IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

MULTI-CLASSIFICATION FRAMEWORK FOR MALICIOUS SEQUENCE PATTERN MATCHING

KARTHIKEYAN.A[1], ASHWATHY.G. L[2], BRINDHA.R[3], DHARANI.V[4], VIDHYA.T[5]
1 ASSOCIATE PROFESSOR 2,3,4,5 UG SCHOLAR
COMPUTER SCIENCE AND ENGINEERING
MAHENDRA ENGINEERING COLLEGE (AUTONOMOUS), NAMAKKAL, TAMILNADU, INDIA

ABSTRACT

Malware detection developer faced a problem for a generation of new signature of malware code. A very famous and recognized technique is pattern based malware code detection technique. This leads to the evasion of signatures that are built based on the code syntax. In this paper, discuss some well known method of malware detection based on semantic feature extraction technique. In current decade, most of authors focused on malware feature extraction process for generic detection process. The effectiveness of the Malicious Sequence Pattern Matching technique for malware detection invites for moderation and improvement of the current system and method. Some authors used rule mining technique, some other used graph technique and some also focused on feature clustering process of malware detection.

Keywords: Malware detection, code syntax & Sequence Pattern

1.INTRODUCTION

1.1 DATA MINING

Data mining is an interdisciplinary subfield of computer science. It is the computational process of discovering patterns in large data sets involving methods at the intersection of artificial intelligence, machine learning, statistics, and database systems. The overall goal of the data mining process is to extract information from a data set and transform it into an understandable structure for further use. Aside from the raw analysis step, it involves database and data management aspects, data pre-processing, model and inference considerations, interestingness metrics, complexity considerations, post-processing of discovered structures, visualization, and online updating. Data mining is the analysis step of the "knowledge discovery in databases" process, or KDD. The term is a misnomer, because the goal is the extraction of patterns and knowledge from large amounts of data, not the extraction (mining) of data itself. It also is a buzzword and is frequently applied to any form of large-scale data or information processing (collection, extraction, warehousing, analysis, and statistics) as well as any application of computer decision support system, including artificial intelligence, machine learning, and business intelligence.

The book Data mining: Practical machine learning tools and techniques with Java (which covers mostly machine learning material) was originally to be named just Practical machine learning, and the term data mining was only added for marketing reasons. Often the more general terms (large scale) data analysis and analytics – or, when referring to actual methods, artificial intelligence and machine learning – are more appropriate. The actual data mining task is the automatic or semi-automatic analysis of large quantities of

data to extract previously unknown, interesting patterns such as groups of data records (cluster analysis), unusual records (anomaly detection), and dependencies (association rule mining). This usually involves using database techniques such as spatial indices. These patterns can then be seen as a kind of summary of the input data, and may be used in further analysis or, for example, in machine learning and predictive analytics. For example, the data mining step might identify multiple groups in the data, which can then be used to obtain more accurate prediction results by a decision support system. Neither the data collection, data preparation, nor result interpretation and reporting is part of the data mining step, but do belong to the overall KDD process as additional steps. The related terms data dredging, data fishing, and data snooping refer to the use of data mining methods to sample parts of a larger population data set that are (or may be) too small for reliable statistical inferences to be made about the validity of any patterns discovered. These methods can, however, be used in creating new hypotheses to test against the larger data populations.

1.2 CLASSIFICATION AND PREDICTION:

Classification is the process of finding a model that describes the data classes or concepts. This derived model is based on the analysis of sets of training data. The derived model can be presented in the following forms:

Classification (IF-THEN) Rules
Decision Trees
Mathematical Formulae
Neural Networks

The list of functions involved in these processes is as follows:

Classification - It predicts the class of objects whose class label is unknown.

Its objective is to find a derived model that describes and distinguishes data classes or concepts. The Derived Model is based on the analysis set of training data i.e. the data object whose class label is well known.

Prediction - It is used to predict missing or unavailable numerical data values rather than class labels. Regression Analysis is generally used for prediction. Prediction can also be used for identification of distribution trends based on available data.

Outlier Analysis - Outliers may be defined as the data objects that do not comply with the general behavior or model of the data available. Evolution Analysis - Evolution analysis refers to the description and model regularities or trends for objects whose behavior changes over time.

The list of steps involved in the knowledge discovery process:

Data Cleaning - In this step, the noise and inconsistent data is removed.

Data Integration - In this step, multiple data sources are combined.

Data Selection - In this step, data relevant to the analysis task are retrieved from the database. Data Transformation - In this step, data is transformed or consolidated into forms appropriate for mining by performing summary or aggregation operations.

Data Mining - In this step, intelligent methods are applied in order to extract data patterns. Pattern Evaluation - In this step, data patterns are evaluated.

Knowledge Presentation - In this step, knowledge is represented.

1.3 BACKGROUND OF THE PROJECT

It discovers malicious sequential patterns based on the mechanism instruction sequences extracted from the Windows Portable Executable (PE) files, then use it to construct a data mining framework, called MSPMD (Malicious Sequential Pattern based Malware Detection), to detect new malware samples, and then NAIVE BAYES classifier is constructed for malware detection based on the discovered patterns. The residential data mining framework composed of the proposed sequential pattern mining method and NAIVE BAYES classifier can well portray the malicious patterns from the composed file sample set to effectively detect newly unseen malware samples. it can reduce the execution time for detect the malicious sequences affected files. The multi-classification method also detects the types of malware affect the file.

2. LITERATURE SURVEY

2.1 TITLE: DLL MINER: STRUCTURAL MINING FOR MALWARE DETECTION. AUTHOR: MASOUD NAROUEI, MANSOUR AHMADI.

Existing anti-malware products usually use signature-based techniques as their main detection engine. Although these methods are very fast, they are unable to provide effective protection against newly discovered malware or mutated variant of old malware. Heuristic approaches are the next generation of detection techniques to mitigate the problem. These approaches aim to improve the detection rate by extracting more behavioral characteristics of malware. Although these approaches cover the disadvantages of signature-based techniques, they usually have a high false positive, and evasion is still possible from these approaches. In this paper, we propose an effective and efficient heuristic technique based on static analysis that not only detect malware with a very high accuracy, but also is robust against common evasion techniques such as junk injection and packing. Our proposed system is able to extract behavioral features from a unique structure in portable executable, which is called dynamic-link library dependency tree, without actually executing the application.

2.2 TITLE: AN INTELLIGENT PE-MALWARE DETECTION SYSTEM BASED ON ASSOCIATION MINING AUTHOR: YANFANG YE · DINGDING WANG · TAO LI.

The proliferation of malware has presented a staid threat to the security of computer systems. Traditional signature-based anti-virus systems fail to identify polymorphic metamorphic and new, previously unseen malicious executables. Data mining methods such as Naive Bayes and Decision Tree contains studied on small collections of executables. In this paper, hidden on the analysis of Windows APIs called by PE files, we extend the Intelligent Malware Detection System (IMDS) using Objective-Oriented Association mining based classification. IMDS is an integrated structure consisting of three major modules: PE parser, OOA rule maker, and rule based classifier. An OOA_Fast_FP-Growth algorithm is adapted to competently generate OOA rules for classification. A broad experimental study on a large group of PE files obtained from the anti-virus laboratory of King Soft Corporation is performed to evaluate various malware detection approaches. Promising investigational results demonstrate that the precision and efficiency of our IMDS system outperform popular anti-virus software such as Norton Antivirus and McAfee Virus Scan, as well as preceding data mining based finding systems which employed Naive Bayes, Support Vector Machine (NAIVE BAYES) and Decision Tree techniques.

2.3 TITLE: DATA MINING METHODS FOR DETECTION OF NEW MALICIOUS EXECUTABLES AUTHOR: MATTHEW G. SCHULTZ AND ELIZA ESKIN

A serious safety threat today is malicious executables, especially new, unseen malicious executables often arriving as email attachments. These new- fangled malicious executables are formed at the rate of thousands every year and pose a serious safety threat. Current anti-virus systems attempt to detect these new malicious programs with heuristics generated by hand. This approach is costly and oftentimes hopeless. In this paper, we present a data-mining framework that detects new, before unseen malicious executables accurately and mechanically. The data- mining framework mechanically found patterns in our data set and second-hand these patterns to detect a set of new malicious binaries. Comparing our finding methods with a usual signature based method; our method more than doubles the current detection rates for new malicious executables. Data, such as byte code, and use these patterns to detect future instances in similar data. Our framework uses classifiers to identify new malicious executables. A classifier is a rule set, or detection model, generated by the data mining algorithm that was skilled over a given set of training data.

2.4 TITLE: AUTOMATIC MALWARE CATEGORIZATION USING CLUSTER ENSEMBLE AUTHOR: YANFANG YE, TAO LI

Malware categorization is significant problem in malware analysis and has attracted a lot of attention of computer safety researchers and anti-malware industry recently. Today's malware samples are created at a rate of millions per day with the expansion of malware writing techniques. There is thus an urgent need of effective methods for routine malware categorization. The last few years, many clustering techniques have been employed for automatic malware classification. However, such techniques have isolated successes with partial effectiveness and efficiency, and few have been applied in actual anti-

malware industry. In this paper, resting on the analysis of instruction frequency and function-based instruction sequences, we increase an Automatic Malware Categorization System (AMCS) for mechanically grouping malware samples into families that share some common distinctiveness using a cluster ensemble by aggregating the clustering solutions generated by dissimilar base clustering algorithms. We propose a principled cluster group framework for combining individual clustering solutions based on the agreement partition. The domain knowledge in the form of sample-level constraints can be naturally included in the collection framework. In addition, to account for the characteristics of quality representations, to propose a hybrid hierarchical clustering algorithm which combines the virtues of hierarchical clustering's. The classification results of our AMCS system can be used to create signatures for malware families that are useful for malware detection.

2.5 TITLE: AUTOMATIC MALWARE CATEGORIZATION USING CLUSTER ENSEMBLE AUTHOR: YANFANG YE, TAO LI

Scanning files for signatures is an established technology, but exponential growth in unique malware programs has caused a detonation in signature database sizes. One solution to this problem is to use string signatures, each of which is a next byte sequence that potentially can match many variants of a malware family. However, it is not clear how to routinely generate these string signatures with a sufficiently low false positive rate. Hancock is the first string signature creation system that takes on this challenge on a large scale. To minimize the false positive rate, Hancock features a scalable model that estimates the occurrence possibility of arbitrary byte sequences in good ware programs, a set of library code recognition techniques, and diversity-based heuristics that ensure the contexts in which a signature is implanted in containing malware files are similar to one another. With these techniques shared, Hancock is able to mechanically generate string signatures with a false positive rate below 0.1%.

3. SYSTEM ANALYSIS

3.1 EXISTING SYSTEM

Due to its damage to Internet security, malware (e.g., virus, worm, Trojan) and its finding has caught the attention of both anti malware industry and researchers for decades. To protect genuine users from the attacks, the most significant line of defense against malware is anti-malware software products, which mostly use signature-based method for detection. However, this method fails to recognize new, unseen malicious executable. To solve this problem, in this paper, based on the instruction sequences extracted from the file sample set, we propose an effective sequence mining algorithm to discover malicious sequential patterns, and then J48 classifiers constructed for malware detection based on the discovered patterns. The developed data mining framework composed of the proposed sequential pattern mining method and J48 classifier can well characterize the malicious patterns from the collected file samples to effectively detect newly unseen malware samples.

3.2 PROPOSED SYSTEM

Sequence mining algorithm to discover malicious sequential patterns based on the machine instruction sequences extracted from the Windows Portable Executable (PE) files, then use it to construct a data mining frame work, called MSPMD (**M**alicious **S**equential **P**attern based **M**alware **D**etection), to detect new malware samples. The main contributions of this paper can be summarized as follows: Instruction sequences are extracted from the PE (Portable Executable) files as the preliminary features, based on which the malicious sequential patterns are mined in the next step. The extracted instruction sequences can well indicate the potential malicious patterns at the micro level. In addition, such kind of features can be easily extracted and used to generate signatures for the traditional malware detection systems. An effective sequential pattern mining algorithm, called MSPE (Malicious **S**equential **P**attern **E**xtraction), to discover malicious sequential patterns from instruction sequence.

MSPE introduces the concept of objective- oriented to learn patterns with strong abilities to distinguish malware from benign files. Moreover, we design a filtering criterion in MSPE to filter the redundant patterns in the mining process in order to reduce the costs of processing time and search space. This strategy greatly enhances the efficiency of our algorithm. NAIVE BAYES classifier for malware detection: We propose NAIVE BAYES classifier as detection module to identify malware. Different from

the traditional k-nearest-neighbor method, NAIVE BAYES chooses *k* automatically during the algorithm process.

More importantly, the NAIVE BAYES classifier is well-matched with the discovered sequential patterns, and is able to obtain better results than other classifiers in malware detection. To conduct a series of experiments to evaluate each part of our framework and the whole system based on real sample collection, containing both malicious and benign PE files. The results show that MSPMD is an effective and efficient solution in detecting new malware samples.

3.3 SYSTEM ARCHITECTURE

The ML is constructed by using Koras and opencv2 architecture. These frameworks give us two advantages. First, they allow us to plan neural networks and secondly they help us precompile the training algorithms so that an execution on GPU is possible which generally runs at a very fast rate for neural networks.

The malware files have an identifying hash value which consists of 20 characters. The hash value is unique for each malware file. Each file also has a class identifier. This is an integer that indicates the family of each malware family. The raw data in each file is the hexadecimal form of binary contents of each malware. The dataset also has some metadata information such as function calls extracted from binaries using the IDA disassembler tool.

Malware poses a serious threat to the computers of individuals, enterprises and other organizations. In the Windows operating system (OS), Application Programming Interface (API) calls are an attractive and distinguishable feature for malware analysis and detection as they can properly reflect the actions of portable executable (PE) files.

4.1 HARDWARE REQUIREMENTS:

PROCESSOR : Any Processor above 500 MHz
RAM : 128MB. HARD
DISK : 10 GB. COMPACT DISK :
650 MB.
INPUT DEVICE : Standard Keyboard, Mouse. OUTPUT
DEVICE : VGA Monitor

4.2 SOFTWARE REQUIREMENTS:

OPERATING SYSTEM : Windows OS
FRONT END : JAVA

5.SYSTEM TESTING

5.1 STATIC VS DYNAMIC TESTING

There are many approaches available in software testing. Reviews, walkthroughs, or inspections are referred to as static testing, whereas actually executing programmed code with a given set of test cases is referred to as dynamic testing. Static testing is often implicit, as proofreading, plus when programming tools/text editors check source code structure or compilers (pre-compilers) check syntax and data flow as static program analysis. Dynamic testing takes place when the program itself is run. Dynamic testing may begin before the program is 100% complete in order to test particular sections of code and are applied to discrete functions or modules. Static testing involves verification, whereas dynamic testing involves validation. Together they help improve software quality. Among the techniques for static analysis, mutation testing can be used to ensure the test-cases will detect errors which are introduced by mutating the source code.

The box approach

Software testing methods are traditionally divided into white- and black-box testing. These two approaches are used to describe the point of view that a test engineer takes when designing test cases.

5.2 TESTING LEVELS

There are generally four recognized levels of tests: unit testing, integration testing, system testing, and acceptance testing. Tests are frequently grouped by where they are added in the software development process, or by the level of specificity of the test. The main levels during the development process as defined by the SWEBOK guide are unit-, integration-, and system testing that are distinguished by the test target without implying a specific process model. Other test levels are classified by the testing objective.

5.2.1 UNIT TESTING

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors. Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle.

5.2.2 INTEGRATION TESTING

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed. Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

5.2.3 WHITE-BOX TESTING

White-box testing (also known as clear box testing, glass box testing, transparent box testing and structural testing) tests internal structures or workings of a program, as opposed to the functionality exposed to the end-user. In white-box testing an internal perspective of the system, as well as programming skills, are used to design test cases. The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. This is analogous to testing nodes in a circuit, e.g. in-circuit testing (ICT). While white-box testing can be applied at the unit, integration and system levels of the software testing process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

API testing (application programming interface) – testing of the application using public and private APIs

Code coverage – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)

Fault injection methods – intentionally introducing faults to gauge the efficacy of testing strategies

Mutation testing methods

Static testing methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important function points have been tested. Code coverage as a software metric can be reported as a percentage for: *Function coverage*, which reports on functions executed *Statement coverage*, which reports on the number of lines executed to complete the test 100% statement coverage ensures that all code paths or branches (in terms of control flow) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

5.2.4 BLACK-BOX TESTING

Black-box testing treats the software as a "black box", examining functionality without any knowledge of internal implementation. The testers are only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all- pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

5.2.5 GREY-BOX TESTING

Grey-box testing (American spelling: gray-box testing) involves having knowledge of internal data structures and algorithms for purposes of designing tests, while executing those tests at the user, or black-box level. The tester is not required to have full access to the software's source code. Manipulating input data and formatting output do not qualify as grey-box, because the input and output are clearly outside of the "black box" that we are calling the system under test. This distinction is particularly important when conducting integration testing between two modules of code written by two different developers, where only the interfaces are exposed for test.

5.2.6 SOFTWARE PERFORMANCE TESTING

Performance testing is generally executed to determine how a system or sub- system performs in terms of responsiveness and stability under a particular workload. It can also serve to investigate measure, validate or verify other quality attributes of the system, such as scalability, reliability and resource usage.

5.2.7 USABILITY TESTING

Usability testing is to check if the user interface is easy to use and understand. It is concerned mainly with the use of the application.

5.2.8 VALIDATION TESTING

Lot of module are available in the billing system, there are define in below concepts. In this software each module and particularly every, entity it should be tested, the particular input it could be got a correct input and correct out put it should be showed, at the same time the mobile no's only gotten digits only more than values not be supported. The report module and upload module and the user register module every module and this activity, performance to be tested in every entity successfully tested.

6. SCREENSHOTS

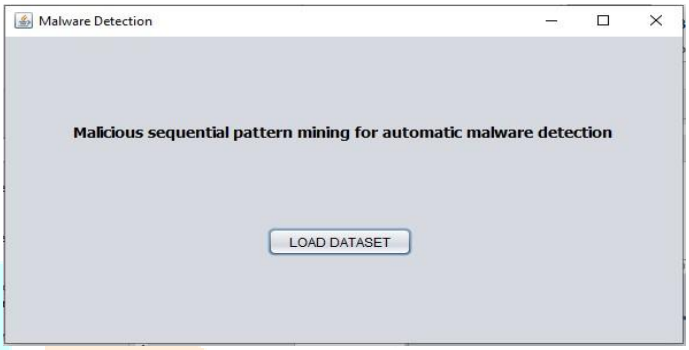

FIG 1 HOME PAGE OF MALWARE

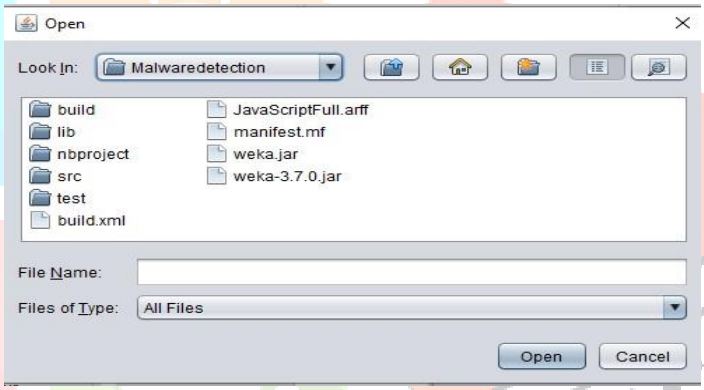

FIG 2 FILE SELECTION

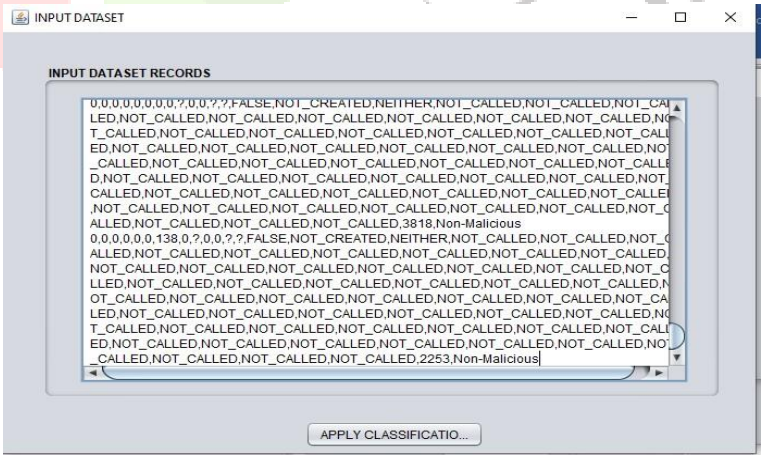

FIG 3 INPUT DATASET

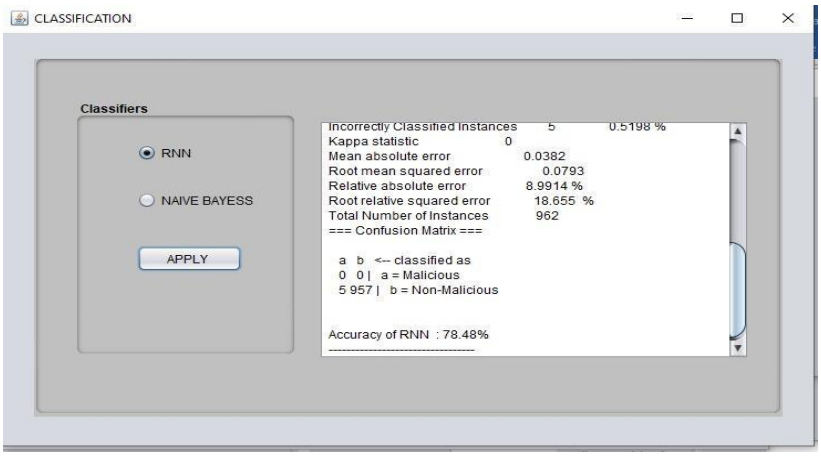

FIG 4 CLASSIFICATION OF RNN

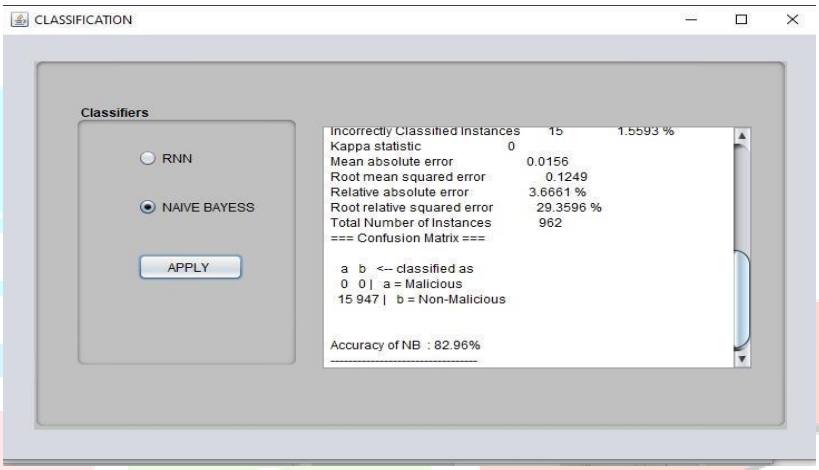

FIG 5 CLASSIFICATION OF NAIVE BAYESS

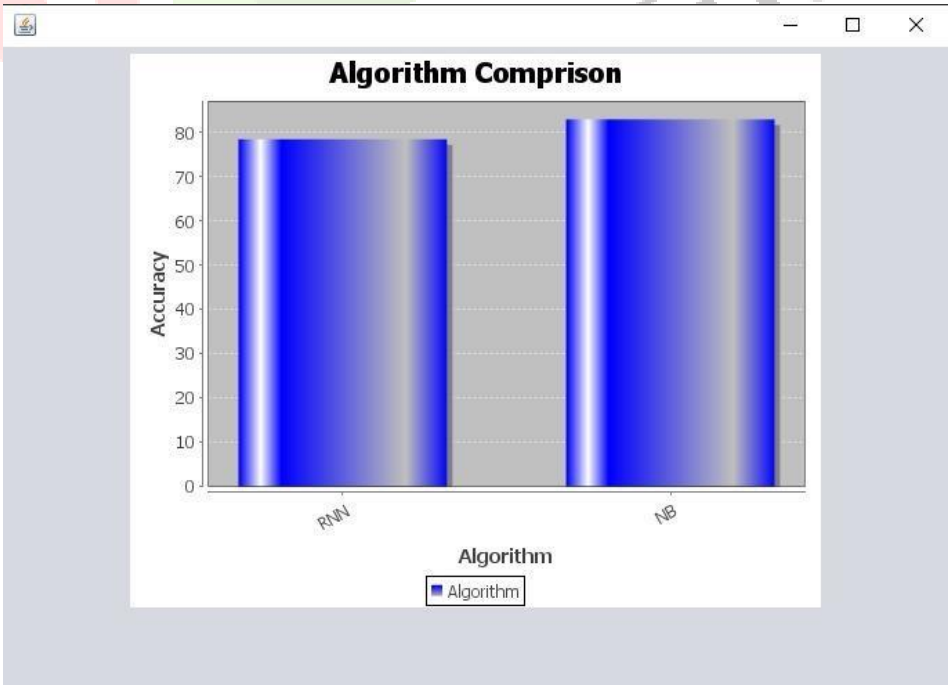

FIG 6 PAGE OF ACCURACY RESULTS

7.CONCLUSION

To develop a data-mining- based detection framework called Malicious Sequential Pattern based Malware Detection (MSPMD), which is composed of the proposed sequential pattern mining algorithm (MSPE) and NAIVE BAYES classifier. It first extracts instruction sequences from the PE file samples and conducts feature selection before mining; then MSPE is applied to generate malicious sequential patterns. For the testing file samples, after feature representation, NAIVE BAYES classifier is constructed for malware detection. The promising experimental results on real data collection demonstrate that our frame works out performs other alternate data mining base detection methods in identifying new malicious executables. Unlike the previous researches which are unable to mined is criminative features, we propose to use sequence mining algorithm on instruction sequence to extract well representative features.

8. REFERENCES

[1] Narouei, M., Ahmadi, M., Giacinto, G., Takabi, H., &Sami, A. (2015). DLL Miner: Structural mining for malware detection. Security and Communication Networks, 8.

[2] Ye, Y., Wang, D., Li, T., Ye, D., & Jiang, Q. (2008). An intelligent PE- malware detection system based on association mining. Journal in computer virology, *4*, 323–334.

[3] Ye, Y., Li, T., Chen, Y., & Jiang, Q. (2010). Automatic malware categorization using cluster ensemble. In Proceedings of the 16th international conference on knowledge discovery and data mining (pp.95–104).

[4] Wchner, T., Ochoa, M., & Pretschner, A. (2014). Malware detection with quantita-tive data flow graphs. In Proceedings of the 9th ACM symposium on information, computer and communications security (pp.271–282).

[5] Nissan., Moskovitch, R., Rokach, L., & Elovici, Y. (2014). A Novel approach for active learning methods for enhanced PC malware detection in windows OS. Expert Systems with Applications, 41,5843–5857.

[6] Ahmadi, M., Sami, A., Rahimi, H., & Yadegari, B. (2013). Malware detection by behavioral sequential patterns. Computer Fraud & Security, 2013, 11 – 19.

[7] R a d , B. B., Masrom, M. , & Ibrahim, S. (2012). Opcodes histogram f o r classifying metamorphic portable executable small ware. In Proceedings of international conference one- learning and e-technologies in education (pp.209–213).