italy), Fabio Martinelli (Institute for Informatics and Telematics National Research Council Pisa, Italy), ”R-PackDroid”, *IEEE Transactions on Image Processing*, April 03-07, 2017

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

[15] Chengyu Zheng(B), Nicola Dellarocca, Niccol`o Andronio, Stefano Zanero, and Federico Maggi. Place:-DEIB, Politecnico di Milano, Milan, Italy, ”GreatEatlon”, *IEEE Transactions on Image Processing*, ICST Institute for Computer Sciences, Social Informatics and Telecommunications Engineering 2017

**Web References:**

[1]<https://blog.barkly.com/ransomware-statistics-2017>

[2]<https://gbhackers.com/new-ransomware-attackandroidphoneswhich-lookslike-a-wannacry/>

[3]http://www.phonearena.com

[4]<http://searchsecurity.techtarget.com/definition/ransomware>/

[5]<https://nakedsecurity.sophos.com/2017/05/19/wannacry-how-safe-is-your-android-phone-from-this-ransomware/>

[6]<https://www.androidauthority.com/ransomware-attacks-android-increased-751266/>

[7]<https://cybersecurityventures.com/ransomware-damage-report-2017-5-billion/>

[8]<https://labs.mwrinfosecurity.com/assets/resourceFiles/mwri-behavioural-ransomware-detection-2017-04-5.pdf>

***28***