FINAL YEAR PROJECT

**MALWARE DETECTION USING HYBRID ANALYSIS**

horizontal line

# Guide : Dr. S. Valli, Professor

# Team Members

1.VENKATESH P 2015103051

2.HARIISH N 2015103592

3.NARAYANAN 2015103040

# 

# 

# 

# 

# 

# 

# 

# 

# 

# 

# PROBLEM STATEMENT

The sensitivity of mobile platforms and their potential for abuse, several security issues not too dissimilar to those already affecting traditional IT network counterparts are beginning to surface. The most significant issue is the emergence of traditional malware such as viruses, worms, Trojan horses, and rootkits. Malicious software in this context behaves similarly to the same threats on traditional IT networks. Mobile malware is the highest threat to the security of IoT data, user’s personal information, identity, and corporate/financial information. We considered static, dynamic, and hybrid detection analysis. In this performance analysis, we compared static, dynamic, and hybrid analyses on the basis of data set, feature extraction techniques, feature selection techniques, detection methods, and the accuracy achieved by these methods. Therefore, we identify suspicious API calls, system calls, and the permissions that are extracted and selected as features to detect mobile malware.

# 

# INTRODUCTION

The open source Android platform allows developers to take full advantage of the mobile operating system, but also raises significant issues related to malicious applications. On one hand, the popularity of Android absorbs attention of most developers for producing their applications on this platform. The increased numbers of applications, on the other hand, prepares a suitable prone for some users to develop different kinds of malware and insert them in Google Android market or other third party markets as safe applications. Malware has become more harmful than in the past as the number of intelligent systems and Internet-connected devices increased dramatically. Therefore one of the most important issues in cyber security has become the detection of previously unknown malware in the shortest time possible in order to stop it from becoming epidemic and from harming users.

The most significant issue is the emergence of traditional malware such as viruses, worms, Trojan horses, and rootkits. Malicious software in this context behaves similarly to the same threats on traditional IT networks. In this case malware may be targeted at ex-filtrating sensitive data from the mobile platform or further leveraging the compromised asset to access sensitive industrial/critical infrastructures in IoT networks in which the mobile device has legitimate access to the corresponding IoT devices. Second, developers are beginning to offer software for free and instead generate revenue with data analytics and marketing which can invade a user’s privacy.

Detection of malware is a crucial task, as malware developers hide their malicious activities and introduce new methods to avoid detection. Anti-malware software must cope with new technologies such as code obfuscation, mimicry attacks etc. For Windows and other operating systems, use of resources is not a critical concern, whereas for the operating system of mobile devices resource usage is always a concern. Limited use of resources means that the process of malware detection is not a straightforward problem for the mobile platform. Detecting the malicious activities of mobile applications using limited resources within a limited time period is a challenge to researchers.

A wide range of Android operating systems are used. Old versions of Android may suffer from different security issues. Additionally, many smartphone and tablet vendors run third party apps markets which may act as a source of malicious applications. People from China and Asia choose to download apps in their local languages which are available at third party apps store and are a potential source of malware. Crowdsourcing is used for the prevention of malware in Android. However, fake reviews from users can be a security threat. Joining a site such as Admob is comparatively easier than joining iAd as no identity proof is required. This encourages the ad-based malware developers. These security threats can cause issues like personal data leakage, social, business, financial loss etc.

Researchers have used static, dynamic, and hybrid processes to detect mobile malware and malicious activities. Researchers’ primary concerns involve accuracy levels, and most the research papers describe the performance of their detection process using accuracy metrics. For the operating system of mobile devices, performance overhead should be considered, as higher accuracy may cause higher overhead. Accuracy and performance overhead need to be well balanced to make the detection process efficient.

The static feature is formed by analyzing the structure and format of the sample and then extracting the hash value, string information, function information, header file information, and resource description information. The technology obtains most of the malware information from the malware itself, thus the analysis results are relatively comprehensive. However, static features cannot correctly discriminate malware when the static information is packed or obfuscated or compressed [7], making it difficult for static features to express the true purpose of malware, thus affecting the accuracy of detection.

Dynamic features are the behavior of the sample execution and the features of the debug record, such as file operations, the creation and deletion of processes, and other dynamic behaviors. Since the malicious behaviors of malware at dynamic runtime can’t be concealed, the extracted dynamic features provide a more realistic description than the static features. However, the extraction of dynamic features needs to be run in a virtual environment [8], which will be reset and restored to the previous state after each malicious sample has be analyzed to ensure that the virtual environment is a real user environment. As a result, features extraction efficiency is much lower than for static features.

We analyze the ongoing research efforts covering the three basic categories: static, dynamic, and hybrid analysis. These analyses represent the data set, features, feature selection method, detection method, and the accuracy. We also have mentioned the literature gap and the limitations of current research efforts. Thereby we have identified the suspicious feature lists which are commonly used by malware developers. The main contributions of this research effort are divided into the following main areas:

• Defining the mobile malware detection process for IoT networks.

• Determining the security limitations for mobile platforms in industrial IoT networks.

• A comparative analysis of static, dynamic, and hybrid detection processes and their limitations and scopes.

• Identifying the suspicious permission, API call, and system call lists to enable IoT application developers in the safe use of APIs.

We propose to combine permission and API (Application Program Interface) calls and use machine learning methods to detect malicious Android Apps. In our design, the permission is extracted from each App's profile information and the APIs are extracted from the packed App file by using packages and classes to represent API calls. By using permissions and API calls as features to characterize each Apps, we can learn a classifier to identify whether an App is potentially malicious or not.

**RELATED WORKS**

**1.Significant Permission Identification for Machine-Learning-Based Android Malware Detection**

This paper was published in IEEE July 2018 by J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an and H. Ye, vol. 14, no. 7, pp. 3216-3225. SVM and a small dataset to test our proposed MLDP model. The SVM determines a hyperplane that separates both classes with a maximal margin based on the training dataset that includes benign and malicious applications. In this case, one class is associated with malware, and the other class is associated with benign apps. Then, we assume the testing data as unknown apps, which are classified by mapping the data to the vector space to decide whether it is on the malicious or benign side of the hyperplane. Then, we can compare all analysis results with their original records to evaluate the malware detection correctness of the proposed model by using the SVM. The problems that were encountered in this methodology was that the algorithm uses only static features, So the efficiency is based on these features alone. It does predict the malware with much accuracy when dynamic features are considered along with static features. The ideas that were adopted from this paper include static features used for malware detection and efficient feature selection as no app requests all the permissions, and the ones that an app requests are listed in the Android application package (APK) as part of *manifest.xml.* Three levels of data pruning methods were proposed to filter out permissions that contribute little to the malware detection effectiveness. Thus, they can be safely removed without negatively affecting malware detection accuracy. The results indicate that when a support vector machine is used as the classifier, we can achieve over 90% of precision, recall, accuracy, and F-measure, which are about the same as those produced by the baseline approach while incurring the analysis times that are 4–32 times less than those of using all permissions.

**2. Malware Threats and Detection for Industrial Mobile-IoT Networks**

This paper was published in IEEE March 2018 by S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail and M. M. Hassan. This paper analyzes the efforts regarding malware threats aimed at the devices deployed in industrial mobile-IoT networks and related detection techniques. We considered static, dynamic, and hybrid detection analysis. In this performance analysis, we compared static, dynamic, and hybrid analyses on the basis of data set, feature extraction techniques, feature selection techniques, detection methods, and the accuracy achieved by these methods. Therefore, we identify suspicious API calls, system calls, and the permissions that are extracted and selected as features to detect mobile malware. This will assist application developers in the safe use of APIs when developing applications for industrial IoT networks. The detection method using K-nearest Neighbor (KNN) achieves best performance in accuracy. Random Forest (RF) and Support Vector Machine (SVM) are mostly used as detection methods and also exhibit high accuracy. The problems that were encountered in this methodology was that it fails to detect unknown/new variant of malware. There is also a need for selection of proper feature set to incorporate unknown behaviours. It also fails to predict the behaviour of malicious apps from their past actions. The ideas that were adopted from this paper include comparing and contrasting static, dynamic, and hybrid analyses on the basis of data set, feature extraction techniques, feature selection techniques, detection methods, and the accuracy achieved by these methods. Therefore, we identify suspicious API calls, system calls, and the permissions that are extracted and selected as features to detect mobile malware. This will assist application developers in the safe use of APIs when developing applications for industrial IoT networks.

**3.An Android malware detection method based on AndroidManifest file**

This paper was published in IEEE August 2016 by X. Li, J. Liu, Y. Huo, R. Zhang and Y. Yao. This paper found that the statistical information of Android components (mainly activity) from the Manifest file cannot be ignored, based on the traditional method of Android permission detection. In this paper, a new feature vector is extracted from the AndroidManifest file, which combines the permission information and the component information of the Android application. We combine the naive Bias classification algorithm, and propose a malicious application detection method based on AndroidManifest file information. The experimental results show that the new method performance better than that of the traditional permission detection. The problems that were encountered in this methodology was that the algorithm uses only static features, So the efficiency is based on these features alone. It does predict the malware with much accuracy when dynamic features are considered along with static features. The ideas that were adopted from this paper include the malware detection method based on AndroidManifest file. Sample statistical analysis results showed that the component information in the AndroidManifest file shows different statistical distribution patterns in different kinds of applications. The experimental results show that the new method performance better than that of the traditional permission detection.

**4. Fest: A feature extraction and selection tool for Android malware detection**

This paper was published in IEEE February 2016 by K. Zhao, D. Zhang, X. Su and W. Li. Feature Extraction and Selection Tool (Fest), a feature-based machine learning approach for malware detection. We first implement a feature extraction tool, AppExtractor, which is designed to extract features, such as permissions or APIs, according to the predefined rules. Then we propose a feature selection algorithm, FrequenSel. Unlike existing selection algorithms which pick features by calculating their importance, FrequenSel selects features by finding the difference their frequencies between malware and benign apps, because features which are frequently used in malware and rarely used in benign apps are more important to distinguish malware from benign apps. In experiments, we evaluate our approach with 7972 apps, and the results show that Fest gets nearly 98% accuracy and recall, with only 2% false alarms. Moreover, Fest only takes 6.5s to analyze an app on a common PC, which is very time-efficient for malware detection in Android markets. The problems that were encountered in this methodology was that the works are inefficient due to lack of feature selection, which also results in the imbalance between accuracy and recall, and time overhead in building classifiers. The ideas that were adopted from this paper include extracting all the features which indicate the functions and behaviors in apps by *AppExtractor*, and an algorithm *FrequenSel* to select typical features which help distinguish malware from benign apps. *FrequenSel* selects features by finding the difference their frequencies between malware and benign apps, because features which are frequently used in malware and rarely used in benign apps are more important to distinguish malware from benign apps.

**5.Permission-Based Android Malware Detection**

This paper was published in IEEE February 2016 by Aung, Zarni & Zaw, Win. Android-based smartphone users can get free applications from Android Application Market. But, these applications were not certified by legitimate organizations and they may contain malware applications that can steal privacy information for users. In this paper, a framework that can detect android malware applications is proposed to help organizing Android Market. The proposed framework intends to develop a machine learning-based malware detection system on Android to detect malware applications and to enhance security and privacy of smartphone users. This system monitors various permission based features and events obtained from the android applications, and analyses these features by using machine learning classifiers to classify whether the application is goodware or malware. The performances of machine learning techniques were evaluated using the true positive rate, false positive rate and overall accuracy. The problems that were encountered in this methodology was that the algorithm uses only static features, So the efficiency is based on these features alone. It does predict the malware with much accuracy when dynamic features are considered along with static features. The ideas that were adopted from this paper include static features used for malware detection and efficient feature selection as no app requests all the permissions, and the ones that an app requests are listed in the Android application package (APK) as part of *manifest.xml*

**6. Machine Learning for Android Malware Detection Using Permission and API Calls**

This paper was published in IEEE February 2016 by N. Peiravian and X. Zhu, the permission is extracted from each App's profile information and the APIs are extracted from the packed App file by using packages and classes to represent API calls. By using permissions and API calls as features to characterize each Apps, we can learn a classifier to identify whether an App is potentially malicious or not. An inherent advantage of our method is that it does not need to involve any dynamical tracing of the system calls but only uses simple static analysis to find system functions involved in each App. In addition, because permission settings and APIs are always available for each App, our method can be generalized to all mobile applications. Experiments on real-world Apps with more than 1200 malware and 1200 benign samples validate the algorithm performance. The problems that were encountered in this methodology was that it fails to detect unknown/new variant of malware. There is also a need for selection of proper feature set to incorporate unknown behaviours. It also fails to predict the behaviour of malicious apps from their past actions. The ideas that were adopted from this paper include the permission extracted from each App’s profile information and the APIs are extracted from the packed App file by using packages and classes to represent API calls. By using permissions and API calls as features to characterize each Apps.The proposed framework extracts permissions from Android applications and further combines the API calls to characterize each application as a high dimension feature vector.

**ARCHITECTURE DIAGRAM**

## 

## 

## 

## 

## 

## 

## DETAILS OF MODULES

**1.Input( APK Dataset)**

The first module of the malware detection system is the data set. The data set represents a representative collection of malware and benign ware. A proper data set is essential for analysis of the behavior of the malware. Examples of both the malware and the benign apps for proper detection are taken under consideration.. The most common sources of Android Mobile Malware are the Genome project, Contagio, DREBIN data set, Virus Share. The most widely used sources for known benign apps are Google Play, App China, Amazon, Android Market. Contagio360.zip and Malware1260.zip were some of the datasets used.

**2. Preprocessing of data**

**2.1 Feature Extraction**

**INPUT:** List of APK files (Malware and Benignware)

The feature Extractor is the component which extracts the desired features from the malware and benign apps. The mechanism that extracts the static features is called a Static Feature Extractor.Static features can be extracted from the manifest file, dex file, and byte code. The most widely used tool is the APK tool. From the APK tool, we obtain the APK file, Manifest file, classes.dex, and Smali file. We perform reverse engineering on the dataset followed by parsing to extract the static features. The mechanism that extracts the dynamic features is called the Dynamic Feature Extractor. The log file is generated by executing the application in a virtual machine. From the data set, we collect the malware and benign ware and execute these in the virtual machine, which generates the run time log file. From the log file, we can extract the dynamic features. We now combine the static and dynamic features to obtain the combined feature set which is further used for feature selection.

**OUTPUT**: Extracted Features

Static Feature Extraction :

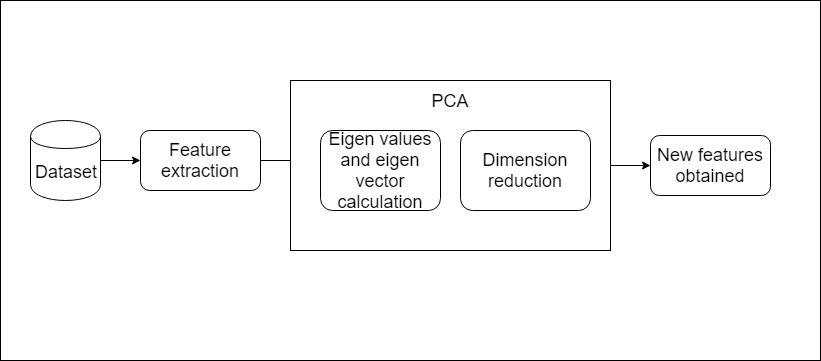
Dynamic Feature Extraction :

**2.2 Feature Selection**

**INPUT:** Extracted Features

There are many features, but we need to select those among them which will provide better accuracy in the classification process. We use the principal component analysis algorithm here to select the features that are needed for the detection of malware. The first step in this algorithm involves feature selection .In principal component analysis, this relationship is quantified by finding a list of the principal axes in the data, and using those axes to describe the dataset. Using Scikit-Learn's PCA estimator,The fit learns some quantities from the data, most importantly the "components" and "explained variance".To see what these numbers mean, let's visualize them as vectors over the input data, using the "components" to define the direction of the vector, and the "explained variance" to define the squared-length of the vector.Using PCA for dimensionality reduction involves zeroing out one or more of the smallest principal components, resulting in a lower-dimensional projection of the data that preserves the maximal data variance. We then perform feature evaluation to evaluate the significance of these ranked features

**OUTPUT** : Selected Features



**3 Machine learning and Malware detection**

**INPUT:** Selected Features

**3.1 Building the model**

The detection method or the classifier is used to determine whether an app is malware or not. Based on the features, the classifier classifies apps as malware or benign ware. Most classifiers use machine learning. Classifiers based on machine learning use one or multiple classifiers. Layered classifiers may also be used in the detection process. We now build the model using KNN,SVM etc. classification algorithms.

**3.2 Testing**

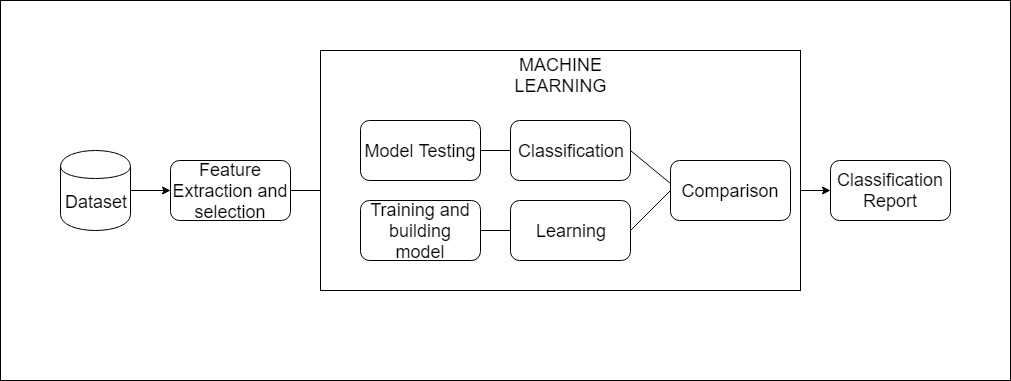
We first split the dataset into the train dataset and test dataset. We use the train dataset to build the model. Now after building the model we use the test dataset which will be upto 20% of the dataset to see if the model is trained properly and providing almost accurate results.

**3.3 Classification report**

We first calculate the performance metrics of our model. The accuracy produced by the model is therefore recorded in the report. After which we perform a comparative analysis has been carried out based on static, dynamic, and hybrid analysis for the Android mobile malware detection process. We evaluate them considering the number of used malware, feature extraction process, selected features, detection method, and accuracy.

## OUTPUT: Classified applications

**BLOCK DIAGRAM:**



## 

## 

## 

## 

## 

## 

## 

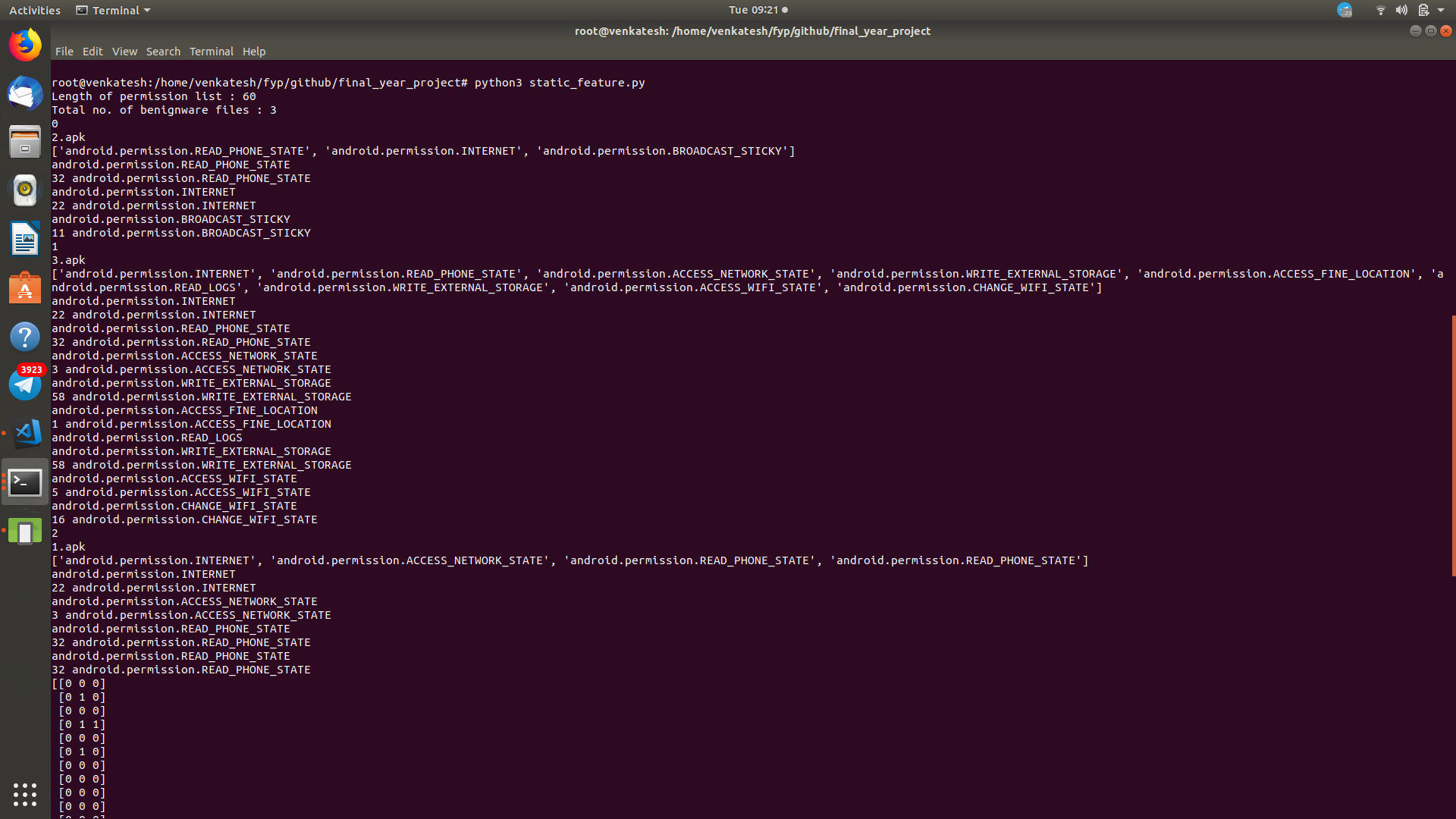
## 

## 

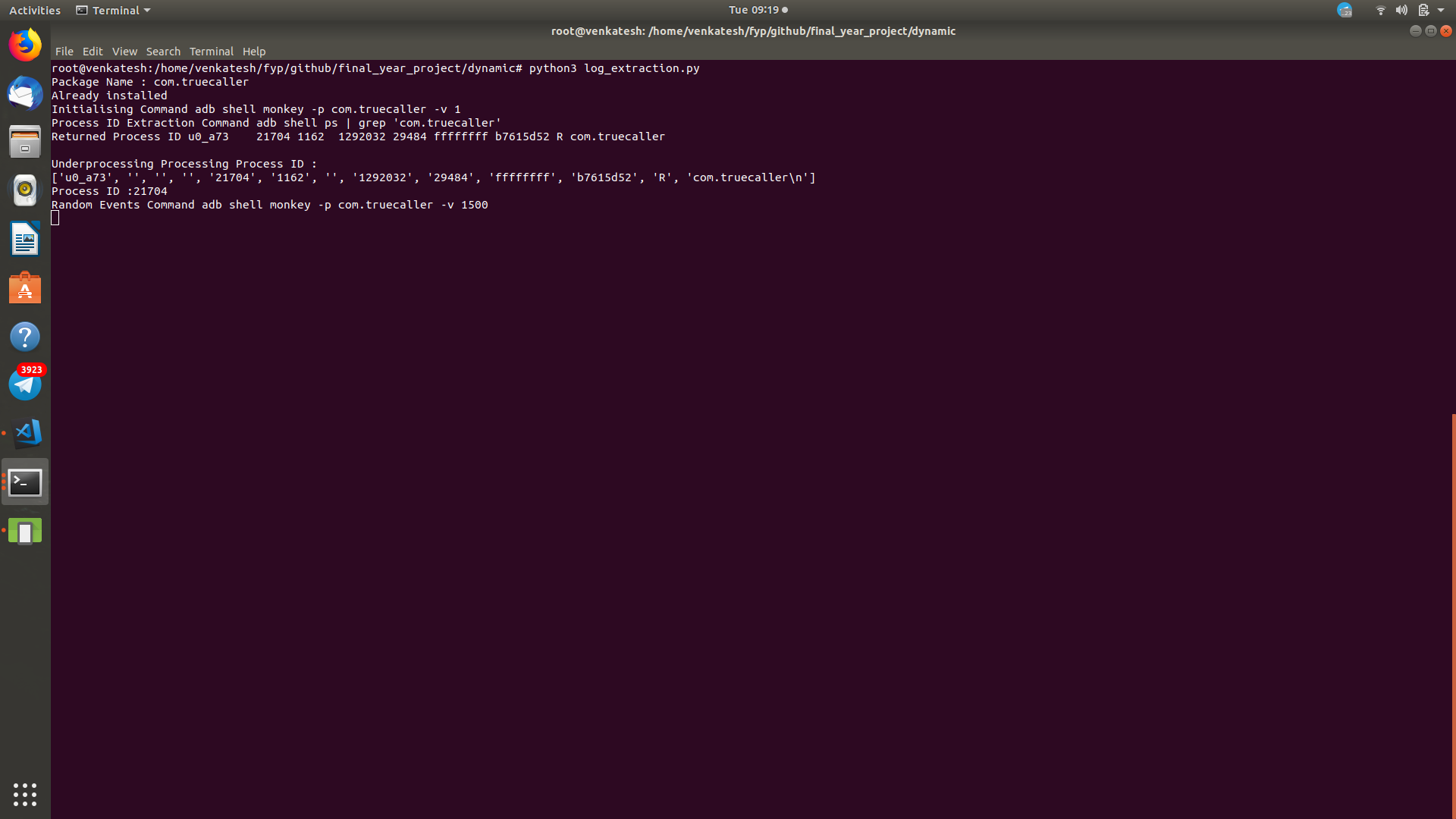
## 

## 

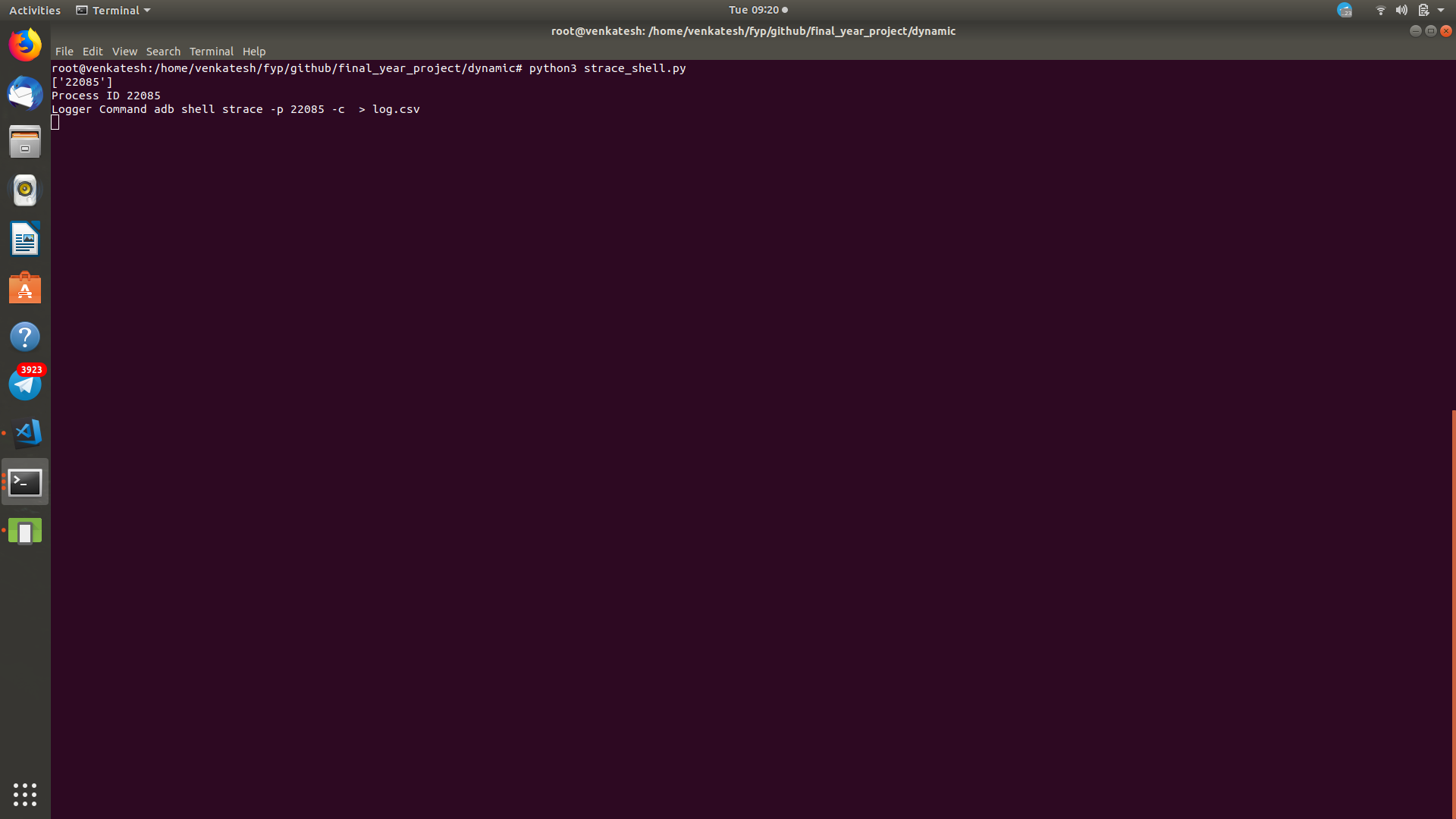
## IMPLEMENTATION AND SCREENSHOTS



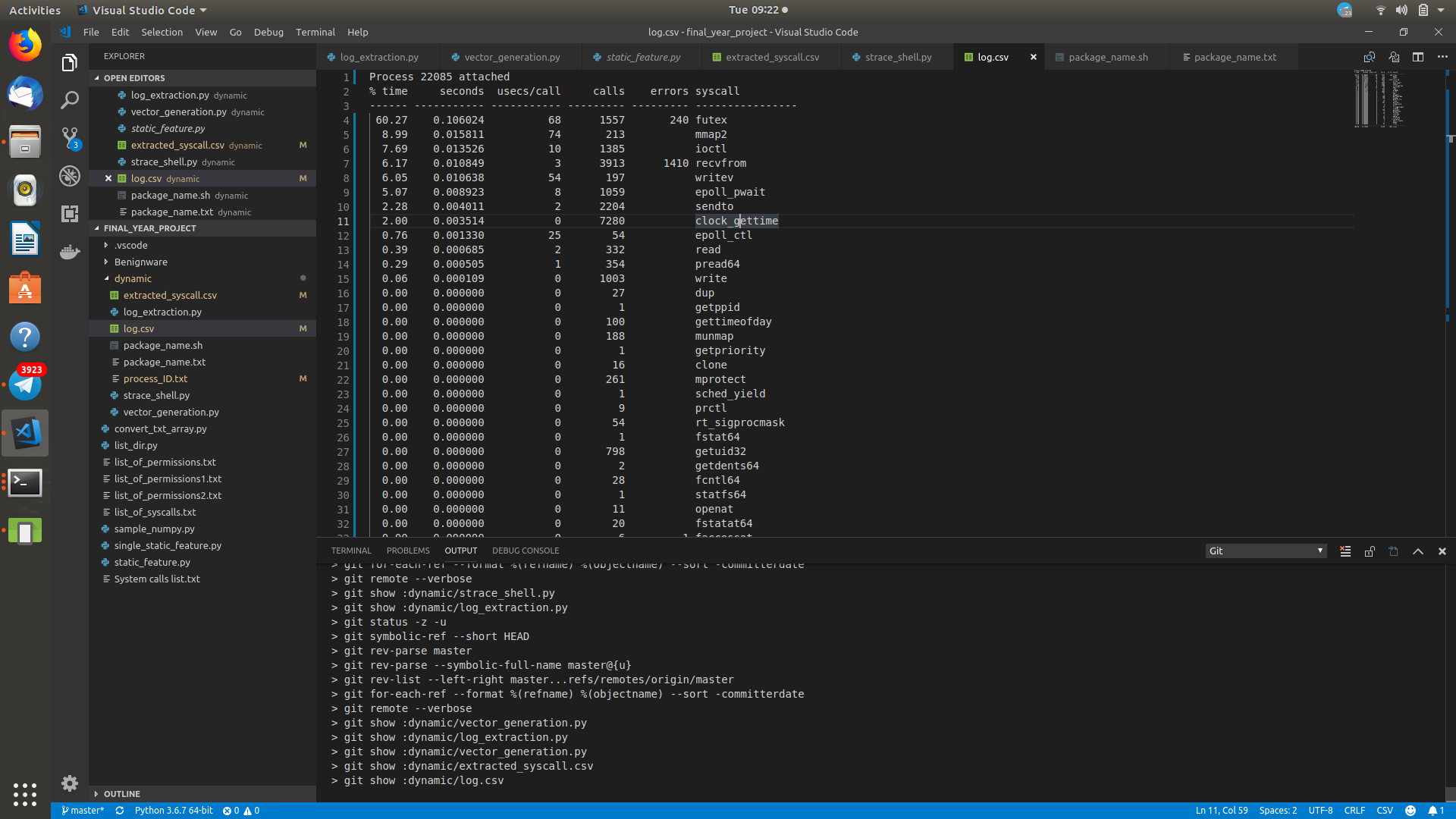
Static Feature Extraction



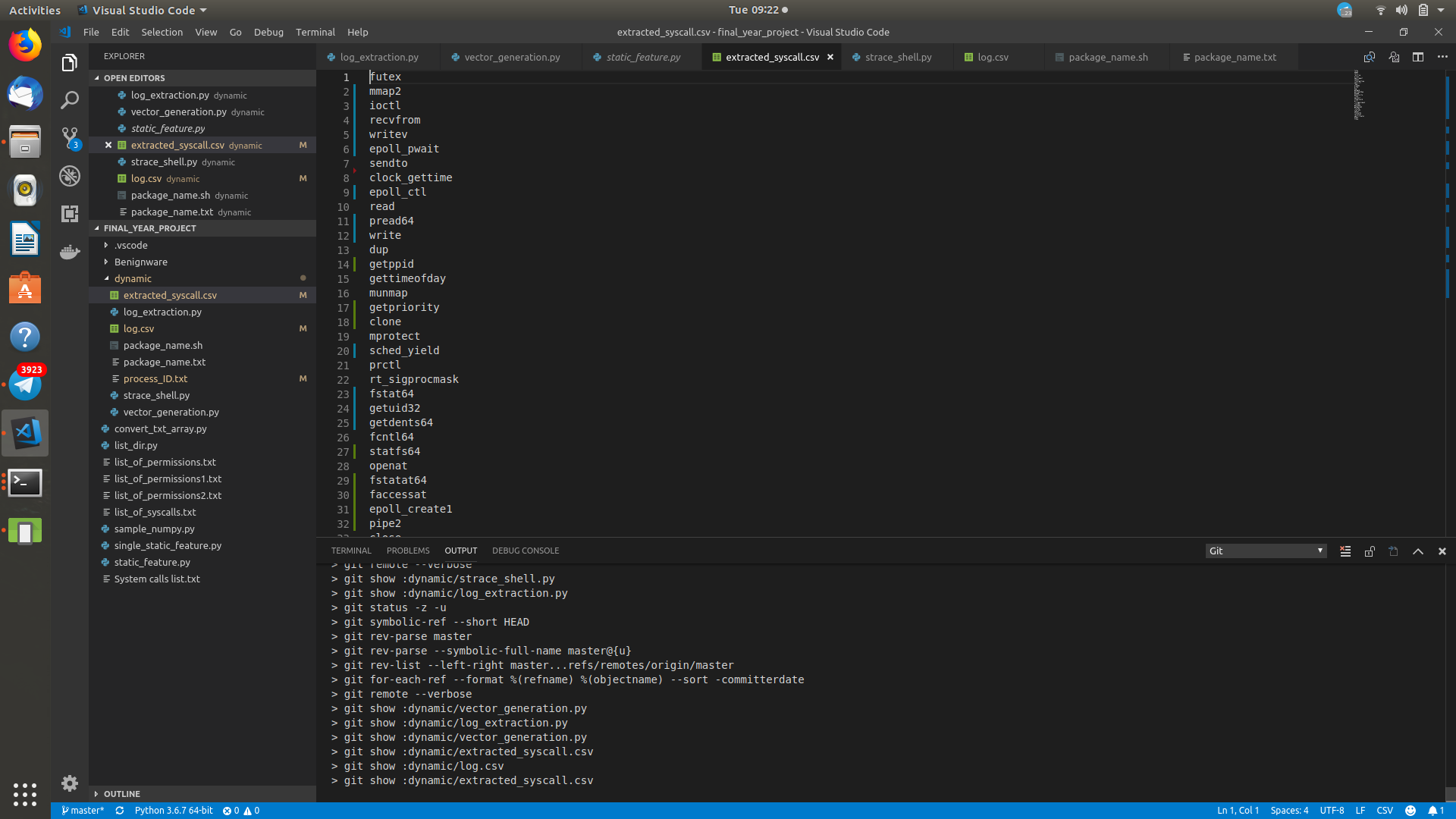
Dynamic Feature Extraction (Generating Events)



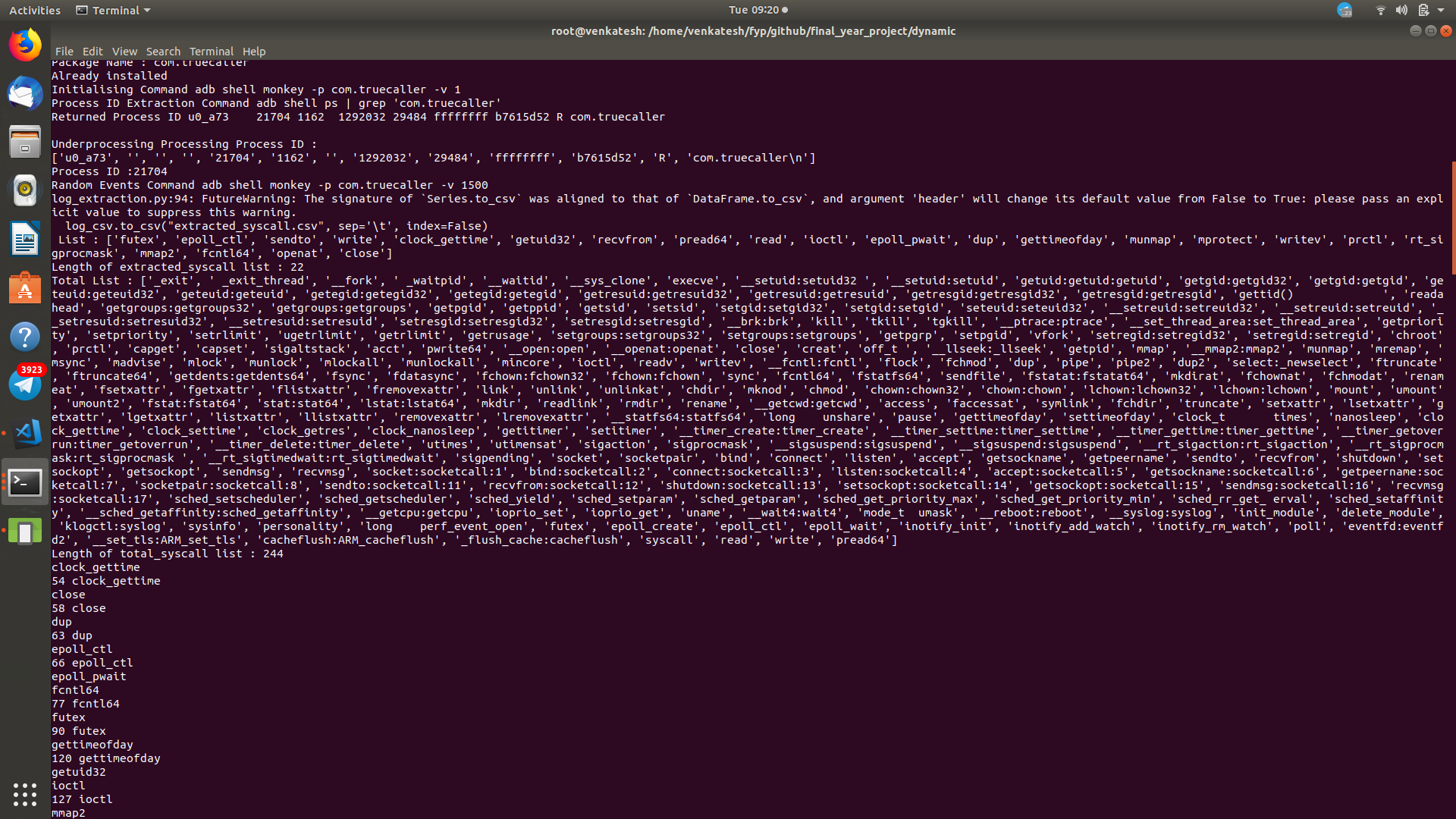
Dynamic Feature Extraction (Extracting log)



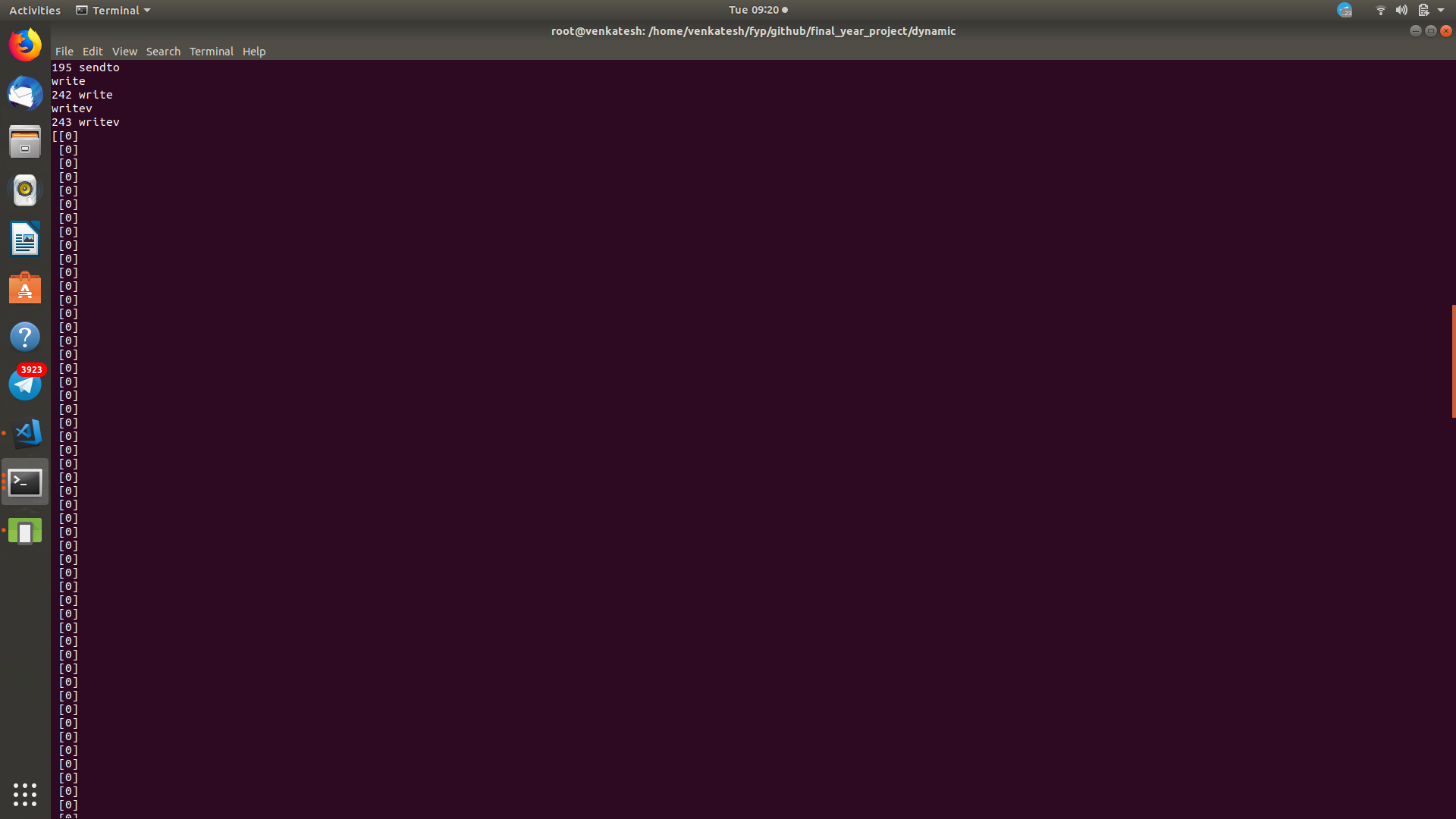
Dynamic Feature Extraction (log File)



Dynamic Feature Extraction (Extracted system call)



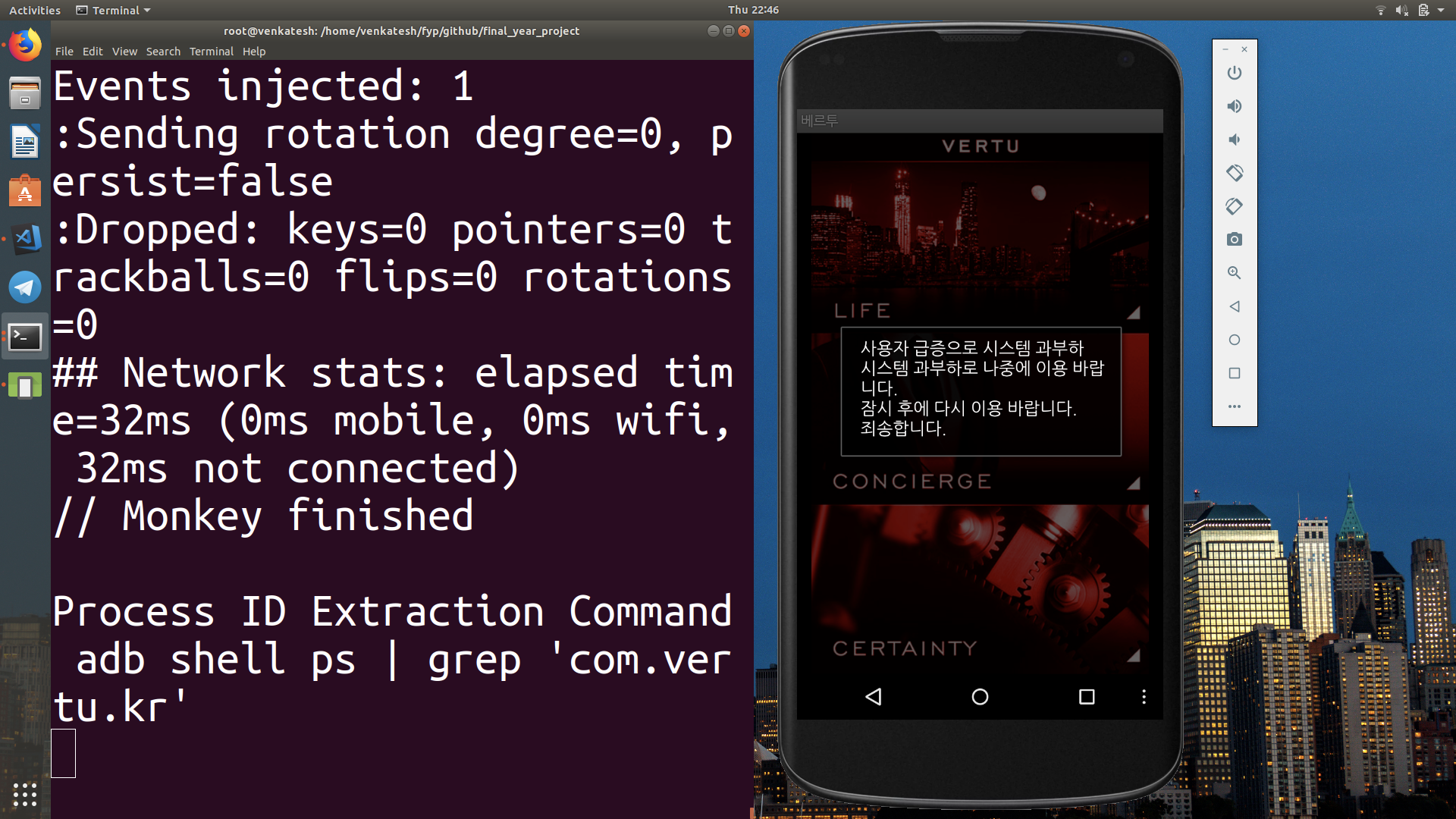
Dynamic Feature Extraction (Generating vector)



Dynamic Feature Extraction (Generated vector)

## 

## 

Random Event Generation

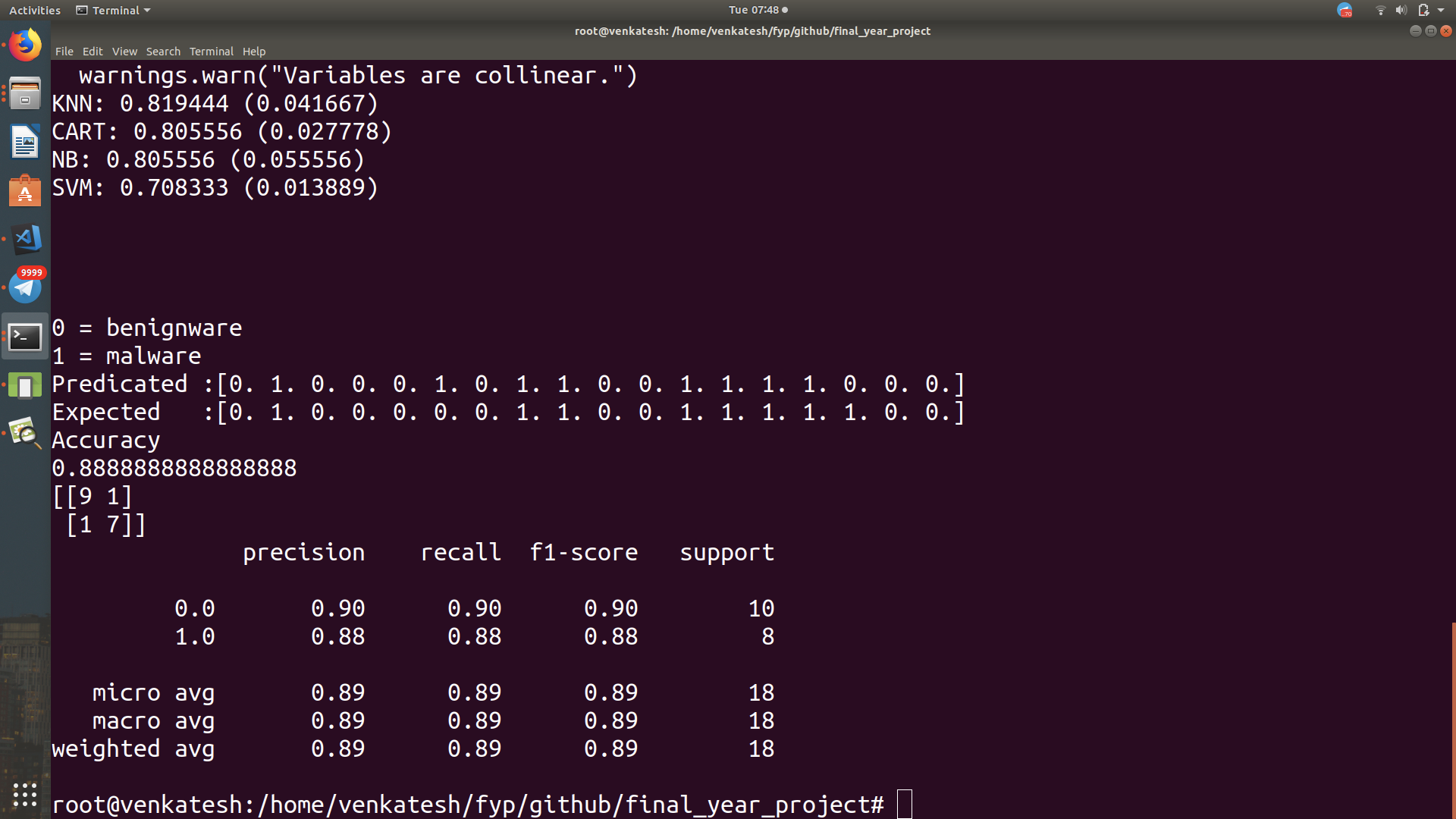
## 

## 

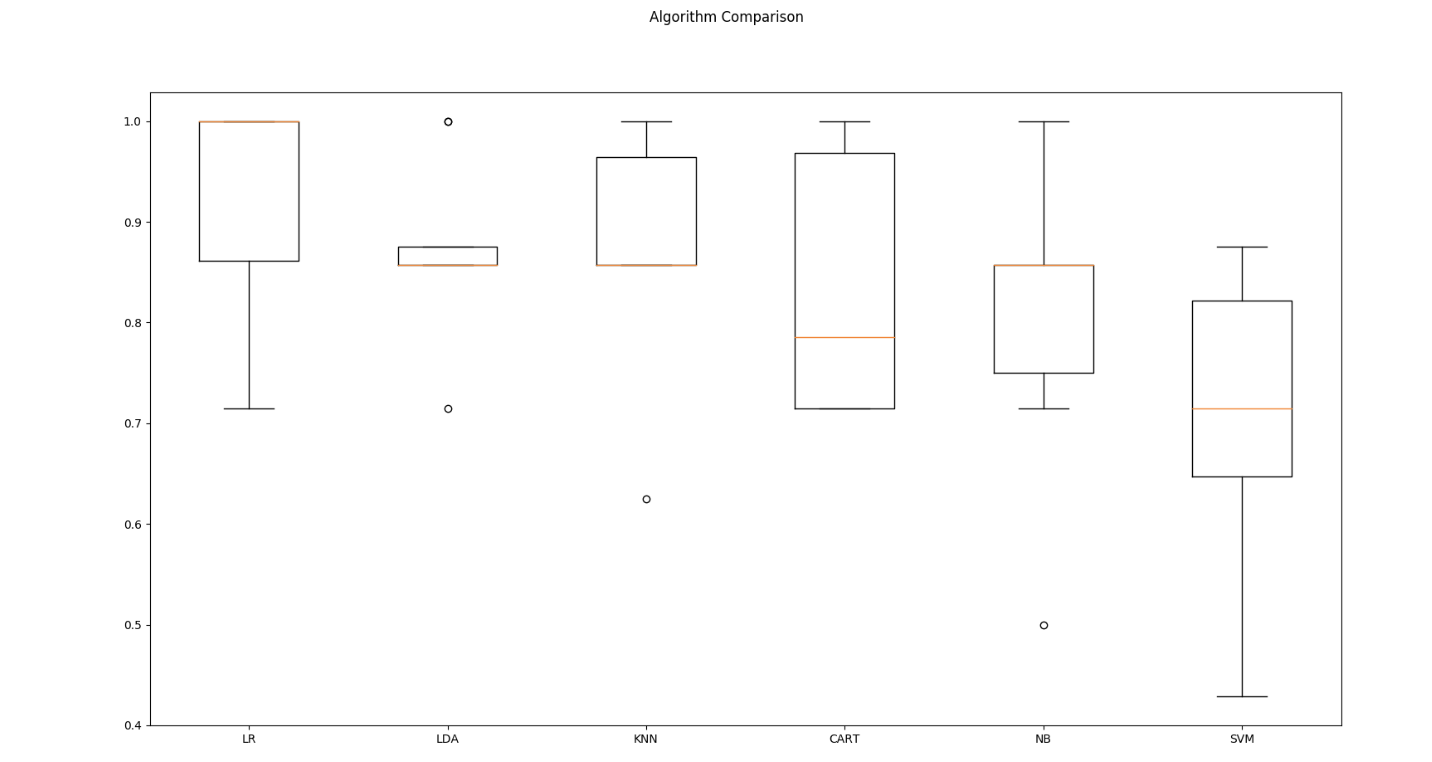
KNN train test set

## 

Train test set



Test cases



Comparison between algorithms

## 

Error when activities are not defined in application

## 

## 

## 

## 

## 

## PERFORMANCE METRICS

The True Positive Rate (TPR) defines the percentage of benign apps identified accurately, where

TPR=TP/(TP+FN)

TP is the number of accurately identified benign apps and FN is the number of incorrectly identified benign apps.

The False Positive Rate (FPR) defines the percentage of incorrectly identified malware apps, where

FPR=FP/(TN+FP).

FP is the number of incorrectly identified malware and TN is the number of correctly identified malware.

Accuracy is a metric used to describe the overall performance. Accuracy is the percentage of correctly identified apps, where

ACC=(TP+TN)/(TP+TN+FP+FN).

## 

## DATASET

Required Dataset : Malware application and Benign Application

Dataset will be collected from the following repositories.

1. The Drebin Dataset
2. Android Malware Genome Project
3. The Droidcat Dataset
4. AndrooZoo
5. AMD (Android Malware Dataset)

## 

## 

## 

## 

## 

## 

## 

## 

## 

## REFERENCES

[1] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an and H. Ye, "Significant Permission Identification for Machine-Learning-Based Android Malware Detection," in *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216-3225, July 2018.

[2] S. Sharmeen, S. Huda, J. H. Abawajy, W. N. Ismail and M. M. Hassan, "Malware Threats and Detection for Industrial Mobile-IoT Networks," in *IEEE Access*, vol. 6, pp. 15941-15957, 2018.

[3] X. Li, J. Liu, Y. Huo, R. Zhang and Y. Yao, "An Android malware detection method based on AndroidManifest file," *2016 4th International Conference on Cloud Computing and Intelligence Systems (CCIS)*, Beijing, 2016, pp. 239-243.

[4] K. Zhao, D. Zhang, X. Su and W. Li, "Fest: A feature extraction and selection tool for Android malware detection," *2015 IEEE Symposium on Computers and Communication (ISCC)*, Larnaca, 2015, pp. 714-720.

[5] Aung, Zarni & Zaw, Win. (2013). Permission-Based Android Malware Detection. International Journal of Scientific and Technology Research. 2. 228-234.

[6] N. Peiravian and X. Zhu, "Machine Learning for Android Malware Detection Using Permission and API Calls," 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, 2013, pp. 300-305.

[7] I. Bulut and A. G. Yavuz, "Mobile malware detection using deep neural network," 2017 25th Signal Processing and Communications Applications Conference (SIU), Antalya, 2017, pp. 1-4.