Received: September 27, 2020. Revised: January 12, 2021.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Classification of Malware with MIST and N-Gram Features Using Machine Learning

Udayakumar Nandagopal[1]* **Subbulakshmi Thirumalaivelu[1]**

[1]*Vellore Institute of Technology Chennai, India*
* Corresponding author's Email: udayakumar.n2014@vit.ac.in

Abstract: Malwares (Malicious Software's) has increased rapidly in the recent years over the internet, In-order to detect the malwares many anti-malware strategies also been introduced but most of them relay on signature based techniques. But signature based technique failed to detect the new and unknown malwares, so signature based method failed to provide the better result. So it is important to find an technique that could able to detect the both new and unknown malwares without fail. Dynamic malware analysis with sandbox setting provides the best way to analyse the malware in the isolated environment. In this work, inside the cuckoo sandbox malwares were executed for observing behavioural accomplishments of the each and every Malicious Application. After executing the malware inside the sandbox, it will record the activities and finally the report will be provided in the JSON format. System calls needs to be generated from JSON file, MIST file is generated from the JSON file format. This will be organised in the N-Grams after extracting the system call, which helps construct the classifier using the Gained Information (GI) as the selection technique for the submission. Then the overall based on the N-Gram value and Byte length, various classifiers are evaluated, in that random forest has provided the best result.

Keywords: Malware detection, Gained information, MIST, Hypervisor, Classification, N-Gram, Sandboxing, Feature selection.

1. Introduction

The malicious code that has been created with the aim and purpose of compromising operations of the victim's machine or device is said to be as Malware (Malicious Software). With the help of malicious software the attacked can able to disrupt the operations which are running normally on the computer over the network connectivity [1]. Malwares reaches the victims over the multiple network devices or over the multimedia networks, once entered in to the victim's networks malwares looks for the vulnerabilities. After finding the vulnerable port or vulnerable program it infects the victim's network and victim's computer. The antivirus software's which has been installed to protect the computer operated within the host machine based on signature based method, this method failed to detect the new and unknown malwares for example ransom wares.

One of the newest defence tools used to classify behaviour-based malware is computerised malware analysis systems (or sandboxes) [2, 3]. Sandbox techniques provide the isolated environment to execute the malware, by using the sandbox one can safely execute malwares and it provides the run time actions of the malware. Sandbox techniques is also said to be as automated technique. Instead manually execute and monitor the action of each and every malware individually, using sandbox is primary component for automated approach of malware identification. Majority of the sandboxes follow the actions of a user mode method at the device call interface. Process which interconnects the operating system and computer to perform certain operations at the level of user is said to be as system calls. These tasks include reading file information, passing packets across the network, and logging registry entries. Even more informative information can be obtained by looking more into the design of a method.

In this article, we proposed a concept of extraction the system calls from the JSON file, the report which was generated by the cuckoo sandbox when executed the malware in an isolated environment. MIST (Malware instruction set) is implemented in-order to extract the system calls from JSON file. Subsequent to separating system calls, such system calls are utilized to create N-Gram succession with the predefined term of N esteems, for example 2, 3, 4, and afterward the Obtained Gained Information(GI) choice strategy is utilized to decide the score for every N-Gram. The top N-Gram esteems are then picked based on the GI score and the characterization system performed.

Similar to this, the remaining portion of the document is numbered. In section 2, we research the context and portrayal of the MIST instruction. Throughout section 3, we look at past studies to classify malicious executable. In section 4, we will clarify our suggested solution. Experimental findings are discussed in section 5. Finally, section 6 draws inferences.

2. Background

The essential capacity of the malware recognition system is to distinguish both perceived and obscure malware and to keep up the security of the gadget while carrying out its responsibility. Malware can be inspected in two different ways, for example Understanding and translation of the Code of Ethics. All in all, the investigation of the code is finished by statically obtaining a full depiction of the program. The significant drawback of the code creation methodology is that control strategies, for example, double packers, polymorphism, and hostile to investigate techniques regularly disable it. Malware behaviour is tracked when operating on the host machine by behaviour detection. Examination of malware based on activity is an important way to monitor the activities of the malware, as the activity report [3] is used in some modern surveillance techniques. In general, in an isolated environment, behavioural malware detection programme runs a malware sample to obtain particular device level behaviours via the detection and recording of malware-invoked system calls. In the report portion, a summary of the malware sample 's behaviour is tabulated. A textual or XML-based operation report is generated by monitoring suites, which contains the malware's device-level operations, including system call information. A human observer can efficiently test textual or XML-based formats because, due to unfavourable effects on the programme's runtime, they are not suitable for more automated processing.

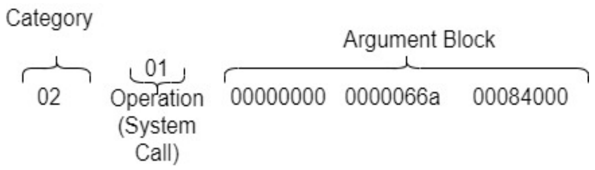


Figure. 1 System call representation in MIST format

XML formats are insufficient to follow coherent norms of behaviour. Due to the addition, which can increase the size of the analysis, text representations are complex, unlike XML. Contrasted with printed and XML-based organizations, MIST is utilized to record all machine-level activities in which Systems call contentions are composed (Fig. 1).

The first field division represents the system call area, and a specific system call is seen in the second field section. In the MIST remark, the kind of System arguments blocks and its size rely upon the particular framework demand. MIST portrayal is an intuitive procedure utilized by AI calculations for the viable and dependable distinguishing proof of malware activity [4, 5].

3. Related work

Many dynamic malware sandbox methods have been suggested in the literature, using sandbox technologies to perform dynamic malware analysis [6]. Willems et al. [7] likewise made an open source technique called CWSandbox, which empowers a malware test to be run either in a local domain or in a mimicked Windows condition. The hook functions of the analytics section monitor the calls to the API. Another complex malware detection programme that analyses the effective path of malware execution, such as the current stealthy rootkit kernel, is DRAKVUF [8]. Moreover, DRAKVUF successfully addresses issues with the discovery of capture systems calls by other sandbox systems [7]. On the other hand, sandbox- centered virtualization strategies [2, 9] assume a significant part in understanding the risky reality of the working framework set off by the styles and activities of existing malware variations.

Cuckoo [3] is another malware analytics application that delivers a complete Windows executable file activity report when run in a remote environment. In a virtualized environment, Cuckoo will analyse several malicious records (running, server, etc.) and malicious web pages. Cuckoo can follow API calls and the overall conduct of the info document and can rapidly be converted into the current framework. The actual use of the sandbox system [10, 11] is adequate to provide the technical report with the technical actions of the data in an executable file. However, an exhaustive manual examination involves a precise malware evaluation based on a sandbox database. However, the sandbox also includes a report on the monitoring system for benign executable files. For these instances it is a difficult activity to precisely distinguish real malware behaviours from other innocuous executable programs. As an unstructured type, the sandbox report is available to reliably extract real semant details (e. g. device call). Writers Ricket et al. [4] also sought, on the basis of the programme's call chain, to classify efficiently malware. For various inputs [12-14] the got systems calls grouping requested by Usually, the extraction technique for N-Grams and it is utilized. In other works, Tesauro et al. [15] have used N-Grams as a malicious detection function. Most common classes picked N-Grams in both the directories of benign and malware. The N-Grams are superior in this work which requires a broader variety of characteristics [12]. Recent studies have shown that the best outcomes for classifying a malicious executable in benign executable files are possible via the GI-based selection.

Algorithms for machine learning are seen as a possible way to correctly identify malware from stable executable files. Malicious workable classification algorithm is designed that exists in the wild by encoding N-Grams [16]. A computer-based malware analysis method was proposed to translate the sandbox-generated report to MIST format with machine learning [17, 18] to recognise unchanged malware with identical behaviour [19].

The analysis of malware first requires detection. There are majorly three methods for this - pattern matching, static analysis and dynamic analysis. Even though static analysis is a big improvement when compared with pattern matching, it is still found wanting while dealing with malware that have self-modifying code [20]. Much research has been done in the past for classification of malware based on dynamic and behavioural analysis. A comprehensive analysis using static and dynamic methods for detecting malware was proposed in [21] which successfully classified malware executable using the benefits of both static and dynamic analyses. The efficiency and the classification result saw great improvement using the integrated method, as compared with the standalone dynamic and static methods.

Dynamic analyses exploit shared patterns for classification of malware. Rieck, Holz, Willems et al learn and discriminate malware behaviour proposing a new method in [17]. This method takes three stages to complete: malware behaviour monitoring in a sandbox environment; malware classifier training with learning algorithms; and behaviour models ranking for classification. The file is classified as infected depending on the observed behavior of the file in the supervised and monitored environment. This is common practice for behavioural analyses and classification. Different features evaluate classifier performance [22, 23]. Rieck, Trinius et al [24] introduce a framework which allows automatically recognizing new and unfamiliar malware classes with identical or comparable behaviour for the automatic analysis of malware behaviour using machine learning. They suggest an incremental method for behaviour analysis using clustering and classification.

Another way for malware classification is extracting malicious API sequences. This is done using voting expert’s algorithm over API calls, or by n-gram searching over API call segments. An n-gram classification algorithm was introduced [25] where it classifies the malware instance using its n-gram features apropos to its family. This methodology may show acceptable performance in practice. Sornil, Liangboonprakong [23] propose a feature set constructed from n-gram sequential patterns for an effectual classification of malware families. The features are extracted by deconstructing files.

In [24], Rieck et al propose a special representation of behaviour denoted as malware instruction set (MIST) inspired from instruction sets used in processor design. This is to optimize processing reports for the dynamic analysis. The problem addressed was the complexity of otherwise used text-based representation. The exponential size upsurge of the report affects run-time of analysis algorithms, making it quite unfavourable. Moreover, MIST being in hexadecimal notation makes it an easier target for feature selection and engineering during machine learning and classification.

Cuckoo sandbox was used in this work for capturing the system level behaviours of the malware. Those system level behaviours are said to be as system calls are the process that are generated by the executable files and that are reported by the sandbox in this work. In this work GI function discovery is used to choose the correct functionality of FFV (Final feature Vector).

4. Proposed work

Based on the analysis done in the above section, it has been concluded that combining the all techniques together we may get the better result i.e by combining the System Calls and N-Gram features.

A heuristic methodology called N-Grams audit is utilized to isolate malware documents from secured records dependent on the grouping of System calls. It

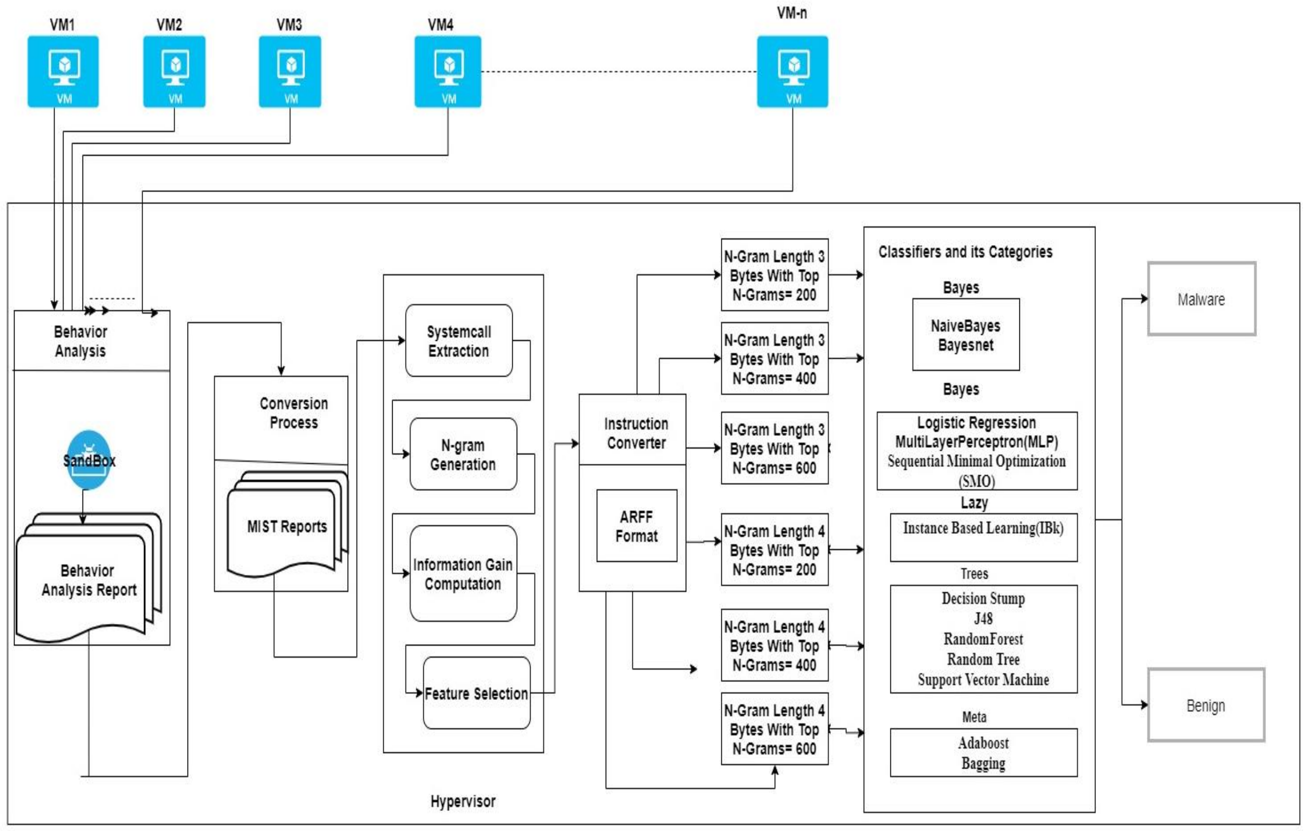

VM1
VM2
VM3
VM4
VM-n
VM
Behavior Analysis
SandBox
Behavior Analysis Report
Conversion Process
MIST Reports
Systemcall Extraction
N-gram Generation
Information Gain Computation
Feature Selection
Instruction Converter
ARFF Format
N-Gram Length 3 Bytes With Top N-Grams= 200
N-Gram Length 3 Bytes With Top N-Grams= 400
N-Gram Length 3 Bytes With Top N-Grams= 600
N-Gram Length 4 Bytes With Top N-Grams= 200
N-Gram Length 4 Bytes With Top N-Grams= 400
N-Gram Length 4 Bytes With Top N-Grams= 600
Classifiers and its Categories
Bayes
NaiveBayes Bayesnet
Bayes
Logistic Regression MultiLayerPerceptron(MLP) Sequential Minimal Optimization (SMO)
Lazy
Instance Based Learning(IBk)
Trees
Decision Stump J48 RandomForest Random Tree Support Vector Machine
Meta
Adaboost Bagging
Malware
Benign
Hypervisor

Figure. 2 System design of the work proposed

utilises the GI method in which the GI size of and N grams is calculated and, on the basis of the highest GI score, extracts the highest N graphs to produce the FFV necessary for classification, Fig. 2 explains the assessment proposed. Were initially malwares are executed in the VM's and its behavioural reports forwarded to the Cuckoo sandbox as JSON file and those JSON files will be converted to MIST files, from the MIST file system call extraction taken place. After the extraction of system call from MIST file, N-Gram Generation and Feature selection has been performed based on the GI score. By next those information has been converted to ARFF format in order to perform the analysis with the various chosen classifiers. Based on the GI score , Top N-Gram of 200,400,600 are chosen with the N-Gram with the length of 3 and 4. The classifiers chosen were Adaboost Classifier – ABC, Decision Stump – DS, Random Forest Algorithm – RF, Naïve Bayesian, Classifier – NBC, Logistic Regression Classifier – LRC, Random tree classifier – RTC, J48 Decision Tree – J48C, Bagging(Bootstrap Aggregating) – BBA, IBk (Instance Based earning) – IBK, Sequential Minimal, Optimization – SMO, Bayes Net – BN, Multi-Layer Perceptron –MLP, Support Vector machine – SVM, J48Craft – J48C.

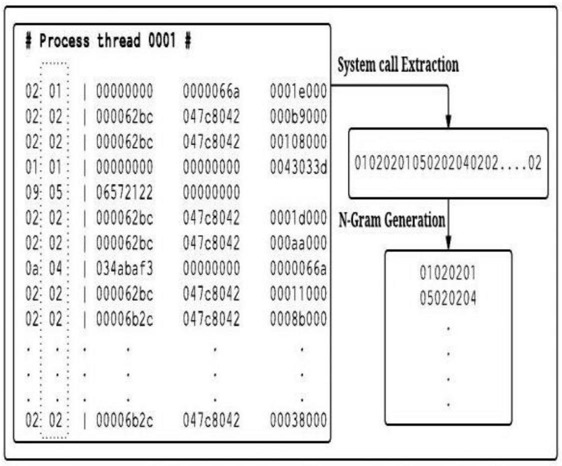
4.1 Behavior analysis

As the cuckoo sandbox is a different hypervisor entity, it analyses malware behaviour on VMs to get a conduct review report for the activity in JavaScript Object Notation format (JSON).

4.2 Conversion of JSON to MIST process

When getting to MIST, research reports created in JSON design are pre-handled as a standard arrangement that utilizes a decreased document size and diminishes preparing time. Since our strategy is limited to following oversaw System calls, we are worried about MIST records (System call as appeared in Fig. 1) region of activity for producing N-Grams (4 bytes) documents as appeared in Fig. 3. We follow the steps below to create the N-Gram files: (i) Extraction of system calls (ii) Generating N-Gram (iii) N-Gram Grading (iv) Double delete (Duplicate Removal)

We pick just the cycle field in the principal phase of system call extraction, for example as appeared in Fig. 4 system calls of all amiable examples report MIST records (1 to n) and system calls of all the malware MIST documents (1 to n), Because we have



```
# Process thread 0001 #

02: 01 | 00000000   0000066a   0001e000
02: 02 | 000062bc   047c8042   000b9000
02: 02 | 000062bc   047c8042   00108000
01: 01 | 00000000   00000000   0043033d
09: 05 | 06572122   00000000
02: 02 | 000062bc   047c8042   0001d000
02: 02 | 000062bc   047c8042   000aa000
0a: 04 | 034abaf3   00000000   0000066a
02: 02 | 000062bc   047c8042   00011000
02: 02 | 00006b2c   047c8042   0008b000
 .   . .    .          .          .
 .   . .    .          .          .
 .   . .    .          .          .
02: 02 | 00006b2c   047c8042   00038000
```

Figure. 3 Snippet of N-Gram extraction

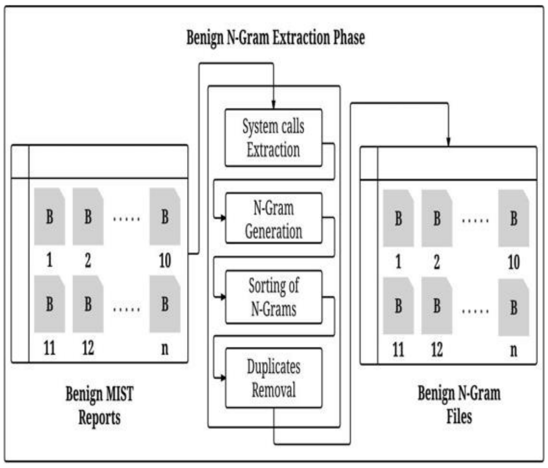
a framework level chronicle all things considered. The determined cycle fields are kept in a document and grouped to shape variable length N-Grams in arrangement, for example N Gram of 2, 3, 4, etc.

The longer the N-Grams size, the more the characteristics are portrayed. As seen in Fig 3., an extraction snippet. When generating the N-Grams, we likewise grouped N-Grams of four bytes in the second period of the age cycle. The created N-Grams are arranged in plummeting request in the third means to get the most noteworthy request N-Grams arrangement. In the fourth step, if explicit N-Grams are to be gathered, the copies ought to be extricated after the arranging cycle. The extraordinary N-Grams can be utilized for a superior scope of capacities and hence have better gathering.

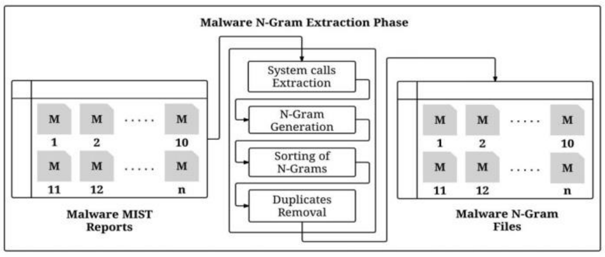
The following definition is a requirement for a selective approach to the feature, because it cannot be achieved without N-Gram being generated. N-Grams records for Benign documents B1, B2, B3...Bn and for Malware records M1, M2, ..., Mn are produced and Union activities are performed for both the Malware and Benign documents B1UB2..UBN and Malware records M1UM2..UMn are acted in this stage.

Duplicates must be eliminated in order to get the unique N-Gram of Malware and Non-Malware files by sorting the file by N-gram value from lower to higher after the union N-Gram files of Non-malware. Occurrences of each unique malware, non-malware are observed and that could be tabulated in the frequency table of N-Gram.

The contingency table will at that point be set up as per the qualities found in the N-Gram recurrence table for the amiable class and the malware classification as found in Fig. 5. For the calculation of Value of Information [10], By using the below mentioned equation (refer with: Eq. (1)) the Gain Information is calculated for the features generated

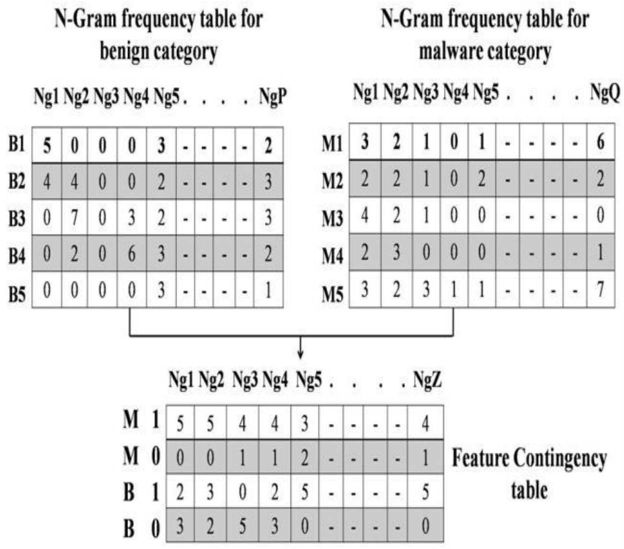


(a)



(b)

Figure. 4 Extraction process of system call: (a) N-Gram files generation steps for benign inputs and (b) N-Gram files generation steps for malware inputs



N-Gram frequency table for benign category

	Ng1	Ng2	Ng3	Ng4	Ng5	.	.	.	.	NgP
B1	5	0	0	0	3	-	-	-	-	2
B2	4	4	0	0	2	-	-	-	-	3
B3	0	7	0	3	2	-	-	-	-	3
B4	0	2	0	6	3	-	-	-	-	2
B5	0	0	0	0	3	-	-	-	-	1

N-Gram frequency table for malware category

	Ng1	Ng2	Ng3	Ng4	Ng5	.	.	.	.	NgQ
M1	3	2	1	0	1	-	-	-	-	6
M2	2	2	1	0	2	-	-	-	-	2
M3	4	2	1	0	0	-	-	-	-	0
M4	2	3	0	0	0	-	-	-	-	1
M5	3	2	3	1	1	-	-	-	-	7

Feature Contingency table

		Ng1	Ng2	Ng3	Ng4	Ng5	.	.	.	.	NgZ
M	1	5	5	4	4	3	-	-	-	-	4
M	0	0	0	1	1	2	-	-	-	-	1
B	1	2	3	0	2	5	-	-	-	-	5
B	0	3	2	5	3	0	-	-	-	-	0

Figure. 5 N-Gram frequency table with feature contingency table for the classes of benign and malware

and stored in the contingency table.

$$\mathrm{GI(N-Gram)} = \sum_{\mathrm{Val\_\ N-Gram} \in \{0,1\}} \sum_{\mathrm{T} \in \{T_i\}} \mathrm{A(Val\_\ N-Gram, T)} \log \frac{\mathrm{A(Val\_\ N-Gram,T)}}{\mathrm{A(Val\_\ N-Gram),A(T)}} \quad (1)$$

In above equation (refer with: Eq. (1)), T- is the type of file i.e. Benign(Normal/Non-malware) or Malware, N – Gram Value is represented by Val _ N – Gram, if

Val _ N – Gram =1 means that N-Gram is present in either the Benign(Normal/Non-malware) or Malware, otherwise the value of N – Gram is represented by Val _ N - Gram = 0. A (Val _ N – Gram, T) is the total number of N – Gram items in the T where in which the value of N – Gram takes on by Val _ N – Gram. A (Val _ N – Gram) is the total number of Benign(Normal/Non-malware) or malware items present in the dataset which used for training present in the N – Gram i.e. the Val _ N – Gram. A (T) is the total number of input file type which belongs either Benign(Normal/Non-malware) or malware. Based on the GI score, the N-Grams are sorted in a non-increased order and the highest number of N-Grams is identified as the best characteristics for classification purposes.

4.3 Converting instructions from MIST to ARFF

The Instruction converter measure is for changing over the MIST record System call data to the Attribute-Relation File Format (ARFF). ARFF is an ASCII text record, in which assortments of characteristics are shared and spoken to in list. This cycles in expected to give contribution to WEKA classifier apparatus or even we utilize any programming language for classification.

5. Experimental result

For our experimental work we used the MIST data consist of benign files (3000 files) and Malware files (3100 files). This consist of different categories such as 1000 number of each Swizzor, Basun and Autoit, 100 Number of Kelihos files for making the Malware MIST file which has been downloaded from public websites. As mentioned above, we extracted 2 bytes, 3 bytes, and 4 bytes of N-Grams of different sizes to determine the N-Gram value to obtain the highest detection efficiency. In previous works researchers were done analysis based on N-Gram or without N-Gram. Here we used the GI score to select features and based on that the better N-Gram Value and N-Gram Bytes are selected for analysis.

Moreover, the best attributes for various N-Gram lengths were drawn at every K esteem. The best highlights were pre-handled to get ready ARFF documents for the chose N-Grams by means of the Instruction Converter. The ARFF documents have been submitted for order on the WEKA technique. Enormous scopes of investigations were directed to figure out which classifier with a low False Positive Rate (FPR) accomplished best malware location productivity. We assessed the performance of several algorithms of classification specified in the WEKA method.

Table 1. Classification results obtained without removing duplicate attributes and without applying N-Gram

Algorithm	TP Rate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
NBC	0.89	0.005	0.89	0.89	0.84	0.89	0.779
DS	0.87	0.028	0.86	0.86	0.86	0.89	0.776
RFA	0.84	0.056	0.84	0.84	0.84	0.88	0.841
IBk	0.83	0.071	0.82	0.82	0.82	0.81	0.478
RTC	0.83	0.067	0.82	0.82	0.82	0.82	0.804
BSA	0.8	0.011	0.8	0.8	0.7	0.84	0.826
LRC	0.8	0.1	0.8	0.8	0.7	0.81	0.774
MLP	0.8	0.102	0.8	0.8	0.7	0.85	0.804
J48C	0.8	0.18	0.79	0.79	0.79	0.81	0.826
ABC	0.79	0.13	0.79	0.78	0.78	0.83	0.73
J48 DT	0.77	0.126	0.77	0.77	0.77	0.81	0.774
BN	0.77	0.123	0.77	0.854	0.77	0.82	0.804
SMO	0.73	0.126	0.76	0.82	0.72	0.7	0.641
SVM	0.74	0.24	0.6	0.701	0.61	0.6	0.796

The result values of TPR (True Positive Rate), Precision, FPR (False Positive Rate), F-Measure, Recall which obtained for selected classifiers are tabulated and shown in the following tables Table 1, Table 2, Table 3, Table 4, Table 5, Table 6, Table 7, Table 8, and Table 9.

Initially we performed two experiments i.e the results which we obtain without removing duplicate attributes and with duplicate attributes, also we evaluated the classifiers by performing the feature selection. The result obtained before and after duplicate removal, feature selection operation gave the slight variation in the accuracy.

The dataset produced using MIST has some attributes that had completely zero values for all the malwares, so need to perform analysis after removing those attributes. Analysis report that obtained after removing the duplicate and null value attributes has shown in Tables 2. The accuracy values obtained for various classifiers have shown in Table 2, in which random forest algorithm provides the highest accuracy of 85%, whereas before with null value and duplicate attributes, the same random forest algorithm has provided 84% of accuracy and now it has been increased by 1%.

Table 2. Table 2: Classification results obtained without removing duplicate attributes and without applying N-Gram

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.78	0.23	0.79	0.78	0.78	0.84	0.79
DS	0.79	0.23	0.8	0.79	0.79	0.75	0.7
RFA	0.84	0.16	0.84	0.84	0.84	0.87	0.85
NBC	0.57	0.44	0.58	0.47	0.45	0.73	0.48
LRC	0.8	0.2	0.8	0.8	0.8	0.81	0.8
RTC	0.76	0.23	0.76	0.76	0.76	0.76	0.77
J48 DT	0.77	0.23	0.77	0.77	0.77	0.81	0.77
BBA	0.8	0.21	0.8	0.8	0.8	0.84	0.8
IBk	0.82	0.18	0.82	0.82	0.82	0.81	0.83
SMO	0.73	0.33	0.76	0.73	0.72	0.702	0.73
BN	0.77	0.33	0.77	0.77	0.77	0.82	0.77
MLP	0.76	0.25	0.76	0.76	0.76	0.85	0.8
SVM	0.63	0.25	0.71	0.63	0.59	0.59	0.83
J48C	0.79	0.21	0.79	0.79	0.79	0.81	0.73

Table 3. Classification results obtained after feature selection

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.78	0.23	0.79	0.88	0.78	0.83	0.789
DS	0.79	0.22	0.8	0.89	0.79	0.75	0.796
RFA	0.84	0.16	0.84	0.94	0.84	0.88	0.841
NBC	0.67	0.34	0.58	0.57	0.45	0.73	0.478
LRC	0.8	0.2	0.8	0.9	0.8	0.81	0.804
RTC	0.82	0.17	0.82	0.92	0.82	0.82	0.826
J48 DT	0.77	0.22	0.77	0.87	0.77	0.81	0.774
BBA	0.8	0.2	0.8	0.9	0.8	0.84	0.804
IBk	0.82	0.18	0.82	0.92	0.82	0.81	0.826
SMO	0.73	0.32	0.76	0.83	0.72	0.7	0.73
BN	0.77	0.22	0.77	0.87	0.77	0.82	0.774
MLP	0.8	0.2	0.8	0.9	0.8	0.85	0.804
SVM	0.64	0.44	0.7	0.74	0.61	0.6	0.826
J48C	0.79	0.2	0.79	0.89	0.79	0.81	0.73

Table 4. N-Gram Classification Results Length of 3 bytes with Top N-Grams=200

Algorithm	TP Rate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
NBC	0.99	0.006	0.99	0.99	0.94	0.99	88.9
DS	0.97	0.038	0.96	0.96	0.96	0.99	89.6
RFA	0.94	0.066	0.94	0.94	0.94	0.98	**94.1**
IBk	0.93	0.081	0.92	0.92	0.92	0.91	57.8
RTC	0.93	0.077	0.92	0.92	0.92	0.92	90.4
BSA	0.9	0.111	0.9	0.9	0.9	0.94	92.6
LRC	0.9	0.102	0.9	0.9	0.9	0.91	87.4
MLP	0.9	0.102	0.9	0.9	0.9	0.95	90.4
J48C	0.9	0.108	0.89	0.89	0.89	0.91	92.6
ABC	0.89	0.13	0.89	0.88	0.88	0.93	83
J48 DT	0.87	0.129	0.87	0.87	0.87	0.91	87.4
BN	0.87	0.129	0.87	0.874	0.87	0.92	90.4
SMO	0.83	0.226	0.86	0.83	0.82	0.8	74.1
SVM	0.74	0.34	0.8	0.741	0.71	0.7	89.6

Table 5. N-Gram Classification Results Length of 3 bytes with Top N-Grams=400

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.88	0.13	0.89	0.88	0.88	0.94	89
DS	0.89	0.13	0.9	0.89	0.89	0.85	90
RFA	0.94	0.06	0.94	0.94	0.94	0.97	**95**
NBC	0.57	0.34	0.68	0.57	0.55	0.83	58
LRC	0.9	0.1	0.9	0.9	0.9	0.91	90
RTC	0.86	0.13	0.86	0.86	0.86	0.86	87
J48 DT	0.87	0.13	0.87	0.87	0.87	0.91	87
BBA	0.9	0.11	0.9	0.9	0.9	0.94	90
IBk	0.92	0.08	0.92	0.92	0.92	0.91	93
SMO	0.83	0.23	0.86	0.83	0.82	0.8	83
BN	0.87	0.13	0.87	0.87	0.87	0.92	87
MLP	0.86	0.15	0.86	0.86	0.86	0.95	90
SVM	0.73	0.35	0.81	0.73	0.69	0.69	93
J48C	0.89	0.11	0.89	0.89	0.89	0.91	83

As previously seen, the analysis report was obtained without removing duplicate null value attributes and with null value attributes; Tables 3 show the analysis report after performing feature selection for the MIST dataset after removing null value attributes. The accuracy values obtained for various classifiers have shown in Table 3, in which Random Forest Algorithm provides the highest accuracy of 84%, next to that random tree classifier provides 82% of accuracy.

Other than evaluating classifiers before and after removing duplicates, feature selection, we performed two experiments: In the primary investigation, we considered three bytes of N-Gram to pick the top N-Grams dependent on the greatest GI Value. The top N-Grams were picked as far as 200, 400, 600 and the best precision was 94.1 % for 200 N-Grams, for N-Gram Value of 400 got exactness of 95 %, and exactness of 94.1 %t for 600 N-Grams for the Random Forest Classifier. The Random Forest classifier had recorded the highest TPR. It was found that the Random Forest classification obtained the best results.

By refereeing the Table. 4 It should be remembered that the highest precision was given by the RFA (Random Forest Algorithm). The second highest precision is given by Random Tree. Table 5: Provides the N-Gram Length 3 byte classification results with Top N-Gram=400, thereby providing the highest TPR and lowest FPRR according to all the classifiers, Random forest provided the highest TPR and lowest FPR.

Table 6. N-Gram classification results length of 3 bytes with top N-Grams=600

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.88	0.13	0.89	0.88	0.88	0.93	88.9
DS	0.89	0.12	0.9	0.89	0.89	0.85	89.6
RFA	0.94	0.06	0.94	0.94	0.94	0.98	**94.1**
NBC	0.57	0.34	0.68	0.57	0.55	0.83	57.8
LRC	0.9	0.1	0.9	0.9	0.9	0.91	90.4
RTC	0.92	0.07	0.92	0.92	0.92	0.92	92.6
J48 DT	0.87	0.12	0.87	0.87	0.87	0.91	87.4
BBA	0.9	0.11	0.9	0.9	0.9	0.94	90.4
IBk	0.92	0.08	0.92	0.92	0.92	0.91	92.6
SMO	0.83	0.22	0.86	0.83	0.82	0.8	83
BN	0.87	0.12	0.87	0.87	0.87	0.92	87.4
MLP	0.9	0.1	0.9	0.9	0.9	0.95	90.4
SVM	0.74	0.34	0.8	0.74	0.71	0.7	92.6
J48C	0.89	0.1	0.89	0.89	0.89	0.91	83

Table 7. N-Gram Classification Results Length of 4 bytes with Top N-Grams=200

Algorithm	TP Rate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
NBC	0.99	0.006	0.99	0.99	0.94	0.99	88.9
DS	0.97	0.038	0.96	0.96	0.96	0.99	89.6
RFA	0.94	0.066	0.94	0.94	0.94	0.98	**94.1**
IBk	0.93	0.081	0.92	0.92	0.92	0.91	57.8
RTC	0.93	0.077	0.92	0.92	0.92	0.92	90.4
BSA	0.9	0.111	0.9	0.9	0.9	0.94	92.6
LRC	0.9	0.102	0.9	0.9	0.9	0.91	87.4
MLP	0.9	0.102	0.9	0.9	0.9	0.95	90.4
J48C	0.9	0.108	0.89	0.89	0.89	0.91	92.6
ABC	0.89	0.13	0.89	0.88	0.88	0.93	83
J48 DT	0.87	0.129	0.87	0.87	0.87	0.91	87.4
BN	0.87	0.129	0.87	0.874	0.87	0.92	90.4
SMO	0.83	0.226	0.86	0.83	0.82	0.8	74.1
SVM	0.74	0.34	0.8	0.741	0.71	0.7	89.6

Table 8. N-Gram Classification Results Length of 4 bytes with Top N-Grams=400

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.88	0.13	0.89	0.88	0.88	0.94	89
DS	0.89	0.13	0.9	0.89	0.89	0.85	90
RFA	0.94	0.06	0.94	0.94	0.94	0.97	**93.5**
NBC	0.57	0.34	0.68	0.57	0.55	0.83	58
LRC	0.9	0.1	0.9	0.9	0.9	0.91	90
RTC	0.86	0.13	0.86	0.86	0.86	0.86	87
J48 DT	0.87	0.13	0.87	0.87	0.87	0.91	87
BBA	0.9	0.11	0.9	0.9	0.9	0.94	90
IBk	0.92	0.08	0.92	0.92	0.92	0.91	93
SMO	0.83	0.23	0.86	0.83	0.82	0.8	83
BN	0.87	0.13	0.87	0.87	0.87	0.92	87
MLP	0.86	0.15	0.86	0.86	0.86	0.95	90
SVM	0.73	0.35	0.81	0.73	0.69	0.69	93
J48C	0.89	0.11	0.89	0.89	0.89	0.91	83

Comparing the results obtained using N-Gram length of 3 Bytes with Top N-Grams value of 200,400,600. Random Forest Algorithm Provided the Better Result of 95% accuracy with N-Gram of

Length 3 Byte and the Top N-Gram 400.Whereas the Top N-Gram 200 and 600 provides the accuracy of 94.1%.

Similarly, N-Gram bytes length of four was evaluated in the second experiment, and the findings for best precision were 94.1 percent for N-Gram Value 200, 93.5 percent for N-Grams Top value 400, and 94.1 percent for the RandomForest classifier for 600 N-Grams. We may infer that the Random Forest classifier was the strongest and guaranteed a decent classification for all lengths of N-Gram three and four.

Even we changed the N-Gram Length to 2 bytes, 4 bytes and Top N-Gram Values of 200, 400, 600. The Random Forest Provides the similar result for all the values i.e. 94.1%, only for the N-Gram Length of 3 Bytes with Top value of N-Gram 400 We are getting the 95% of accuracy. So When comparing with all we can use this as the constant value for our Experiments.

The random forest algorithm gave us the highest accuracy while training, and hence it was the chosen classifier for the proposed system. Each of the above modules when cascaded together resulted in a system.

With an accuracy of 95%, the proposed system is able to do a binary classification and predict if a file is malicious or non-malicious. The type of analysis done is dynamic. The test file is run in real time under the sandbox and its feature vector is created using all the cascaded modules, which is further sent to our classifier for prediction.

The overall pipeline is slow and time consuming compared with general systems that are classified based on the static analysis of files, but the results are quite accurate. The model can be further enhanced by training it to do multiclass classification and predict the class of malware as well.

Table 9. N-Gram classification results length of 4 bytes with top N-Grams=600

Algorithm	TPRate	FP RATE	PRECISON	RECALL	F-MEASURE	ROC AREA	Accuracy
ABC	0.88	0.13	0.89	0.88	0.88	0.93	88.9
DS	0.89	0.12	0.9	0.89	0.89	0.85	89.6
RFA	0.94	0.06	0.94	0.94	0.94	0.98	92.7
NBC	0.57	0.34	0.68	0.57	0.55	0.83	57.8
LRC	0.9	0.1	0.9	0.9	0.9	0.91	90.4
RTC	0.92	0.07	0.92	0.92	0.92	0.92	92.6
J48 DT	0.87	0.12	0.87	0.87	0.87	0.91	87.4
BBA	0.9	0.11	0.9	0.9	0.9	0.94	90.4
IBk	0.92	0.08	0.92	0.92	0.92	0.91	92.6
SMO	0.83	0.22	0.86	0.83	0.82	0.8	83
BN	0.87	0.12	0.87	0.87	0.87	0.92	87.4
MLP	0.9	0.1	0.9	0.9	0.9	0.95	90.4
SVM	0.74	0.34	0.8	0.74	0.71	0.7	92.6
J48C	0.89	0.1	0.89	0.89	0.89	0.91	83

6. Uniqueness of this work

Here malware examination has been finished utilizing the different Machine Learning algorithms. Estimated the presentation dependent on three measure's True positives (information focuses those are in reality obvious and are grouped valid), False Positive (information focuses those are in reality bogus however are ordered valid) lastly looked at accuracy of changed models that have been examined.

By notice to the test results, grouping of malware behavioural activities can be a helpful technique in building up a behavioural antivirus. Contrasted the precision rate and different Classifiers, Random Forest Algorithm Provides the better accuracy.

7. Conclusion

Behavioural examination of windows executable records are broke down with the assistance of System calls summoned by the info documents during the hour of execution in the virtual machines, the gathered system calls are pieced into the N-Gram by after the transformation of JSON report to MIST. In order to choose the best features Gained Information Score (GI score) feature selection technique was used and FFV required by the classifier are prepared with the use of GI feature selection technique. Experiment is analysed with various chosen classifiers. From the findings results it was observed Random forest was the classifier which provides the best result among the classifiers chosen for our experiment work. Random forest has provided 95% of accuracy with highest precision and TPR, with lowest FPR. Our future work aim is to construct a model which is capable of calculate the GI – Score for larger N-Grams datasets.

Conflicts of Interest

The authors confirm that this publication has no established conflicts of interest and that there has been no substantial financial funding for this work that may have influenced its result.

Author Contributions (Mandatory)

Conceptualization, Udayakumar Nandagopal and Subbulakshmi Thirumalaivelu; methodology, Udayakumar Nandagopal and Subbulakshmi Thirumalaivelu; software, Udayakumar Nandagopal and Subbulakshmi Thirumalaivelu; validation, Udayakumar Nandagopal and Subbulakshmi Thirumalaivelu; formal analysis, Udayakumar Nandagopal; investigation, Udayakumar Nandagopal; resources, Udayakumar Nandagopal; data curation, Udayakumar Nandagopal; writing-original draft preparation, Udayakuma Nandagopal; writing-review and editing, Udayakumar Nandagopal and Subbulakshmi Thirumalaivelu; visualization, Udayakumar Nandagopal; supervision, Subbulakshmi Thirumalaivelu.

References

[1] Shabtai, R. Moskovitch, Y. Elovici, and C. Glezer, "Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-of-the-art Survey", *Information Security Technical Report*, Vol. 14, No. 1, pp. 16-29, 2009.

[2] T. Mandl, U. Bayer, and F. Nentwich, "Anubis-analyzing Unknown Binaries the Automatic Way", In: *Proc. of Virus Bulletin Conf.*, Geneva, Switzerland, 2009

[3] Cuckoo. 2020. Automated Malware Analysis. [online] Available at: <https:// cuckoosandbox.org> [Accessed 5 October 2020].

[4] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic Analysis of Malware Behavior using Machine Learning", *Journal of Computer Security*, Vol. 19, No. 4, pp. 639-668, 2011.

[5] N. Udayakumar, V. J. Saglani, A. V. Cupta, and T. Subbulakshmi, "Malware classification using Machine Learning Algorithms", In: *Proc. of the 2nd International Conf. on Trends in Electronics and Informatics*, Tirunelveli, India, pp. 1-9, 2018.

[6] N. Udayakumar, T. Subbulakshmi, Ayush Mishra, Shivang Mishra, and Puneet Jain, "Malware Category Prediction using KNN Classifier and SVM Classifiers", *International Journal of Mechanical Engineering and Technology*, Vol. 10, No. 2, pp. 787-797, 2019.

[7] Willems, T. Holz, and F. Freiling, "Toward Automated Dynamic Malware Analysis using Cwsandbox", *IEEE Security & Privacy*, Vol.5, No.2, pp.32-39, 2007.

[8] T. K. Lengyel, S. Maresca, B. D. Payne, G. D. Webster, S. Vogl, and A. Kiayias, "Scalability, Fidelity and Stealth in the DRAKVUF Dynamic Malware Analysis system", In: *Proc. of the 30th Annual Computer Security Applications Conf.*, pp. 386-395, 2014.

[9] M. Neugschwandtner, C. Platzer, P.M. Comparetti, and U. Bayer, "Danubis–dynamic Device Driver Analysis based on Virtual Machine Introspection", In: *Proc. of International Conf. On Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, Berlin, Heidelberg, pp. 41-60, 2010.

[10] Y. Qiao, Y. Yang, J. He, C. Tang, and Z. Liu, "CBM: Free, Automatic Malware Analysis Framework using API Call Sequences", In: *Knowledge Engineering and Management*, Springer, Berlin, Heidelberg, pp. 225-236, 2014.

[11] J. Shi, Y. Yang, C. Li, and X. Wang, "Spems: A Stealthy and Practical Execution Monitoring System based on Vmi", In: *Proc. of International Conf. on Cloud Computing and Security*, Springer, Cham, pp. 380-389, 2015.

[12] D. K. S. Reddy and A. K. Pujari, "N-gram Analysis for Computer Virus Detection", *Journal in Computer Virology*, Vol. 2 No. 2, pp. 231-239, 2006.

[13] S. Jain and Y. K. Meena, "Byte Level n–gram Analysis for Malware Detection", In: *Proc. of International Conf. on Information Processing*, Springer, Berlin, Heidelberg, pp. 51-59. 2011.

[14] H. Parvin, B. Minaei, H. Karshenas, and A. Beigi, A, "A New N-gram Feature Extraction-Selection Method for Malicious Code", In: *Proc. of International Conf. on Adaptive and Natural Computing Algorithms*, Springer, Berlin, Heidelberg, pp. 98-107, 2011.

[15] G. J. Tesauro, J. O. Kephart, and G. B. Sorkin, "Neural Networks for Computer Virus Recognition", *IEEE Expert*, Vol. 11, No. 4, pp. 5-6, 1996.

[16] J. Z. Kolter and M. A. Maloof, "Learning to Detect and Classify Malicious Executables in the Wild", *Journal of Machine Learning Research*, Vol.7, pp. 2721-2744, 2006.

[17] K. Rieck, T. Holz, C. Willems, P. Düssel, and P. Laskov, "Learning and Classification of Malware Behavior", In: *Proc. of International Conf. on Detection of Intrusions and Malware, and Vulnerability Assessment*, Springer, Berlin, Heidelberg, pp. 108-125, 2008.

[18] N. Udayakumar, S. Anandaselvi, and T. Subbulakshmi, "Dynamic Malware Analysis using Machine Learning Algorithm", In: *Proc. of International Conf. On Intelligent Sustainable Systems*, Palladam, India, pp. 795-800, 2017.

[19] N. Udayakumar, A. Khera, L. Suri., C. Gupta, and T. Subbulakshmi, "Bandwidth Analysis of File Transfer Protocol", In: *Proc. of*

International Conf. on Communication and Signal Processing, Chennai, India, pp. 0791-0795, 2018.

[20] P. Szor, *The Art of Computer Virus Research and Defense*. Pearson Education, NJ, 2005.

[21] P. V. Shijo and A. J. P. C. S. Salim, "Integrated Static and Dynamic Analysis for Malware Detection", *Procedia Computer Science*, Vol. 46, pp. 804-811, 2015.

[22] G. Cabau, M. Buhu, and C. P. Oprisa, "Malware Classification based on Dynamic Behavior", In: *Proc. of 18th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing,* Timisoara, Romania, pp. 315-318, 2016.

[23] O. Sornil and C. Liangboonprakong, "Malware Classification Using N-grams Sequential Pattern Features", *International Journal of Information Processing and Management*, Vol. 4, No. 5, pp. 59-67, 2013.

[24] K. Rieck, P. Trinius, C. Willems, and T. Holz, "Automatic Analysis of Malware Behavior using Machine Learning", *Journal of Computer Security*, Vol. 19, No. 4, pp. 639-668, 2011.

[25] A. Pektaş, M. Eriş, and T. Acarman, "Proposal of N-gram based Algorithm for Malware Classification", In: *Proc. of the 5th International Conf. on Emerging Security Information, Systems and Technologies*, pp. 7-13, 2011.