**ग्रीष्म/औद्योगिक परियोजना प्रशिक्षण प्रतिवेदन**

**Summer/Industrial Project Training Report**

On

**Title**

**Malware Detection based on API calls using WEKA**



जून-जुलाई 2017

**June - July, 2017**

(Two Months)

**प्रस्तुत द्वारा /Submitted By** -

कुमारी / Miss : Sneha Chatterjee

पंजी क्रमाक/ Roll No. : R-134214068

पाठ्यक्रम /Course : B. Tech CSE – Cyber Security and Forensics

विश्वविद्यालय**/**University : University of Petroleum and Energy Studies, Dehradun

**पर्यवेक्षक के अधीन/Under the supervision of** -

नाम/ Name : Sh. Yogesh Chandra

पद / Designation : Scientist ‘F’

कार्यालय/ Office  : ईसा / ISSA

**पद्धति अध्ययन एवं विश्लेषण संस्थान**

**Institute for Systems Studies and Analysis**

**रक्षा अनुसंधान एवं विकास संगठन**

**Defence R& D Organisation**

**रक्षा मंत्रालय/Ministry of Defence**

**मेटकाफ भवन, दिल्ली / Metcalfe House, Delhi-110054**

**अभ्यर्थी द्वारा घोषणा**

**DECLARATION BY THE CANDIDATE**

I hereby declare that the work which is being presented by me in this project/study entitled “***Malware Detection based on API calls using WEKA***” is an authentic record of my own work carried out during the period from 22nd May 2017 to 24th July 2017 under the supervision of Sh. Yogesh Chandra, Scientist ‘F’, Institute of Systems Studies and Analysis, Defence R&D Organisation, Ministry of Defence, Metcalfe House, Delhi 110054.

**द्वारा/ By:**

कुमारी / Miss : Sneha Chatterjee

पंजी क्रमाक/ Roll No. : R-134214068

पाठ्यक्रम /Course : B. Tech CSE – Cyber Security and Forensics

विश्वविद्यालय**/**University : University of Petroleum and Energy Studies, Dehradun

तारीख/Date : 24th **जुलाई** 2012/ 24th July 2017

**ACKNOWLEDGEMENT**

I am grateful to Director, ISSA and Head of HRD for providing me the opportunity to carry out my project at this esteemed organization. I wish to express my deep gratitude to Sh. Yogesh Chandra Scientist ‘F’, and Sh. Pallaw Kumar Mishra, Scientist ‘E’, ISSA, DRDO for providing guidance and support in the project work. I would also like to thank all other scientists and the technical staff and my fellow trainees. They were always there at the hour of need and provided with all the help and facilities, which I required for the completion of my project.

**द्वारा/ By:**

कुमारी / Miss : Sneha Chatterjee

पंजी क्रमाक/ Roll No. : R-134214068

पाठ्यक्रम /Course : B. Tech CSE – Cyber Security and Forensics

विश्वविद्यालय**/**University : University of Petroleum and Energy Studies, Dehradun

तारीख/Date : 24th **जुलाई** 2012/ 24th July 2017

**पद्धति अध्ययन एवं विश्लेषण संस्थान** के बारे में

# About the Institute for System Studies and Analysis (ISSA)

Institute for System Studies and Analysis (ISSA) is a premier institution involved in systems analysis of Defence Systems. It provides analysis support to the top echelons of the three services, SA to RM, and DRDO HQs for scientific decision making. It also provides systems analysis support to sister labs and other institutions under the Ministry of Defence. ISSA is primarily devoted to systems analysis and specializes in modeling and simulation in wide range of applications.

ISSA adopts state-of-the-art info-technologies such as Computer Networking, Software Engineering, Distributed Database, Distributed Simulation, Web Technologies, Situational Awareness and Soft-Computing Techniques in development of complex simulation products.

**रक्षा अनुसंधान एवं विकास संगठन के बारे में**

**About the Defence Research and Development Organization**

DRDO was formed in 1958 from the amalgamation of the then already functioning Technical Development Establishment (TDEs) of the Indian Army and the Directorate of Technical Development & Production (DTDP) with the Defence Science Organization (DSO). Today, DRDO is having more than 50 labs, engaged in developing Defence Technologies covering various disciplines like aeronautics, armaments, electronics, combat vehicles, engineering systems, instrumentation, missiles, advanced computing and simulation, special materials, naval systems, life sciences, training, information systems and agriculture. DRDO is backed by over 5000 scientists and about 25,000 other scientific, technical and supporting personnel.

**Vision**  
Make India prosperous by establishing world-class science and technology base and provide our Defence Services decisive edge by equipping them with internationally competitive systems and solutions.

**Mission**

* Design, develop and lead to production state-of-the-art sensors, weapon systems, platforms and allied equipment for our Defence Services.
* Provide technological solutions to the Defence Services to optimise combat effectiveness and to promote well-being of the troops.
* Develop infrastructure and committed quality manpower and build strong technology base.

**Core Competence**

* Deptt of Defence Research and Development (R&D) is working for indigenous development of weapons, sensors & platforms required by the three wings of the Armed Forces. To fulfill this mandate, Deptt of Defence Research and Development (R&D), is closely working with academic institutions, Research and Development (R&D) Centres and production agencies of Science and Technology (S&T) Ministries/Deptts in Public & Civil Sector including Defence Public Sector Undertakings & Ordnance Factories.

**Contents**

**Chapter 1: Abstract** **8**

**Chapter 2: Introduction 9-11**

2.1 General Introduction 9

* 1. Introduction to the project 10

**Chapter 3: Objective 12**

**Chapter 4: Literature Review 13-14**

4.1 Related Work 13

4.2 Research Conclusion 14

**Chapter 5: Implementation 15-20**

5.1 Overview 15

5.2 Installing WEKA 15

5.3 Loading the Training Dataset 17

* 1. Analyzing the Test Dataset 19

**Chapter 6: Result and Conclusion 21-22**

6.1 Result 21

* 1. Conclusion 22

**Chapter 7: Future Work 23**

**Chapter 8: Appendix 24**

**Chapter 8: References and Bibliography** **25**

**1. Abstract**

Malware is a generic term to denote all kinds of unwanted software (e.g., viruses, worms, or Trojan horses). Such software poses a major security threat to computer users. Unfortunately, the problem of malicious code is likely to grow in the future as malware writing is quickly turning into a profitable business. This clearly reveals the need for tools that detects the malware so that they can be quarantined. My intent behind taking up this project was to make use of the WEKA toolkit for malware (malicious programs) detection from the system/API calls and thereby propose a framework in addition to the traditional signature based detection methods. This tool can work as a defence mechanism against malwares spreading from one computer to another. However, the project has been carried out offline to see the effectiveness of the toolkit in identifying malwares within the stored data.

WEKA is considered as one of the most effective workbench for classification of data into distinct identifiable groups and therefore it has been developed to make it capable of studying the characteristics and patterns of various distinct groups by using a number of classification algorithms. This data mining framework has therefore been used to detect malicious programs on the basis of API calls. Several thousand malicious and clean program API calls were collected and analyzed to find out the best features and build models that can classify a given program into a malware or a benign class. In order to achieve the result, different classification algorithms have been used on the dataset and compared on the basis of various measures such as, F-measure, ROC area and accuracy to find out the most efficient algorithms for this particular experiment.

**Keywords:** *malware, benign, WEKA, API calls, F-measure, ROC area, Accuracy*

**2. Introduction**

**2.1 General Introduction**

We are overwhelmed with data. The amount of data that is continuously being created in the world and also in our lives seems ever-increasing and there’s no end in sight. Every action we perform in our day to day lives leads to creation of data. As the volume of data increases, inexorably, the proportion of it that people could understand decreases alarmingly. Hence, to understand the maximum possible extent of data better, a pattern must be identified. The concept of machine learning does this for us.

Data mining is about solving problems by analyzing data already present in databases. In data mining**,** the data is stored electronically and the search is automated or at least augmented by computer. Say for example, the problem is fickle customer loyalty in a highly competitive marketplace. A database of customer choices, along with customer profiles, holds the key to this problem. Patterns of behavior of former customers can be analyzed to identify distinguishing characteristics of those likely to switch products and those likely to remain loyal [1].

One of the most popular tools that can be used for data mining is WEKA (*Waikato Environment for Knowledge Analysis)*. The WEKA workbench is a collection of modern machine learning algorithms and data preprocessing tools. It was developed at the University of Waikato in New Zealand. The system is written in Java and distributed under the terms of the GNU General Public License. It runs on almost any platform and has been tested under Linux, Windows, and Macintosh operating systems and even on a personal digital assistant. It provides a uniform interface to many different learning algorithms, along with methods for pre- and post-processing and for evaluating the result of learning schemes on any given dataset.

WEKA is an open source project, so in the course of a few years, it has developed into a complete Data Mining tool. It was initially designed by University of Waikato as a research and experimentation tool, but was quickly noticed by others who found its huge benefits. In fact, according to a recent web article, Data Mining users usually perform their analysis now a days using WEKA [2].

Some of WEKA’s advantages:

* Contains a lot of algorithms
* Free (most other Data Mining tools are very expensive)
* Open source, so adapting it to your own needs is possible
* Constantly under development (not only by the original designers)

The idea of Data Mining is to build models of datasets that help to retrieve valuable information from the set. The ways to come to such models can be divided in different groups of basic tasks:-

* **Prediction**: Assume an attribute is partially defined by others. It could be possible to predict the value of that attribute for a new record that does not contain this attribute. For example, a prediction could be made of the average balance of a new customer if only some basic attributes are known about him.
* **Classification**: Almost the same as prediction, but with that difference that the value we want to predict is nominal. With classification it could be possible (for example) to determine whether a new customer will refund money if it is lent to him.
* **Clustering**: Grouping records in groups of “similar” records. (’similar’ as in the values of their attributes)
* **Association**: Grouping attributes that seem to have similar values for their values of the records.

Also some related tasks can be specified:

* Loading datasets from databases, text files, spreadsheets, Internet or other types of sources.
* Preprocessing datasets: Modifying, adding or deleting values and attributes, filtering records that are not interesting for the evaluation, etc.
* Testing accuracy or predictive value of a model that has been built.
* Visualizing the records or the model, according to different criteria

**2.2 Introduction to the project**

Various approaches have been proposed for malware detection. Detection techniques proposed earlier were based on static analysis. Static analysis examines the binary code, analyzes all possible execution paths, and identifies malicious code without execution. However, analyzing binary code turns out to be difficult nowadays. As obfuscation techniques become more sophisticated, static analysis can be bypassed by various obfuscation techniques, such as polymorphism, encryption, or packing. In addition, as static analysis relies on a prebuilt signature database, it cannot easily detect new unknown malware until the signature is updated. Besides, some execution paths can be only explored after execution. To overcome these limitations of static analysis and complement it, dynamic analysis has been proposed and is widely used to achieve more effective malware detection.

Techniques based on dynamic analysis execute malware and trace its behaviors. One of the major approaches in dynamic analysis is the API call analysis. This approach detects malware based on analysis of similarity between the behavior of the new and the known ones.

In this project, I have mainly focused on a dataset that contains two types of system API call classes, namely “Malware class” or “Benign class”. Software that “deliberately fulfills the harmful intent of an attacker” is referred to as malicious software or **malware** [3]. These are intended to gain access to computer systems and network resources, disturb computer operations, and gather personal information without taking the consent of system’s owner, thus creating a menace to the availability of the internet, integrity of its hosts, and the privacy of its users. Malwares come in wide range of variations like Virus, Worm, Trojan-horse, Rootkit, Backdoor, Botnet, Spyware, Adware etc. On the other hand, **benign classes** are those which do not harm the system in anyway. They are simple instructions that are used to carry out simple processes. Application program interface (**API**) is a set of procedures, protocols, and tools for building software applications. Every time a call is made to a server in name of an application, it counts as an API request or **API call**.

The dataset chosen for the project was certified by an authentic source. The malware samples present in the dataset were stated to have been duly checked by the recent anti-virus software. Although there were certain subcategories such as worm, trojan, virus, etc, in that data, all the malicious software types were grouped as the 'malware' group and assigned labels as '1'. The remaining benign software programs were assigned label '0'.

After analyzing the dataset using WEKA, the result was used to compare the effectiveness of the different classifying algorithms provided by the said toolkit and on the basis of this comparison, the best suited algorithms could be identified in this project.

**3. Objective**

The main objective of undertaking this project is to detect the presence of malwares from the API calls effectively using WEKA – a data mining toolkit. To implement the concepts of machine learning for problem solving to identify the best data classifying techniques by comparing the results generated after every classification process. The performance measures taken to compare the different classifying algorithms are **F-measure, ROC area** and **Accuracy**.

**4. Literature Review**

**4.1 Related Work**

* This section discusses a brief background and some related works for malware detection in data mining methods. Malicious code is one of the serious threats on the internet platform that is called malware [4].

Kolter (2004) used n-gram and data mining method to detect malicious executable. They used different classifiers including Naive-Bayes, Support Vector Machine, Decision Tree and their boosted versions. They concluded that boosted decision tree gives the best classification results [5].

Rieck (2011) proposed a framework for automatic analysis of malware behavior using machine learning. This framework collected large number of malware samples and monitored their behavior using a sandbox environment. By embedding the observed behavior in a vector space, they apply the learning algorithms. Clustering is used to identify the novel classes of malware with similar behavior. Assigning unknown malware to these discovered classes is done by classification. Based on both, clustering and classification, an incremental approach is used for behavior-based analysis, capable of processing the behavior of thousands of malware binaries on daily basis. However, the same could not be followed in carrying out the project due to the absence of such facilities at DRDO [6].

* This section talks about the classification algorithms provided by WEKA and the comparison between them. The dataset was classified using some classification algorithms such as NaiveBayes, BayesNet, IB1, J48, and regression algorithms. The regression classification method had best performance for classification of malware detection [7]. But in my experiment, Regression algorithm was unavailable for the particular dataset.

Ponciano, Pais and Casal (2015) stated that having analysed the results of 22 classifiers namely NaiveBayes, BayesNet, J48, Regression, REPTree, etc., it was concluded the classifiers that achieved the best results were the J48, BayesNet and NB tree [8].

Deshmukh, Patil, & Panwar (2011) studied and compared BayesNet, Naïve Bayes, SMO, RBFNetwork, Logistic, Decision Trees - J48, ADTree, NBTree and Decision table algorithms on five separate data sets in WEKA. The arrived to a conclusion that the different classification algorithms are designed to perform better for certain types of dataset [9].

**4.2 Research Conclusion**

Perusal of the above discussed research papers was helpful in gaining idea about the functioning of various classification algorithms. It was noticed that different approaches can be made in order to detect malware by using the concept of machine learning. The research paper written by B. Deshmukh, A. S. Patil, & B. V Pawar (2011) reported that the most popular classification algorithm may not give out the most accurate result for every dataset [8].

Hence, in conclusion, it can be stated that the efficiency of different classification algorithms depend on the dataset that is being used. Therefore, while carrying out the project, it was more appropriate to try the algorithms practically on the dataset to deduce a result.

**5. Implementation**

**5.1 Overview**

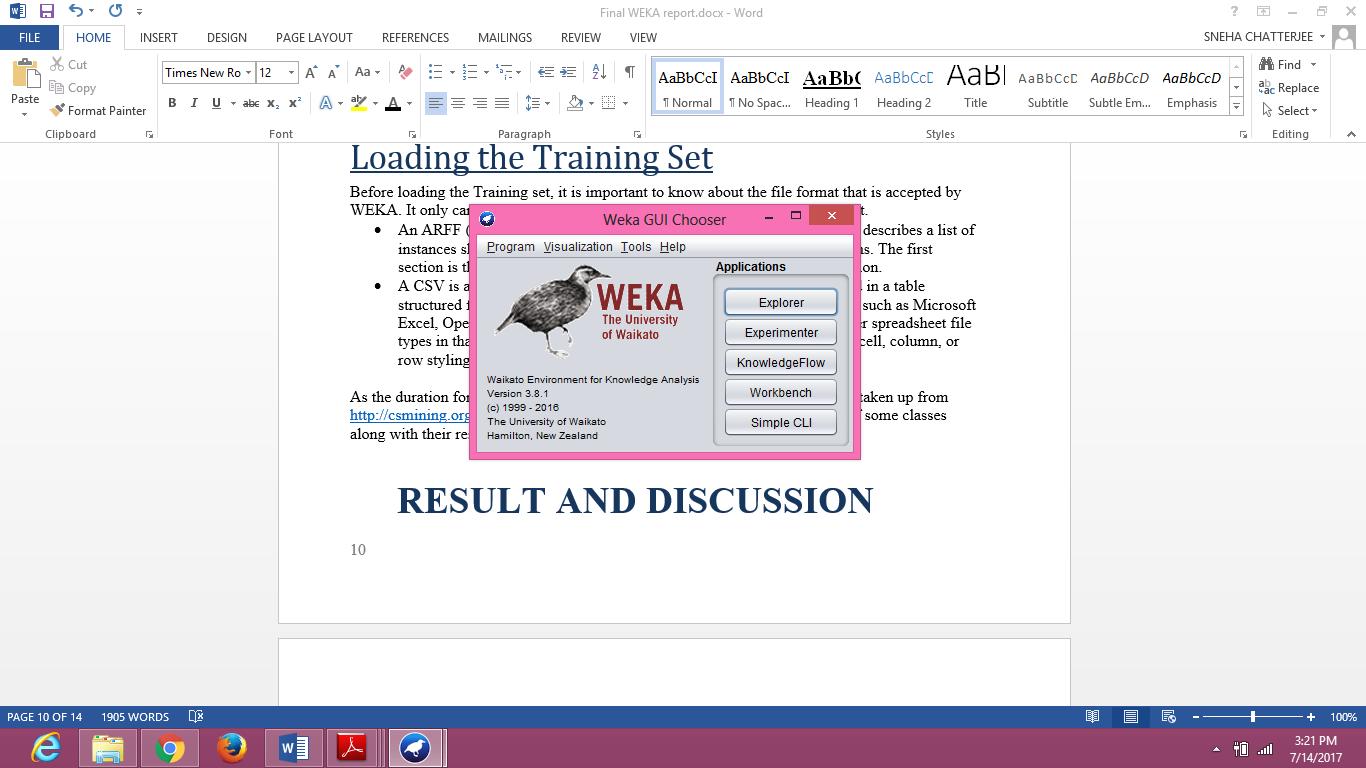
The usefulness of WEKA has already been discussed above. However, the uniqueness of the toolkit is not confined to those advantages only. Therefore, my aim was to see whether the same can also be used where enormous amount of complex data is involved. I am of the view that use of the toolkit for so many other applications can also be established by carrying out similar projects using WEKA.

The methodology that has been followed in this project consists of the following steps:

* **Phase 1 –** This phase included installing WEKA on the system, followed by gaining basic knowledge about the tool and practicing simple instructions based on the study.
* **Phase 2 –** The type of dataset that would be most suitable for the project was decided and an appropriate dataset for this project was obtained from an authentic source for analysis. (**APPENDIX A**)
* **Phase 3 –** After the dataset was obtained, it was formatted in a way so that it would only consist of two columns, “**Class**” and “**API Calls**”. The class used to represent Malware as “1” and Benign as “0”. The resulting dataset was then converted into a WEKA readable format, i.e., in the CSV or ARFF file format. However, while using WEKA, it is quite convenient to convert the file format from CSV to ARFF or vice versa.
* **Phase 4 –** This modified dataset wasthen opened using WEKA. The API Calls were then segregated and tokenized into sets of 3 for more accuracy.
* **Phase 5 –** A new dataset was thus obtained with a large number of attributes in a sequences of 3 API calls each.
* **Phase 6 –** The classifying algorithms were then applied onto the dataset separately and the results were analyzed.
* **Phase 7 –** The results were compared and an analysis report was generated which was used to distinguish between effectiveness of different algorithms and indicate which gave the most accurate output.
* **Phase 8 –** Finally, the results and the findings were noted down in the form of a report.

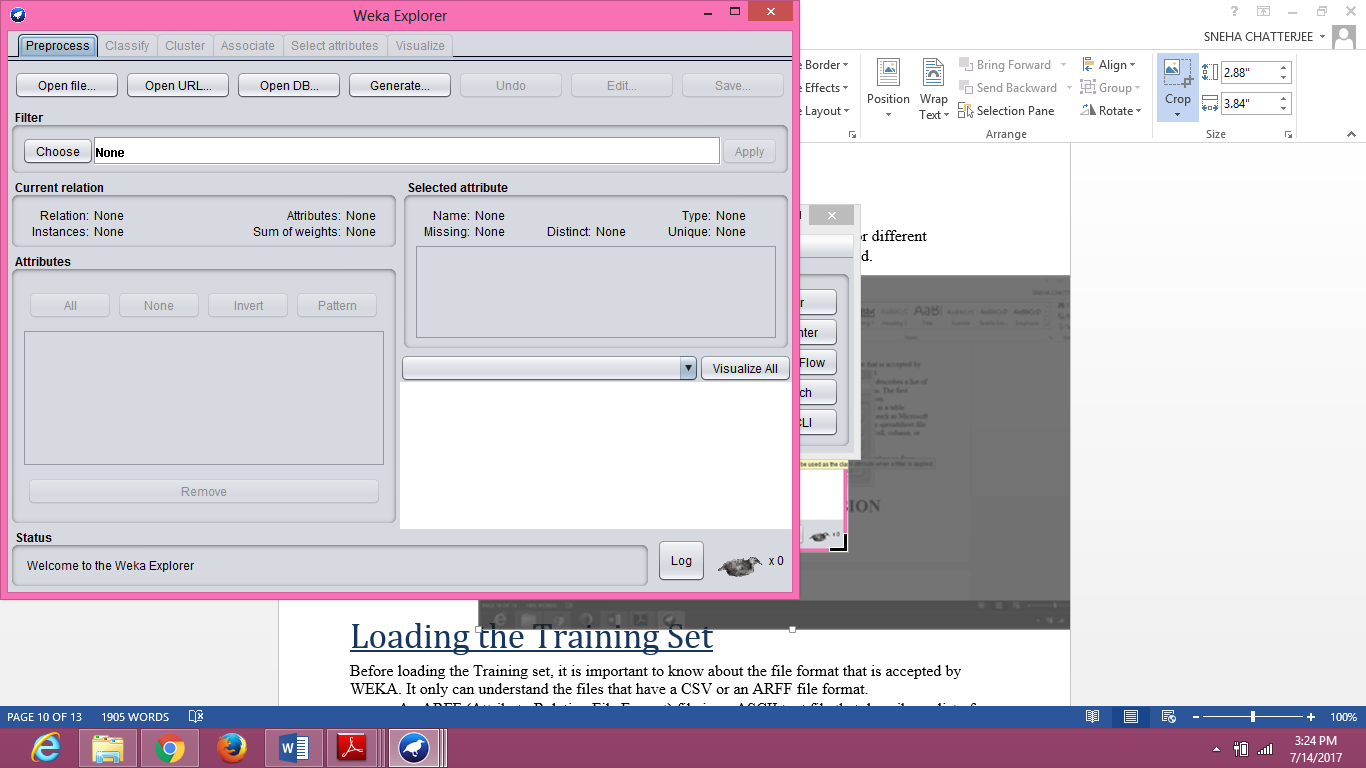
**5.2 Installing WEKA**

WEKA was downloaded from an online source [10] and then installed on the system by answering the yes or no questions during the installation. After completing the process, WEKA got listed on the start menu. On double clicking the icon, the first screen that popped up was the WEKA GUI Chooser.



**Fig 1. WEKA GUI**

The GUI provides us with a variety of options to choose from for different purposes. On choosing the “Explorer” option, the WEKA Explorer was launched.



**Fig 2. WEKA Explorer**

**5.3 Loading the Training Dataset**

Since WEKA uses the concept of machine learning, it must first be trained by using a training dataset that trains it to adopt a specific working pattern. In order to load the Training set, it is important to know about the file format that is accepted by WEKA. It only can understand the files that have a CSV or an ARFF file format as stated earlier.

* An ARFF (Attribute-Relation File Format) file is an ASCII text file that describes a list of instances sharing a set of attributes. ARFF files have two distinct sections. The first section is the **Header** information, which is followed the **Data** information.
* A CSV is a **comma separated values** file which allows data to be saved in a table structured format. CSV files can be used with any spreadsheet program, such as Microsoft Excel, Open Office Calc, or Google Spreadsheets. They differ from other spreadsheet file types in that you can only have a single sheet in a file, they cannot save cell, column, or row styling, and cannot save formulas.

As the time duration available for carrying out the project was very short, the dataset used was taken up from the internet. Initially the intention of the project was to base the analysis on a dataset created by capturing the live system API calls. Some work towards capturing of data was also done. However the dataset was not processed further as raw data obtained directly from my PC required processing and cleansing which was a time consuming task.

Therefore, I chose to work on a pre-processed dataset to finish the project within the stipulated time. The dataset was taken up from an online dataset repository that is publically available for research [11], the CSDMC2010 API sequence corpus which is one of the datasets for the data mining competition associated with ICONIP 2010. This dataset was composed of a selection of Windows API/System-Call trace files, intended for testing on classifiers treating with sequences.

**The corpus description:-**

The dataset contained two parts:

* **TRAINING**: 388 logs out of which there were 320 malware traces labeled as '1' and 68 benign software traces labeled as '0'.

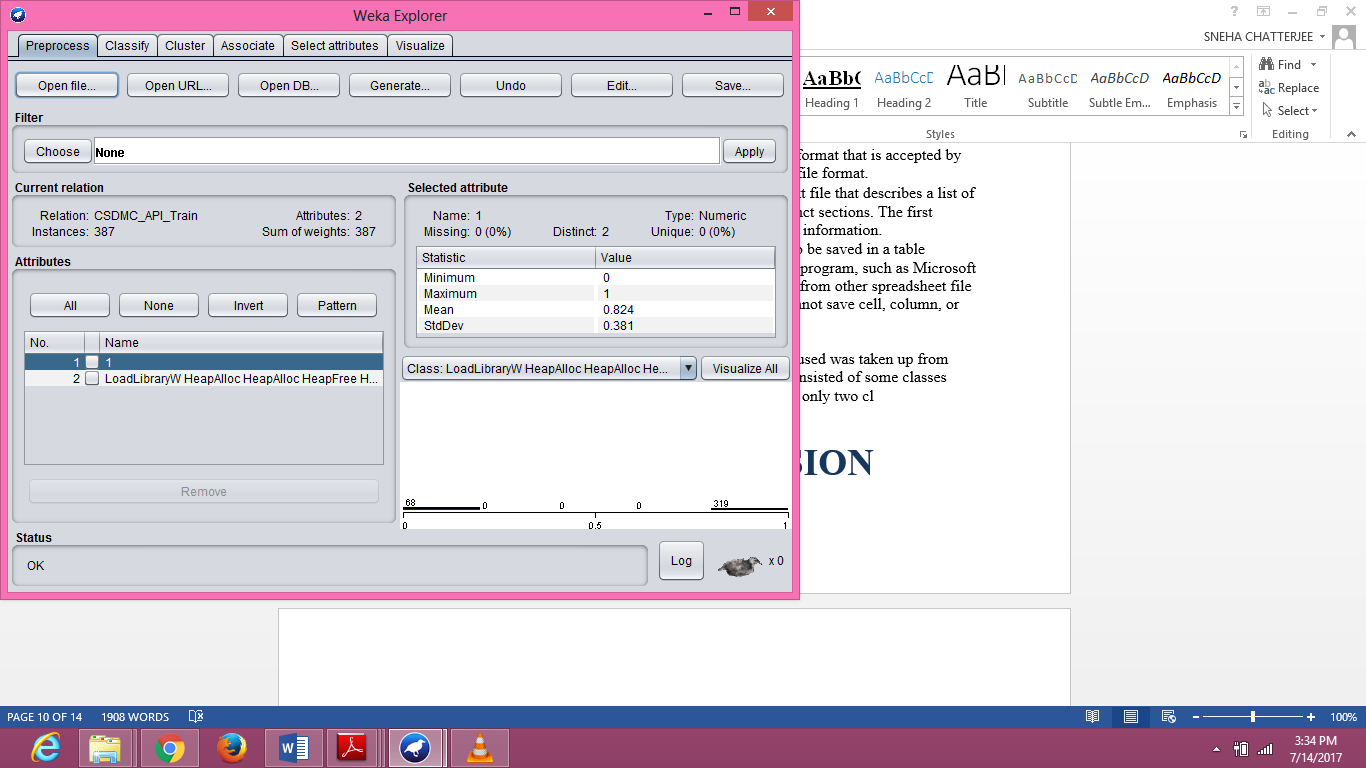
Each line in the data file corresponded to a trace of a designated software. The label was given at the beginning of a line, with a comma separating the label and the corpus.

* **TESTING**: 378 traces with unknown labels - labeled as all 0's in the file.

**Pertinent points:-**

* A subset of the Windows APU/System-Calls which were considered informative for differentiating a malware from a benign software were logged by API monitors when a designated program was running in the system.
* For simplicity, only the names of the APIs were present in the log file without noting the calling-process.
* For completeness, reduplicated calls of the same APIs were all recorded, which resulted in some redundancy in the log though.
* Malware samples were labeled by the state-of-the-art anti-virus software. Although there were certain subcategories such as worm, trojan, virus, etc, all these malicious software types were grouped as 'malware' group, which were assigned labels as '1'. The remaining benign software programs were assigned label '0'.
* The order of the system calls were preserved as good as possible. Efforts were made to make the ordering of the sequences not misleading, by keeping the original order of the records as in the log when time information was not available, or by sorting the sequence according to the timestamp when possible.

On directly running the dataset in WEKA, the following output was retrieved:



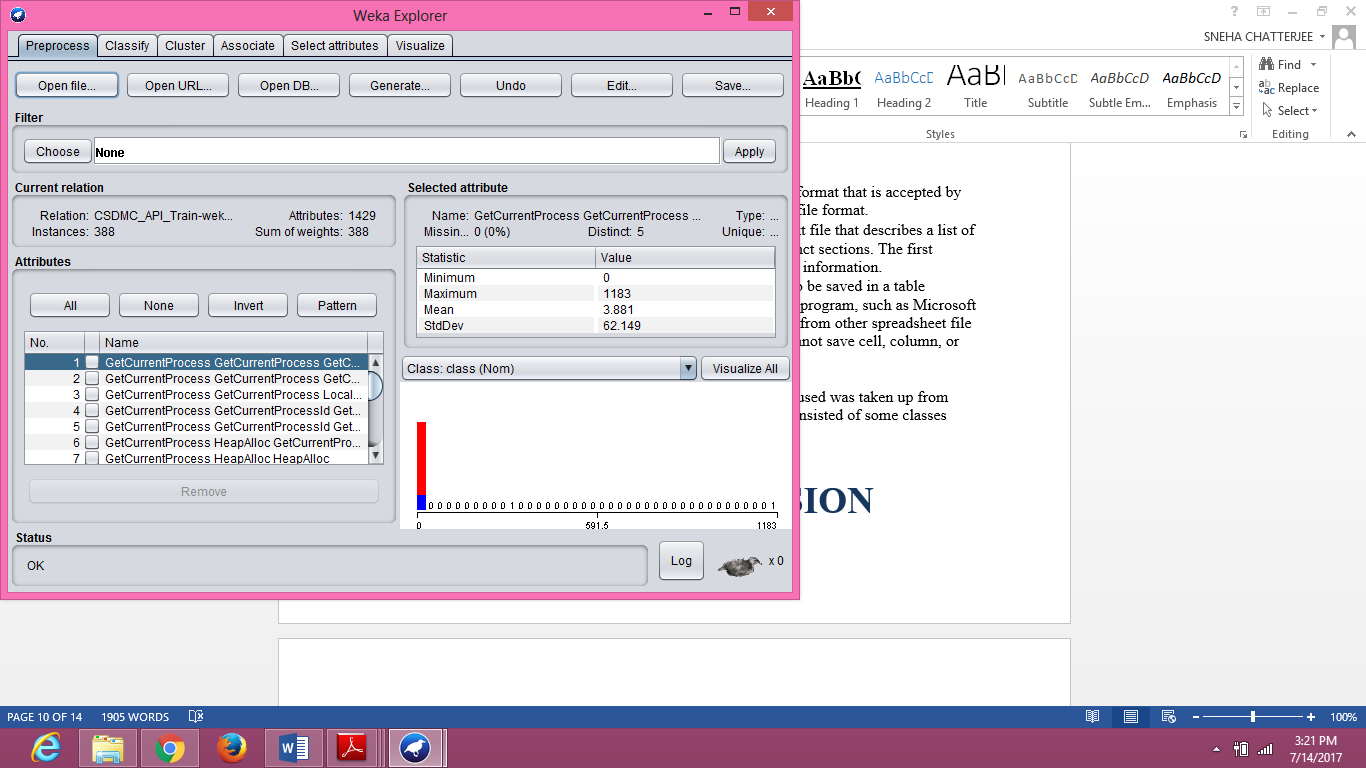
**Fig 3. Opening the dataset in WEKA**

The attributes were not clear in the above display. Therefore, the dataset needed to be formatted such that all the system API calls could be separately displayed. The attributes were previously of “nominal” type. So, first they were converted to string type, followed by using the *StringToWordVector* which further processes the dataset into word frequencies. It converts a string attribute to a vector that represents word occurrence frequencies.

We can further calculate the **Term Frequency** (TF) and **Inverse Document Frequency** (IDF) to count the frequency of the malware classes. On multiplying both the frequencies, we get the Weight of the specific class. The more frequent ones (i.e., the ones having more weight) are the ones that are dangerous and may harm the system.

‘c’

The chances of finding out if an API call is a malicious one or not just by looking at it may be difficult but, if this call is placed in a sequence of say, three calls, we can look at the sequence and infer if the sequence leads to any malicious process or not.



**Fig 4. Tokenization of data**

**5.4 Analyzing the Test Dataset**

The above mentioned steps were repeated with the test dataset and the final output was classified using WEKA. The comparison was done taking into account three parameters, **F-Measure, ROC Area** and **Accuracy Percentage.**

The F-Measure is chosen as an important parameter for the comparison because it includes most of the basic parameters.

*F-Measure =*

Where, precision is the ratio between *true positive* and *predicted positive.* It checks what fraction of those predicted positive are actually positive. Precision is also referred to as **Positive predictive value** (PPV) and can be calculated as follows:

*Precision =*

Recall is the ratio between *true positive* and *actual positive.* It checks what fraction of those that are actually positive were predicted positive. It is also referred to as **Sensitivity** and can be calculated as follows:

*Recall =*

Here, TP = true positives: number of examples predicted positive that are actually positive.

FP = false positives: number of examples predicted positive that are actually negative.

TN = true negatives: number of examples predicted negative that are actually negative.

FN = false negatives: number of examples predicted negative that are actually positive.

A ROC curve is plotted between the TP Rate (Sensitivity) and the FP Rate (Specificity) for different cut-off points of a parameter. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. The area under the ROC curve is a measure of how well a parameter can distinguish between two diagnostic groups (malware/benign). Therefore, more the area under the curve, more precise will be the result.

Accuracy can be calculated as follows:

*Accuracy =*

Top 10 most efficient classification algorithms were chosen from the research papers and were tested on the dataset. The algorithms used to test were BayesNet, NaiveBayes, MultilayerPerception, DecisionTable, PART, JRip, LMT, J48, RandomForest and REPTree.

**6. Result and Conclusion**

**6.1 Result**

Following are the classifiers that were used along with their results:-

|  |  |  |  |
| --- | --- | --- | --- |
| **CLASSIFIER** | **F-MEASURE** | **ROC AREA** | **ACCURACY PERCENTAGE** |
| BayesNet | 0.936 | 0.976 | 94.07% |
| NaïveBayes | 0.921 | 0.927 | 92.52% |
| MultilayerPerceptron | 0.934 | 0.965 | 93.81% |
| DecisionTable | 0.969 | 0.985 | 96.90% |
| PART | 0.974 | 0.943 | 97.42% |
| JRip | 0.977 | 0.956 | 97.68% |
| LMT | 0.979 | 0.980 | 97.93% |
| J48 | 0.977 | 0.967 | 97.68% |
| RandomForest | 0.987 | 0.994 | 98.71% |
| REPTree | 0.979 | 0.954 | 97.93% |

**Fig 5. Graphical representation of classifiers on the basis of comparison between the value of F-measure, ROC Area and Accuracy Percentage**

From the above results, it could be seen that **RandomForest** was the most efficient algorithm as compared to the other.

**6.2 Conclusion**

The result suggested the use of **RandomForest** for this particular type of dataset as the result obtained had high accuracy %, greater area under the ROC curve as compared to the ROC Area of other classification algorithms and higher F-measure value. Therefore, towards the end of the project, it can be concluded by saying that this approach can be followed in order to address the problem of malware detection from the system/API calls. The problem could be correctly solved.

**7. Future Work**

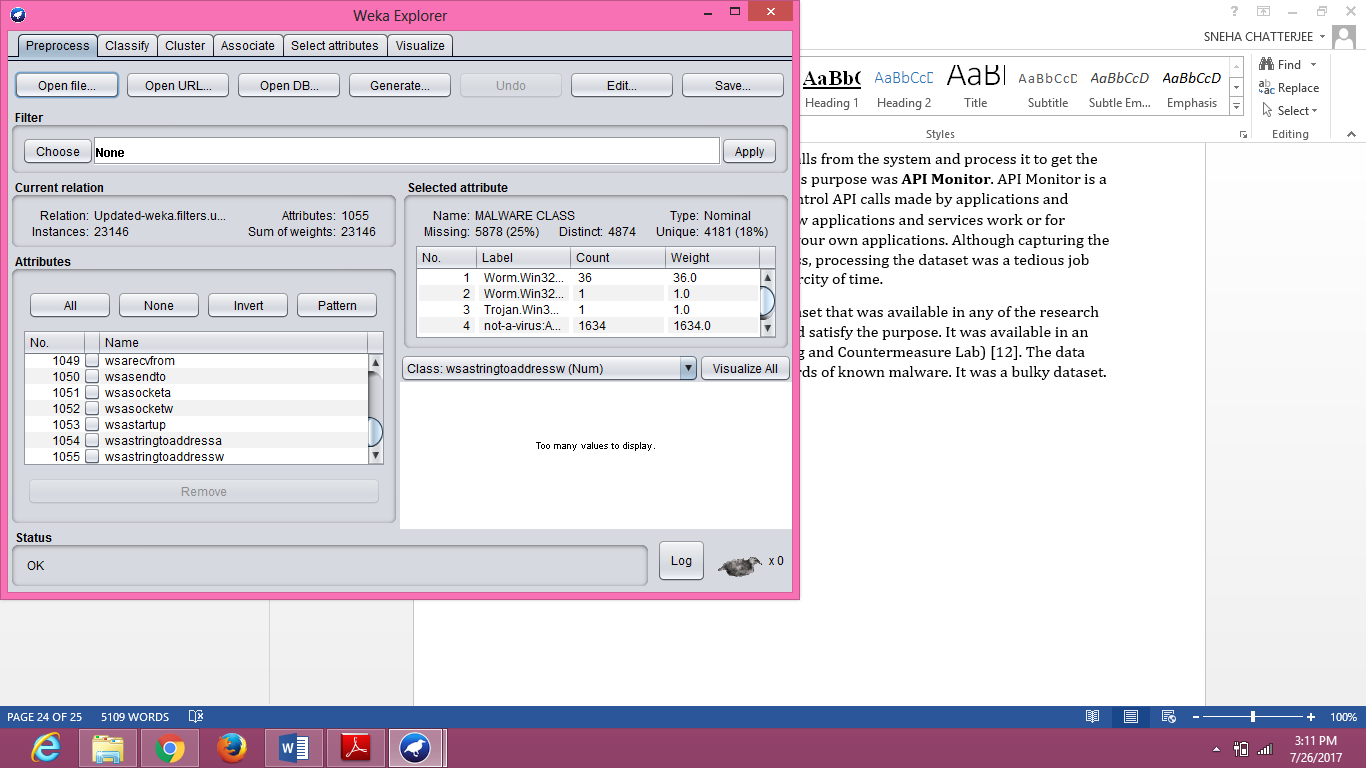
As I was not very well acquainted with data mining and machine learning before, it took me some time to understand the basic concepts. Due to scarcity of time, the project has been carried out by using a dataset that was taken up from an online dataset repository. Further, the same experiment can be carried out on a dataset that includes the API calls captured live from the system. This can be done by using of specific software such as, API Monitor or Process Monitor. After capturing the calls, the dataset must be processed and the above analysis can be carried forward using it.

This approach can be useful in order to develop a model that can act as a defence mechanism against malwares that spread from one system to another. Large amount of training data must be fed to the model before dispatching it for more accuracy and efficiency.

**7. Appendix**

Initially, the idea was to capture live API calls from the system and process it to get the required dataset. The software used for this purpose was **API Monitor**. API Monitor is a free software that lets you monitor and control API calls made by applications and services. It is a powerful tool for seeing how applications and services work or for tracking down problems that you have in your own applications. Although capturing the system’s API calls was a time taking process, processing the dataset was a tedious job that could not be completed due to the scarcity of time.

The second approach was search for a dataset that was available in any of the research websites. I came across a dataset that could satisfy the purpose. It was available in an online dataset repository of HCRL (Hacking and Countermeasure Lab) [12]. The data was clearly labeled and it had 23,000 records of known malware. Although, the malware file itself was not provided, instead full list of API sequences and hash information was made available. It was a bulky dataset. After tokenizing, the following result was obtained:

****

But, the problem with this dataset was that there were a large number of instances and the dataset was not processed. Pruning was required to remove the unwanted data so that a precise dataset could be formed. Without manual processing and pruning, the dataset was unable to produce a fruitful output and hence, this dataset was discarded.

**8. References**

[1] Witten, I. H., Frank, E., Hall, M. A., & Pal, C. J. (2016). The WEKA workbench. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, Third Edition, 2016.

[2] Goopta, C., Wheeler, B. and Joshi, M. (2017). Six of the Best Open Source Data Mining Tools - The New Stack. [online] The New Stack. Available at: https://thenewstack.io/six-of-the-best-open-source-data-mining-tools/ [Accessed 17 Jul. 2017].

[3] Bayer, U., Moser, A., Kruegel, C. and Kirda, E. (2006) Dynamic Analysis of Malicious Code. *Journal in Computer Virology*, **2**, 67-77.

[4] D. B. Ekta Gandotra and S. Sofat, “Malware analysis and classification: a survey,” Journal of Information Security, vol. 5, pp. 56–64, 2014.

[5] Kolter, J. and Maloof, M. (2004) Learning to Detect Malicious Executables in the Wild. Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 470-478.

[6] Rieck, K., Trinius, P., Willems, C. and Holz, T. (2011) Automatic Analysis of Malware Behavior Using Machine Learning. Journal of Computer Security, 19, 639-668.

[7] Norouzi, M., Souri, A. and Samad Zamini, M. (2016). *A Data Mining Classification Approach for Behavioral Malware Detection*, pp. 6-8.

[8] Ponciano, R., Pais, S. and Casal, J. (2015). Using Accuracy Analysis to Find the Best Classifier for Intelligent Personal Assistants - ScienceDirect. [online] ScienceDirect. [Accessed 22 Jul. 2017].

[9] Deshmukh, B., Patil, A. S., & Pawar, B. V. (2011) Comparison of Classification Algorithms using WEKA on Various Datasets, 86-90.

[10] Cs.waikato.ac.nz. (n.d.). *Weka 3 - Data Mining with Open Source Machine Learning Software in Java*. [online] Available at: http://www.cs.waikato.ac.nz/ml/weka/citing.html [Accessed 26 Jul. 2017].

[11] Csmining.org. (n.d.). *Malicious datasets \* - Csmining Group*. [online] Available at: http://csmining.org/index.php/malicious-software-datasets-.html [Accessed 26 Jul. 2017].

[12] Youngjoon Ki, Eunjin Kim, and Huy Kang Kim, “A Novel Approach to Detect Malware Based on API Call Sequence Analysis,” International Journal of Distributed Sensor Networks, vol. 2015, Article ID 659101, 9 pages, 2015. doi:10.1155/2015/659101