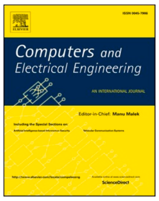
Computers and Electrical Engineering xxx (xxxx) xxx

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng



Regression coefficients as triad scale for malware detection

Saud S. Alotaibi [a, *]

[a] Department of Information Systems, College of Computer and Information Systems, Umm Al Qura University, Makkah, Saudi Arabia

ARTICLE INFO

Keywords:
Malware detection
Call sequences
Control flow graphs
Triad scale
T-test
Portable executable
API-call

ABSTRACT

The malware detection methods are classified into two categories, namely, dynamic analysis (active analysis) and static analysis (passive analysis). These methods undergo unusual obstruction, and challenges that are process complexity, limitation over detection accuracy. The static method serves to discover malicious applications using various parameters like permission analysis, signature verification. It can be regularly obfuscated. Dynamic techniques entail investigating the performance of an application by administering it in a restricted environment. The complex version of a portable executable often emerges with an intervention by hardening the dynamic analysis centric malware detection methods. The various constraints of these dynamic and static models contribute to this manuscript represents a Multi-Level Malware detection using Triad Scale (MLMTS) built on regression coefficients. The proposed method MLMTS spans into three levels, such that the first and second level performs static analysis, and the third level performs the dynamic analysis. The second and third levels of the hierarchy invoke upon the ambiguous decision of their respective predecessor level. The proposed work is based on the Machine Learning (ML) model that determines the triad scale by applying linear regression for each level of malware detection. The call sequences of the portable executable, arguments passed to these call sequences and their fallouts (resultant values) in respective order of three levels of the MLMTS method. The experimental study manifests the significance of the proposal compared to the other recent malware detection methods.

1. Introduction

The malicious software that is often termed as malware intends to infiltrate, infect, or intrude the cryptographic verification of the owner in the computer system. According to contemporary statistics [1], an average of 400 million malware models is recognized per annum. Currently, the malware family has boosted through the software modules engineered by incredible software skills [2]. The attacks can anonymize the source of the attack [2] and considerably succeeds in hacking the potential industrial structures known as the Stuxnet [3]. Anonymized sources are significant challenges to contemporary malware detection strategies. Extensive utilization of computer-networks is vulnerable to potential malware attacks. The dynamic network connectivity exposes the vulnerabilities of the corresponding network, which entertains the attackers to exploit these vulnerabilities to inject the malware into the respective system. The Intel organization has estimated the impact of malware in terms of loss of revenue as above 400 billion (USD 400 * 10^9) dollars worldwide per annum [1]. These statistics concreting the need for potential malware detect and defense mechanisms.

The malware detection by static analysis intends to determine the malware scope of the file structure. The static analysis detects malware by exploring the call sequences and byte sequences of the given portable executable (PE). The static analysis compares the call

* Corresponding author.
E-mail address: ssotaibi@uqu.edu.sa.

https://doi.org/10.1016/j.compeleceng.2020.106886
Received 20 December 2019; Received in revised form 6 October 2020; Accepted 14 October 2020

sequences and byte sequences of the given PE, which represents the malware samples [4] using the statistical methods. The contextual factors of the target domain have often influenced the other inputs such as data-flow, dependent executables, data, and control flows to compare with malware samples. These static methods can explore the overall execution path without process overhead. However, the static analysis methods of malware detection are often abandoning due to obfuscation notions of call identities, binary structures, and also denies exploring the execution path of the given PE. Dynamic analysis detects the malware by executing the given PE in a virtual environment, which aids to track the events, data flow, and in-built memory. The behavior of these data flows and events are further used to determine malware behavior. However, the dynamic analysis methods often exhibit the processing overhead as NP-hard [4]. Hence, these dynamic analysis methods have considered as the counterpart of static analysis methods, which can enhance the malware detection accuracy [4]. During the runtime, the behavioral log of the PE manifests the resources and limits to detect only malware having a behavioral flow. Hence, it might need inputs to be stimulated for exploring the overall segments of code. One of the time-consuming factors is the challenges encountered to execute the program in a controlled area within the stipulated time. Moreover, some contemporary malware categories can identify the virtual environment and hibernate the malevolent intent of the corresponding PE [5] as a countermeasure to malware detection by dynamic analysis.

However, dynamic analysis of malware detection exhibits low-level mutation strategies like obfuscation or packing of runtime, which could not influence behavioral attributes as significant benefits. Besides, malware detection by dynamic analysis can provide actual information regarding data-flow and control. Hence, dynamic models are often prioritized for implementing the static-analysis in malware identification.

Malware detection by static analysis performs without actually running the program [5]. The opcode sequences (extracted by disassembling the binary) control flow graphs extracted from the given PE are used as inputs by these static analysis measures [6].

The malware detection methods of both static and dynamic analysis categories have advantages and disadvantages. The static analysis does not implement executable programs, so its main advantage over its dynamic counterpart is free from implementation costs. However, static analysis suffers from a lack of support for filled symbols and restrictions related to sophisticated obfuscation. Compared to static analysis, dynamic analysis can effectively scan packaged malware as it unpacks during execution, and its original code shall load into the main memory. However, the consumption of time and resources is an obvious drawback since malware samples have to be analyzed successively in reality. Consequently, these shortcomings limit its adoption in business analysis applications.

According to the above-mentioned malware-detection statistics, it is observed that there is an opportunity for the research to derive new malware detection strategies. The proposed work explores the usage of machine learning techniques in malware detection.

The organization of the manuscript is as follows. Section 2 examines the related research about malware detection strategies. Section 3 contains a detailed explanation of the methods and materials of the proposed machine learning-based malware detection model. Sections 4 and 5 explored the experimental study and conclusions of the findings in the respective order.

2. Related research

One of the evolving research topics is the selection of optimal features for malware detection using machine learning. The works [7, 8] have presented supervised learning approaches to perform malware detection using static features. The contemporary contribution [7] has considered the APIs, DLLs (Dynamic Link Libraries), and header feature of PE frequencies. The other contribution [8] has considered the statistics of the Opcode frequency of examined PE-files as features. The optimal feature selection methods such as “Information gain” and “hybrid filter-wrapper technique” are used by these two contributions [7,8] in respective order.

The work [9] has presented a method that segregates the assembly program into small features and produces CFGs set in respective to distinct program functions. The work [10] has shown a disassembled dataset and generated CFGs, which illustrates the flow of code segments. The methods described in both of these contributions [9,10] considering the selected features as feature-vector and assessing the cosine similarity measures of the corresponding feature-vector.

As stated, the static models could not quickly identify malware that utilizes evasion methods. Moreover, adding some of the false API-calls in the executable header might defuse the malware detection ability of these static analysis methods. Hence, the dynamic analysis could be required as a complement for the static methods [5].

Some of the dynamic methods are using the API-calls and respective properties to learn the behavior of the given PE. The work [11] presented an approach that discovers the behavior of the given sample by using 2-gram features listed from system calls and their arguments. Here, they detect new malware classes with the same behavior as unknown malware. The work [12] has utilized 4-gram features for modeling the sequences of API-call. Further, this method [12] estimates the average confidence of these 4-grams to categorize the new samples as malware or benign. The contribution [13] has concentrated on memory and register values in the form of a semantic group and then used in the way of 3-gram input to perform NB classification.

The other contemporary static analysis methods are utilizing the memory dump visuals of executable raw files to explore the malware features using data mining. The other contemporary contribution [14] proposed a K-Nearest-Neighbor (KNN) based classification strategy applied to the wavelet transformation of the image portrayed from the binary executables loaded into memory. The other contemporary model [15] has applied a deep learning approach on gray images of the binary executables. However, the major shuffle in the sequence of calls often portrays the new signature of the grayscale images from the binary executables. However, the major sequence calls are reorganized frequently to identify the new signature of the grayscale images, which tends to load the binary executables into the memory.

The contemporary model [16] portrayed a mining method that extracts the features from call sequences related to the behavioral context. However, the technique fails to track the features if obfuscation is applied to the executables. The contemporary method Droidcat [17] has determined a technique for extracting the context-related features from the call sequences of the obfuscated binary executables. The significance of these methods is uncertain about detecting the zeroth day binary executable. For each dynamic feature, the set of four static features has been considered as a set to generate the hash value as signature [18]. The diversity of the features has limited the performance of this method.

The studies on dynamic malware detection have attracted extensive attention [19–21]. These methods often rely on the flow of the call graphs and the execution sequence of the API calls as dynamic features to denote the behavior. These methods are evincing the process complexity, and computational overhead as the call sequence extraction is prominent and straightforward to identify the scope of malware behavior. The detection of malware includes individual functionality and comparatively intricate for providing appropriate countermeasures are known as an analog attack. The other malware detection strategy [22] is considering the credibility of the dynamically loaded linked libraries and the weightage (frequency) of the calls to system-level operations from the portable executables in respective order. The other contemporary model [23] has adopted a method that categorizes the binary executables as trojans, benign, viruses, and worms. This method primarily processes the input dataset through data normalizing, and later it predicts lower and upper boundaries of the features. Further, it discovers the set of rules based on the pattern matching procedure.

The review of contemporary models has addressed the significant scope to define malware prediction strategies using API calls, and input arguments passed to these API calls and their returned values. The majority of the models have relied on either graph flows, n-gram features of control/data among appealed API-calls, or the information of strings to characterize the behavior of binaries. Each of these approaches has benefits and limitations. For instance, some models that rely on similarities among graphs often exhibit constraints such as memory space and time-consuming. Moreover, the graph-mining methods are complex, since their process complexity is often noticed as NP-complete. The other category of methods is using features or data tuples to depict the malware. These methods are not competent to extend the set of usage in features or data tuples. Unlike these contemporary models, the method "Robust features to detect Malicious activity based on API calls, their Arguments and Return values (MAAR)" [24] introduced a novel model to generate features that are independent of the dataset. The model "MAAR" produces a small group of robust features based on the integration of return values and arguments.

A Machine Learning Approach to Predict Advanced Malware (MLAPAM)" [25] and "A Multi-Dimensional Machine Learning Approach (MDMLA) to Predict Advanced malware" [26] are two contemporary malware detection methods. These methods are using distinctive factors of advanced malware as features in the training phase. The advanced malware combines multiple code blocks developed in different languages, applying cryptography and performing obfuscation. The distinct properties explored by advanced malware such as Stuxnet [3], which compares the traditional malware with the features of other malware to define a rule dictionary through the learning phase of the machine-learning approach. The contributions [25,26] have explored new features and derived correlation between these distinguishing features. However, the complexities and constraints observed under malware categories remain the same.

The critical constraints of these contemporary contributions stated in the above description are the poor specificity and sensitivity towards the identification of the zeroth day attack and malware defined under lessons learned from the contemporary malware defense and detection methods. The severe false alarming often appeared in the advanced malware detection are derived by the combination of code developed in several languages, which are encrypted or obfuscated. Besides that, the impact of newly defined malware weakens the dictionary of rules derived from signatures stated and recommended for malware detection [26]. Concerning this, a novel malware detection strategy that uses regression coefficients as a triad scale for malware detection is proposed in this manuscript. The proposed method is centric to the correlation between all modes of input parameters (call sequences, arguments, and fallouts), which intends to boost the sensitivity, specificity, and detection accuracy of the new malware (zero-day attacks, malware critically differs with existing malware's architecture). The proposed model MLMTS derives a scale from the correlation of call sequences, arguments, and fallouts together lead to the improvement in the specificity, sensitivity, and accuracy with minimal false alarms towards malware detection.

3. Methods and materials

The contribution of this manuscript is hybridizing both formats of static and dynamic analysis properties. The proposed method, "Multi-Level Malware detection using Triad Scale (MLMTS)" uses the call sequences, arguments, and fallouts [24] as features. The usage of these features in hierarchical order intends to depict the zero-day attacks and exhibit the features extracted on the dataset possessing the discerning capability. A test-set is utilized to justify the proposal that comprises novel families and variants of malware. MLMTS generates the feature-sets, which combine the call sequences, input arguments, or/and their corresponding return values during runtime. The results exhibit that the proposed model of multi-level linear regression that portrayed a triad scale using projected features for each level is significant than the contemporary models concerning minimal false alarming, maximal accuracy of malware detection with minimal process overhead.

This proposed model is a regression centric specification scale to estimate the call sequences of any portable executable is prone to malevolent or benevolent. The static malware detection methods usually consider the call sequences (the sequence of function calls) of the portable executables (PEs). In contrast, the dynamic methods rely on the execution flow discovered by executing the suspected PEs in a virtual place isolated from the system environment. However, the call sequences centric malware detection methods often are unsuccessful due to the obfuscation version of the calls or new calls involved in a sequence. Nevertheless, dynamic methods are critically infeasible and complexed in terms of execution and detection. Hence, the proposal of this manuscript utilizes the arguments and fallouts (outcomes of the call in a sequence observed from Portable Executable). The triad-scale is defined from the n-gram call sequences, n-gram arguments of each call sequence, and n-gram fallouts of each call sequence of dynamic size range from 1 to n. The block diagram of MLMTS is shown in Fig. 1, and standard and nonstandard acronyms used in the further description have also listed in Table 1.

The overall contribution is a machine learning approach, which contains the training and labels prediction phases. The training process has carried in a sequence of tasks listed as

(i) Preprocessing that removes the unqualified records from the corpus of labelled records given for the training phase of the proposal
(ii) Finding n-gram patterns of the call sequences, which are the sequence of portable executables, exists in one or more of the records labelled either positive or negative.
(iii) Finding the argument patterns of each n-gram call sequence pattern, which are the sequence of portable executables, exists in one or more of the records labelled either positive or negative.
(iv) Finding the fallout patterns of each n-gram call sequence pattern, which are the sequence of portable executables, exists in one or more of the records labelled either positive or negative.
(v) Optimizing the n-gram patterns of call sequences, which are the call sequences having a high correlation with only label positive or negative.

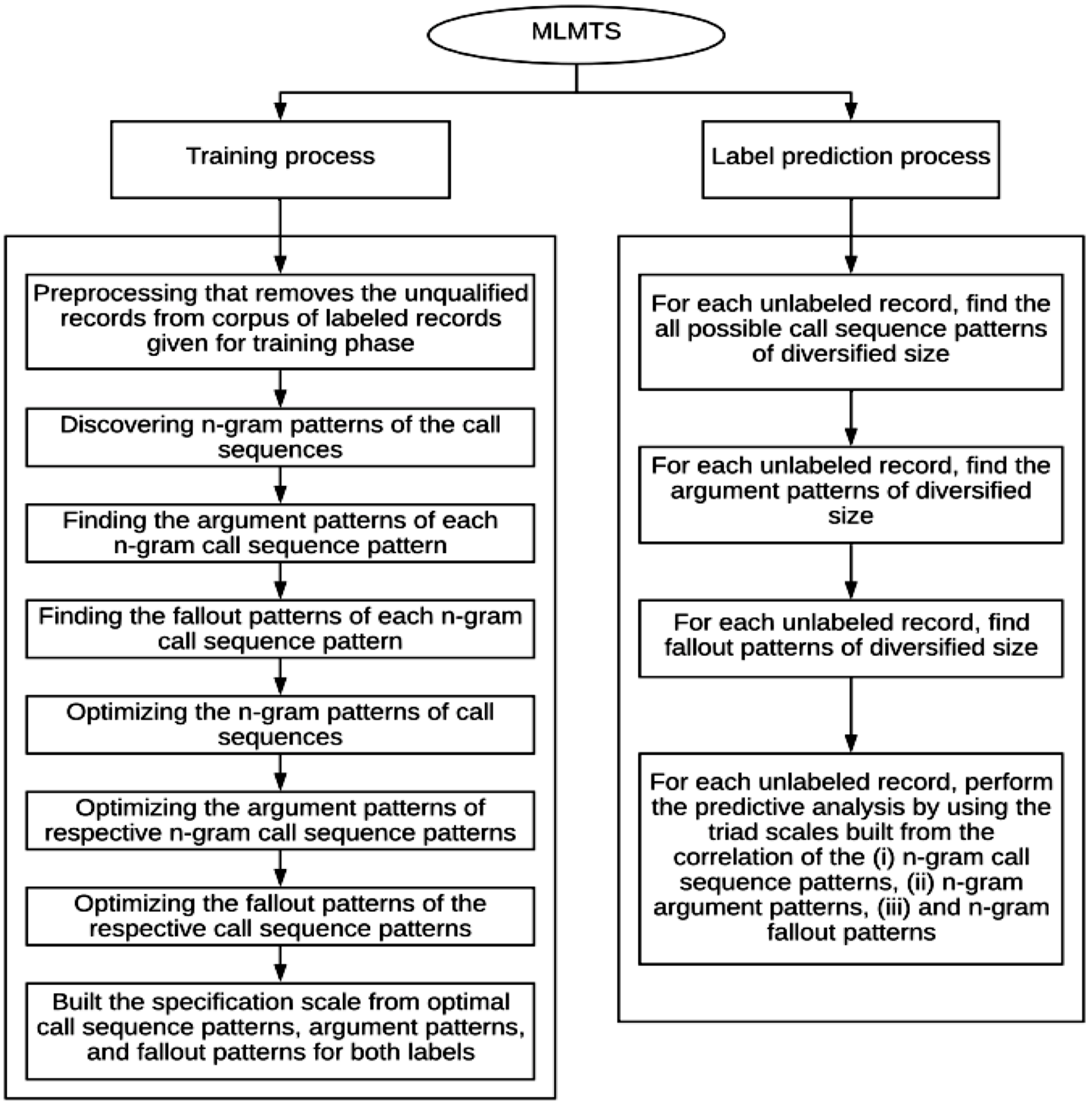

MLMTS
Training process
Label prediction process
Preprocessing that removes the unqualified records from corpus of labeled records given for training phase
Discovering n-gram patterns of the call sequences
Finding the argument patterns of each n-gram call sequence pattern
Finding the fallout patterns of each n-gram call sequence pattern
Optimizing the n-gram patterns of call sequences
Optimizing the argument patterns of respective n-gram call sequence patterns
Optimizing the fallout patterns of the respective call sequence patterns
Built the specification scale from optimal call sequence patterns, argument patterns, and fallout patterns for both labels
For each unlabeled record, find the all possible call sequence patterns of diversified size
For each unlabeled record, find the argument patterns of diversified size
For each unlabeled record, find fallout patterns of diversified size
For each unlabeled record, perform the predictive analysis by using the triad scales built from the correlation of the (i) n-gram call sequence patterns, (ii) n-gram argument patterns, (iii) and n-gram fallout patterns

Fig. 1. Block diagram representation of MLMTS.

Table 1
List of standard and nonstandard acronyms used.

PE	Portable Executable
DLLs	Dynamic Link Libraries
IG	Information Gain
CFG	Control Flow Graph
MLMTS	Machine Learning-based malware detection using Triad Scale
MAAR	Malicious activity based on API calls, their Arguments and Return
MLAPAM	Machine Learning Approach to Predict Advanced malware
MDMLA	Multi-Dimensional Machine Learning Approach
MCC	Matthews's correlation coefficient

(vi) Optimizing the argument patterns of respective n-gram call sequence patterns is the call sequences having a high correlation with only label positive or negative.
(vii) Optimizing the fallout patterns of the respective call sequence patterns is the call sequences having a high correlation with only label positive or negative.
(viii) Built the specification scale for both labels using the optimal call sequence patterns, argument patterns, and fallout patterns

The label prediction process also referred to as the testing phase, carried in multiple phases listed as,

(i) For each unlabeled record, find the all possible call sequence patterns of diversified size
(ii) For each unlabeled record, find the argument patterns of diversified size
(iii) For each unlabeled record, find fallout patterns of diversified size
(iv) For each unlabeled record, perform the predictive analysis by using the triad scales built from the correlation of the (i) n-gram call sequence patterns, (ii) n-gram argument patterns, (iii), and n-gram fallout patterns in respective order.

The following sections reveal the methodology used in each of the phases described above related to training and label prediction of machine learning. The list of notations used in the mathematical model are listed in the following table (see Table 2).

3.1. Preprocessing

This phase discards the portable executables that are not engaged either of the labels positive (malware), negative (benign) in the given training corpus. Further explores the call sequencesCS, argument sequencesAS, and fallout sequences$FS$of each malicious portable executable (PE) of the given training corpus. The call sequence$\{r\exists r \in CS\}$represents the functions stacked in corresponding PE shall execute in the corresponding sequence. An argument sequence$\{ar\exists ar \in AS\}$of the PE represents the arguments in stacked order, which have passed to each function in the call sequence of the respective PE. Similarly, it derives the fallouts sequence$\{fo\exists fo \in FS\}$ of each PE of the given corpus. A fallout sequence represents the return values of the functions that appeared in the call sequence of the respective PE.

Table 2
List of notations used and respective descriptions.

CS	Call Sequences
AS	Argument Sequences
FS	Fallout Sequences
nS	N-Gram Sequences
$P_+, \quad P_-$	Positive and Negative Records
$'i'$	Index
np	N-Gram Pattern
s_+^{np}	Support
v_+^i	Vector
csp	Call Sequence Pattern
$p\tau_2$	Probability Threshold
onP	Optimal N-Gram Features
arP	Argument Patterns
foP	Fallout Patterns
$\|r\|$	Record-Size
csc_+	Confidence
ss_+^{csP}	Specification Scale

3.2. Optimizing the n-gram patterns

Sets arC_+,arC_- from the setAS, and the setsfoC_+,$foC_$from the setFS.

```
Optimizing-N-gram-Patterns($nS,\quad P_+,\quad P_-$)Begin // member function that performs n-gram feature optimization, which receives n-gram sequences$nS$and respective positive and negative records as the sets$P_+,\quad P_-$
  Let an index'$i$' is initialized to 1
  For each index$\{i\exists i = 1, 2, ..., n\}$, Begin // having a value in the range of minimum pattern size$1$to max pattern size$n$
    Extract $i$-gram call sequence patterns from the set$nS$as a set$nP_i$
      For each$i$-gram sequence pattern$\{np\exists np \in nP_i \wedge |np| \cong i\}$Begin // Find the support of the n-gram pattern$np$concerning the label positive, which is as follows
      $s_+^{np} = (\sum_{j=1}^{|P_+|}\{1\exists np \subseteq r \wedge r \in P_+\})*(|P_+|)^{-1}$// this denotes the ratio of source records having the n-gram pattern$np$as subset against the total number of call sequences labelled as positive and listed in the set$P_+$.
      Move the support$s_+^{np}$ to a vector$v_+^i$
      Similarly, find the support of the call sequence pattern$csp$regarding the label negative, which is as follows
      $s_-^{np} = (\sum_{j=1}^{|P_-|}\{1\exists np \subseteq r_j \wedge r_j \in P_-\})*(|P_-|)^{-1}$// this denotes the ratio of call sequences having the call sequence pattern$np$as subset against the total number of call sequences labelled as negative and listed in the set$P_-$.
      Then move the support$s_-^{np}$to a vector$v_-^i$
    End
    Further, scale the variance between the vectors $v_+^i$, $v_-^i$using a competent distribution diversity assessment method called dual tailed $t$-test [27] (see Section3.3).
    If the t-score and p-value [27] observed from the dual tailed $t$-test has used further to estimate the optimality of the$i-gram$call sequence patterns as follows.
    The resultant p-value is greater than the given probability threshold$p\tau$; the $i-gram$patterns are optimal and move these$i-gram$patterns to the set$onP$representing the resultant optimal n-gram features.
    The process explored in previously mentioned statements executes for each index'$i$'value range from 1 to n.
    Return the optimal n-gram features as a set$onP$
  End
End // of the Optimizing-N-gram-Patterns($nS,\quad P_+,\quad P_-$)
```

3.3. T-Test for distribution diversity estimation

The distribution diversity has taken as a parameter to estimate the optimality of the features listed as n-gram call sequence patternscsP, n-gram argument patternsarP, and n-gram fallout patternsfoPtowards the labels positive and negative. The detailed exploration of the optimal feature selection is described in the section above (Section3.2). The t-test is adapted to estimate the distribution diversity of the features towards positive and negative labels. This section delineates the method of performing the t-test to identify distribution diversity. From the contribution [27], the scheme that evaluates the distribution diversity known as the t-test is used for selecting the optimal features relevant to both labels. Here, the t-test is included for selecting the optimal-features associated with both the positive and negative records of the corpusCS.

The diversity between any two distinct vectors can depict using T-score as,

$$\begin{aligned}
v_+^{stdev} &= \left(\sum_{i=1}^{|v_+|}(x_i - \langle v_+\rangle)^2\right)*(|v_+| - 1)^{-1} \\
v_-^{stdev} &= \left(\sum_{j=1}^{|v_-|}(x_j - \langle v_-\rangle)^2\right)*(|v_-| - 1)^{-1} \\
t\_stdev &= \sqrt{v_+^{stdev} + v_-^{stdev}} \\
t - score &= (\langle v_+\rangle - \langle v_-\rangle)*(t\_stdev)^{-1}
\end{aligned} \tag{1}$$

In (Eq (1)), the notations$\langle v_+\rangle$, $\quad\langle v_-\rangle$entails the mean of the respective vectors v_+, $\quad v_-$

The notationsx_i,$x_j$refer to the entries of the vectors in sequence from the index 1 to vector sizes $|v_+|$, $\quad |v_-|$represented by i, $\quad j$in respective order.

Finds the ratio of the absolute difference of the means$\langle v_+\rangle - \langle v_-\rangle$against the aggregate of the standard deviations of corresponding vectorsv_+, $\quad v_-$.

Then compute p-value [27] (degree-of probability) in the t-table [27] for attained t-score. Here, the p-value, which is lower than the probability threshold$p\tau$, signifies that both the vectors are different; therefore, the patterns represented by the entries of the corresponding vectors has said to be optimal-feature.

3.4. Optimizing call sequence n-grams

For each call sequence$\{r\exists r \in CS\}$of the setCS, find all possible call sequence patterns of size 1 to record-size$|r|$. Move all possible call sequence patterns of count$(|r| + 1)*(|r|)*2^{-1}$discovered from the record$\{r\exists r \in CS\}$to the setcsP. The setCShas to partition into two sets such that one represents all the records of the set having a positive label, and the other set represents the records of the negative label. This phase discovers the setsCS_+,CS_-form the setCS. Further, invokes the member function Optimizing_N-gram_Patterns(csP, CS_+,CS_-) that returns optimal call sequence patterns, which has been received as$ocsP$

3.5. Optimizing the argument patterns

Prepare a corpusarCthat contains the set of records as follows. Each record$\{ar\exists ar \in arC\}$of the set shall contain the sequence of arguments, which have passed to the sequence of calls in the record$\{r\exists r \in CS\}$of the corpus$CS$and entails the label assigned to the corresponding call sequence record$\{r\exists r \in CS\}$. Further, partition the corpusarC into two setsarC_+, *and* arC_-, such that the setarC_+contains the records that exist in the corpus$arC$and having the label positive. The other setarC_-contains the records of the corpusarCthat are having label negative. Each record$\{ar\exists ar \in arC\}$ of the set$arC$with index$i$represents the arguments passed to the call sequence of the record$\{r\exists r \in CS\}$with index$i$in the corpusCS. The similar process stated in the above section (Section3.4) has been used to list the total$(|ar| + 1)^*(|ar|)^*2^{-1}$number of argument sequence patterns from each argument sequence$\{ar\exists ar \in AS\}$as a setarP. Further, invoke the function called "Optimizing_N-gram_Patterns(arP, arC_+,arC_-)" (see Section3.2) that returns optimal call sequence patterns, which has been received as a set$oarP$.

3.6. Optimizing the fallout patterns

The process that follows to list the fallout sequence patterns is very much similar to listing argument sequences, which discovers$(|fo| + 1)^*(|fo|)^*2^{-1}$several patterns from each fallout sequence$\{fo\exists fo \in FS\}$. Further, list the fallout sequence patterns as a setfoP. Prepare the corpusfoCthat contains a set of records, such that each record$\{fr\exists fr \in foC\}$contains the sequence of fallouts (returning arguments), resulting from the sequence of calls that exist in the record$\{r\exists r \in CS\}$of the corpus$CS$and entails the label assigned to the corresponding call sequence record$\{r\exists r \in CS\}$. Each record$\{fr\exists fr \in foC\}$of the set$foC$with index$'i'$represents the fallouts exhibited by the call sequence of the record$\{r\exists r \in CS\}$with index$'i'$ in the corpusCS.

Further, partition the corpusfoCinto two setsfoC_+, *and* foC_-, such that the setfoC_+contains the records that exist in the corpus$foC$and having the label positive. The other setfoC_-contains the records of the corpusfoCthat are having label negative. Further, a similar version of optimal pattern selection has adapted that invokes "Optimizing_N-gram_Patterns(foP, foC_+,foC_-)" (see Section3.2) that returns optimal n-gram patterns, which has to receive as a set$ofoP$.

3.7. Triad scale by regression coefficients

This section delivers the regression coefficients for diversified features listed optimal n-gram patterns, which have been discovered from their empirical probabilities of the n-gram patterns.

```
Find-Regression-Coefficients($nP$)begin// the function that discovers regression coefficients of the feature formats
  $ss = (\sum_{j=1}^{|nP|}\{s(np_j)\exists np_j \in nP\})^*|nP|^{-1}$// Find the mean of the empirical probabilities of the optimal n-gram patterns as the regression coefficient
  $ssd = (\sum_{j=1}^{|nP|}\{\sqrt{(ss - s(np_j))^2}\exists np_j \in nP\})^*|nP|^{-1}$// Finding the root mean square distance (mean deviation) of the empirical probability of the optimal n-gram patterns
  $ssl = ss - ssd$// the absolute distance of the empirical probability and respective mean deviation is the lower bound of the regression coefficient
  $ssu = ss + ssd$// the aggregate of the empirical probability and respective mean deviation is the upper bound of the regression coefficient
  Return ($ss, ssl, ssu$)// returns the regression coefficient, respective lower and upper bounds as triad scale
End // of Find-Regression-Coefficients($nP$)
```

3.7.1. Triad scale of the call sequence patterns

The regression coefficients of the call sequence patternscsP_+ have derived by invoking the function Find-Regression-Coefficients (csP_+), which returns the regression coefficients$(ss_+^{csP}, ssl_+^{csP}, ssu_+^{csP})$as triad scale of the call sequence patterns of the positive label. Similarly, the triad scale$(ss_-^{csP}, ssl_-^{csP}, ssu_-^{csP})$of the call sequence patterns for the negative label has to estimate by invoking the function Find-Regression-Coefficients(csP_-),

3.7.2. Triad scale of the argument patterns

The setarP_+argument patterns of the positive label have to pass as an input parameter of the function Find-Regression-Coefficients (arP_+)that return a set of regression coefficients$(ss_+^{arP}, ssl_+^{arP}, ssu_+^{arP})$as a triad scale. Similarly, the triad scale$(ss_-^{arP}, ssl_-^{arP}, ssu_-^{arP})$of the argument patterns for the negative label has to be estimated by invoking the function Find-Regression-Coefficients(arP_+)with argument sequence patterns$arP_-$of the negative label as an input parameter.

3.7.3. Triad scale of the fallout patterns

The regression coefficients$(ss_+^{foP}, ssl_+^{foP}, ssu_+^{foP})$of the fallout patterns of the positive label as triad scale have to receive from the function Find-Regression-Coefficients(foP_+), which has to invoke through the set of argument patterns$arP_+$of the positive label. The triad scale $(ss_-^{foP}, ssl_-^{foP}, ssu_-^{foP})$of the fallout patterns of the negative label has emerged as the return set of the function Find-Regression-Coefficients(arP_-), which has to invoke through the fallout patterns of the setfoP_-as an input parameter.

3.8. Label prediction

For a given unlabeled call sequence$r$of the size$m$ extracted from portable executable, derive all possible call sequence patterns. The total number of patterns discovered from the call sequence$r$ is$m*(m+1)*2^{-1}$which have buffered into the set$csPT$. Further, extract the arguments passed to the calls in the given call sequence record$r$and prepare a recordarTexhibiting these arguments in the sequence of calls of the given record. Similarly, define a recordfoT that represents the calls' fallouts in the sequence of the given record$r$. The sequence of fallouts in foT represents the sequence of the calls in the record$r$.

Further, discover the possible argument patterns from the record$arT$and list them in a set$arPT$. The total number of argument patterns$p$ has to estimate as$p = |arT|*(|arT|+1)*2^{-1}$. Similarly, portray the possible patterns of the fallouts from the record foTas a set$foPT$of size$q = |foT|*(|foT|+1)*2^{-1}$.

Discover the call sequence patterns that are common in both the sets $csP, \quad and \quad csPT$as a set$ccsp$. Then find the positive and negative confidence of the call sequence patterns listed in the set$ccsp$as follows Eq (2), 3, (4).

$$ccsp = csP \cap csPT \tag{2}$$

// discover the call sequence patterns those are common in both the sets $csP, \quad and \quad csPT$as a set$ccsp$

$$\mathrm{csc}_{+} = \left(\sum_{i=1}^{|ccsp|}\{s_{+}^{csp_i} \exists csp_i \in ccsp\}\right)*m^{-1} \tag{3}$$

// Find the call sequence confidence for a positive label, which is the ratio of empirical probabilities of the respective call sequences obtained from the records labelled as positive of the training corpus

$$\mathrm{csc}_{-} = \left(\sum_{i=1}^{|ccsp|}\{s_{-}^{csp_i} \exists csp_i \in ccsp\}\right)*m^{-1} \tag{4}$$

// Find the call sequence confidence for a negative label, which is the ratio of empirical probabilities of the respective call sequences obtained from the records labelled as negative of the training corpus.

Similarly, find the positive and negative confidence of the argument patterns and fallout patterns listed in the sets$arPT, \quad foPT$ Eq (5), 6, (7).

$$carp = arPT \cap arP \tag{5}$$

Find the patterns common in both sets$arPT, \quad arP$

$$arc_{+} = \left(\sum_{i=1}^{|carp|}\{s_{+}^{arp_i} \exists arp_i \in carp\}\right)*p^{-1} \tag{6}$$

Find the arguments confidence for a positive label, which is the ratio of empirical probabilities of the respective argument patterns obtained from the records labelled as positive of the training corpus.

$$arc_{-} = \left(\sum_{i=1}^{|carp|}\{s_{-}^{arp_i} \exists arp_i \in carp\}\right)*p^{-1} \tag{7}$$

Find the arguments confidence for the negative label, which is the ratio of empirical probabilities of the respective argument patterns obtained from the records labelled as negative of the training corpus.

Further phase finds the positive and negative confidence of the fallout patterns as follows in Eq (8), 9, (10).

$$cfop = foPT \cap foP \tag{8}$$

Find the patterns common in both sets $foPT, \quad foP$

$$foc_{+} = \left(\sum_{i=1}^{|cfop|}\{s_{+}^{fop_i} \exists fop_i \in cfop\}\right)*q^{-1} \tag{9}$$

Find the fallout confidence for the positive label, which is the ratio of empirical probabilities of the respective fallout patterns obtained from the records labelled as positive of the training corpus.

$$foc_{-} = \left(\sum_{i=1}^{|cfop|}\{s_{-}^{fop_i} \exists fop_i \in cfop\}\right)*q^{-1} \tag{10}$$

Find the fallout confidence for the negative label, which is the ratio of empirical probabilities of the respective fallout patterns obtained from the records labelled as negative of the training corpus.

Further, these confidence metrics has to correlate with identifying the label, which is as follows

1 Label the given test record as positive regarding call sequence specification scales, if the
   a $\mathrm{csc}_{+} \geq ssu_{+}^{csP}$// Positive confidence$\mathrm{csc}_{+}$of the call sequences of the given test record is greater than the upper bound of the specification scalessu_{+}^{csP}(see Section3.7.1) of the call sequences$csP$for the positive label.

b $\mathrm{csc}_+ \geq ss_+^{csP} \&\& \mathrm{csc}_- < ss_-^{csP}$ // Positive confidence csc_+ of the call sequences of the given test record is greater than the specification scales ss_+^{csP} of the call sequence patterns for a positive label, and negative confidence csc_- of the call sequences of the given test record is less than the specification scales ss_-^{csP} of the call sequences for the negative label.

c $\mathrm{csc}_+ \geq ssl_+^{csP} \&\& \mathrm{csc}_- < ssl_-^{csP}$ // Positive confidence csc_+ of the call sequences of the given test record is greater than or equal to the specification scale lower-bound ssl_+^{csP} of the call sequence patterns for the positive label. And negative confidence csc_- of the call sequences of the given test record is less than the lower-bound of the specification scales ssl_-^{csP} of the call sequences for the negative label.

2 Label the given test record as positive regarding argument specification scales, if the

a $arc_+ \geq ssu_+^{arP}$ // Positive confidence arc_+ of the argument patterns of the given test record is greater than the upper bound of the specification scales ssu_+^{arP} of the argument patterns for the positive label (see Section3.7.2).

b $arc_+ \geq ss_+^{arP} \&\& arc_- < ss_-^{arP}$ // Positive confidence arc_+ of the argument patterns of the given test record is greater than the specification scales ss_+^{arP} of the argument patterns for the positive label, and negative confidence arc_- of the argument patterns of the given test record is less than the specification scale of the argument patterns for the negative label.

c // Positive confidence of the argument patterns of the given test record is greater than or equal to the specification scale lower-bound of the argument patterns for the positive label. And negative confidence of the argument patterns of the given test record is less than the lower-bound of the specification scale of the argument patterns for the negative label.

3 Label the given test record as positive regarding fallout specification scales, if the

a // Positive confidence of the fallout patterns of the given test record is greater than the upper bound of the specification scale of the fallout patterns for the positive label (see Section3.7.3).

b // Positive confidence of the fallout patterns of the given test record is greater than the specification scale of the fallout patterns for the positive label, and negative confidence of the fallout patterns of the given test record is less than the specification scale of the fallout patterns for the negative label.

c // Positive confidence of the fallout patterns of the given test record is greater than or equal to the specification scale lower-bound of the fallout patterns for the positive label. And negative confidence of the fallout patterns of the given test record is less than the lower-bound of the specification scale of the fallout patterns for the negative label.

4 Label the given test record as negative regarding call sequence specification scales, if the

a Positive confidence of the call sequences of the given test record is less than the specification scale of the call sequences for the positive label, and the negative confidence (of the call sequences of the test record) is greater than the upper bound of the specification scale of the call sequences (see Section3.7.2).

b Positive confidence of the call sequences of the given test record is less than the lower-bound of the specification scale of the call sequences for a positive label. And the negative confidence (of the call sequences of the test record) is greater than the specification scale (of the call sequences) for the negative label (see Section3.7.2).

5 Label the given test record as negative concerning argument specification scales, if the,

a // Positive confidence of the argument patterns of the given test record is less than or equals the specification scale of the argument patterns for the positive label. And the negative confidence (of the argument patterns depicted from test records) is greater than the upper bound of the specification scale of the argument patterns for the negative label (see Section3.7.2).

b // Positive confidence of the argument patterns (of the given test record) is less than the lower-bound of the specification scale of the argument patterns for the positive label. And the negative confidence (of the argument patterns of the test record) is greater than the specification scale (of the argument patterns) for the negative label (see Section3.7.2).

6 Label the given test record as negative regarding fallout triad scale, if the,

a // Positive confidence of the fallout patterns of the given test record is less than or equals the specification scale of the fallout patterns for the positive label. And the negative confidence (of the fallout patterns of the test record) is greater than the upper bound of the specification scale of the fallout patterns for the negative label (see Section3.7.3).

b // Positive confidence of the fallout patterns (of the given test record) is less than the lower-bound of the specification scale of the fallout patterns for the positive label. And the negative confidence (of the fallout patterns of the test record) is greater than the specification scale (of the fallout patterns) for the negative label (see Section3.7.3).

In contrast to the above conditions, the given record of call sequences can be treated as suspicious. However, the definition of these conditions is solely domain-specific.

4. Experimental study

This section explores the empirical study carried to assess the performance of the proposed model and the other contemporary models using the benchmark dataset. The performance significance of the proposed model must be scaled by comparing the observed results of the classification assessment metrics namely, “Precision, Specificity, Sensitivity, Accuracy, F-measure, False Alarming, and Matthews’s correlation coefficient (MCC)”. The manifested results from the proposed model MLMTS, the other contemporary models

“A Machine Learning Approach to Predict Advanced malware (MLAPAM)” [25] and “A Multi-Dimensional Machine Learning Approach (MDMLA) to Predict Advanced malware” [26] are compared and concluded with the significance of the proposed model towards malware detection.

4.1. The dataset

The dataset named Apimds[28] has been utilized for experiments, where 23,080 malware samples have been selected randomly from the Malicia-project [29] and Virus Total [30] malware dataset that is boosted by adding known benign calls of 9436. The total amount of records in the final dataset is 32516.

The metric precision indicates the ratio of records labelled correctly as positive to the total amount of falsely labelled records as positive. Fig. 2 represents the graph between 10-folds and precision for MDMLA, MLAPAM, and MLMTS. The average precision for the proposed method MLMTS that is perceived from the 10-fold strategy is 0.99 ± 0.005. While the average precision for the contemporary methods MLAPAM and MDMLA are 0.968 ± 0.003 and 0.98 ± 0.003 in respective order. From the statistics, as shown in Table 3, it is noticed that the proposed model MLMTS performs better when compared to contemporary MLAPAM and MDMLA methods.

The metric specificity denotes the ratio of correctly labelled records as negative against to total amount of negative label records. Fig. 3 represents the graph between 10-folds and specificity for MDMLA, MLAPAM, and MLMTS. Average specificity for the proposed method MLMTS and contemporary methods MLAPAM and MDMLA are observed from the 10-fold cross-validation as 0.976 ± 0.011, 0.926 ± 0.008, and 0.953 ± 0.008, respectively. From the statistics, it is noticed that the proposed model MLMTS outperforms the contemporary MLAPAM and MDMLA methods.

The metric sensitivity denotes the ratio between the test records labelled correctly as positive, and the total amount of the positive records provided for testing. Fig. 4 depicts the graph between 10-folds and sensitivity for MDMLA, MLAPAM, and MLMTS Average sensitivity for the proposed method MLMTS that perceived from the 10-fold scheme is 0.987 ± 0.005. In contrast, the average sensitivity for the contemporary methods MLAPAM and MDMLA are 0.926 ± 0.009 and 0.957 ± 0.009 in respective order. From the statistics, it is noticed that the proposed model MLMTS is more significant when compared to contemporary MLAPAM and MDMLA methods.

The metric accuracy indicates the total performance for selecting the labels of specified unlabeled records that is the ratio of cumulative of correctly labelled positive and negative records against the overall amount of records by both the labels provided for testing. Fig. 5 signifies the graph between 10-folds and accuracy for MDMLA, MLAPAM, and MLMTS. Average accuracy for the proposed method MLMTS and contemporary methods MLAPAM and MDMLA that perceived from the 10-fold scheme are 0.984 ± 0.003, 0.926 ± 0.007, and 0.956 ± 0.007, respectively. From the statistics, it is noticed that the proposed model MLMTS is considered optimal when compared to contemporary MLAPAM and MDMLA methods.

The metric F-measure indicates the weighted harmonic mean of precision & recall. Fig. 6 depicts the graph between 10-folds and F-Measure for MDMLA, MLAPAM, and MLMTS. The average F-Measure for the proposed method MLMTS that is perceived from the 10-fold scheme is 0.983 ± 0.008. In contrast, the average F-measure for the contemporary methods MLAPAM and MDMLA are 0.947 ± 0.006 and 0.967 ± 0.006 in respective order. From the statistics, it is noticed that the proposed model MLMTS is more significant when compared to contemporary MLAPAM and MDMLA methods.

Fig. 7 signifies the graph between 10-folds and false alarming rates for MDMLA, MLAPAM, and MLMTS. The average false alarming rate for the proposed method MLMTS and contemporary methods MLAPAM and MDMLA that perceived from the 10-fold scheme are 0.016 ± 0.003, 0.074 ± 0.007, and 0.044 ± 0.007, respectively. From the statistics, it is noticed that the proposed model MLMTS is optimal with minimal false alarming that compared to contemporary MLAPAM and MDMLA methods.

The metric Matthews Correlation Coefficient (MCC) is utilized in machine learning to measure the quality of binary classifications.

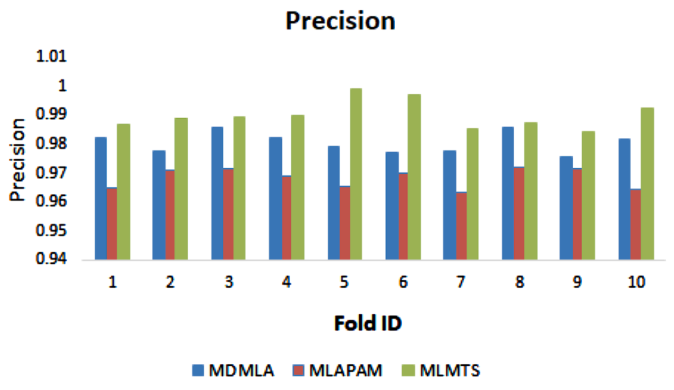


Fig. 2. Graphical representation of precision at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM (Machine Learning Approach to Predict Advanced Malware) methods.

Table 3
Average and Standard Deviation (SD) for the various metrics.

	Precision	Specificity	Sensitivity	Accuracy	F-measure	False Alarming	MCC
MLMTS	0.99 ± 0.005	0.976 ± 0.011	0.987 ± 0.005	0.984 ± 0.003	0.983 ± 0.008	0.016 ± 0.003	0.961 ± 0.007
MLAPAM	0.968 ± 0.003	0.926 ± 0.008	0.926 ± 0.009	0.926 ± 0.007	0.947 ± 0.006	0.074 ± 0.007	0.828 ± 0.014
MDMLA	0.98 ± 0.003	0.953 ± 0.008	0.957 ± 0.009	0.956 ± 0.007	0.967 ± 0.006	0.044 ± 0.007	0.896 ± 0.017

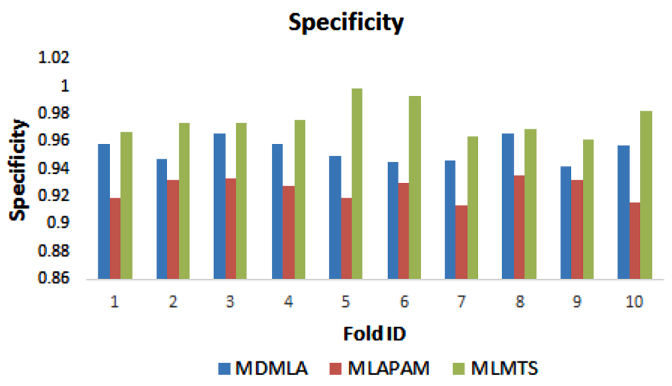


Fig. 3. Graphical representation of specificity at 10-folds for the proposed method MLMTS and contemporary MDMLA and MLAPAM methods.

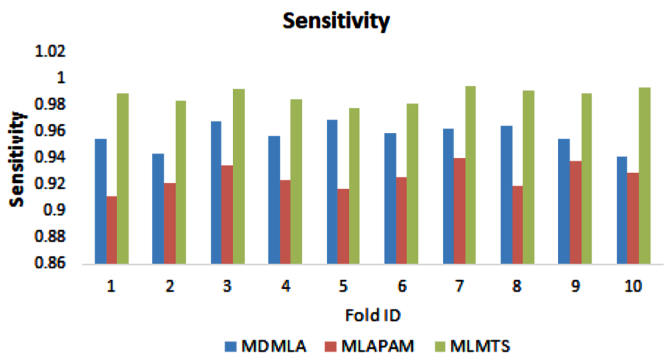


Fig. 4. Graphical representation of sensitivity at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM methods.

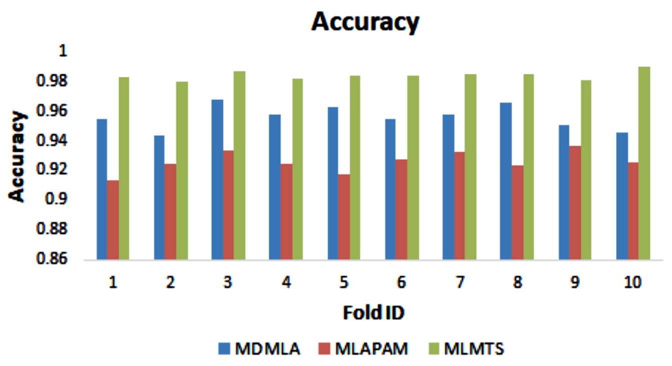


Fig. 5. Graphical representation of accuracy at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM methods.

Fig. 8 depicts the graph between 10-folds and MCC for MDMLA, MLAPAM, and MLMTS. The average MCC for the proposed method MLMTS that is perceived from the 10-fold scheme is 0.961 ± 0.007. In contrast, the average MCC for the contemporary methods MLAPAM and MDMLA are 0.828 ± 0.014 and 0.896 ± 0.017 respectively. From the statistics, it is noticed that the proposed model MLMTS is more significant towards binary classification process that compared to contemporary MLAPAM and MDMLA methods.

All factors of performance analysis verified under the metrics, those recommended for machine learning have acclaimed the

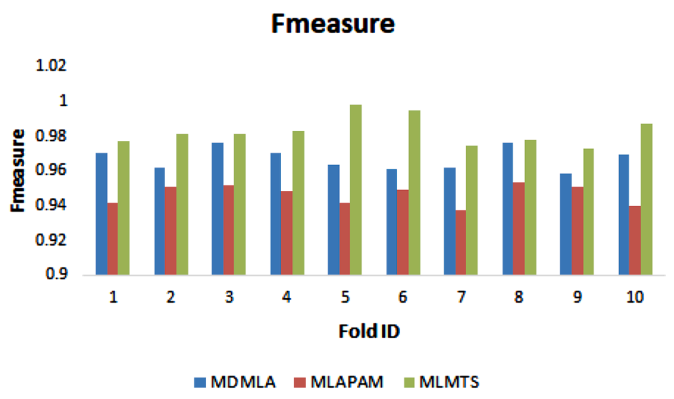


Fig. 6. Graphical representation of F-measure at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM methods.

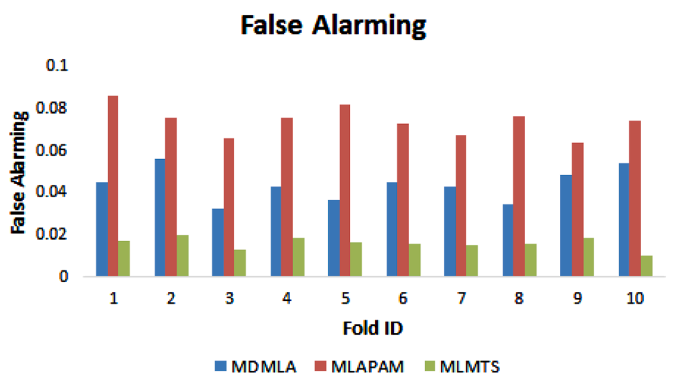


Fig. 7. Graphical representation of false alarming rate at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM methods.

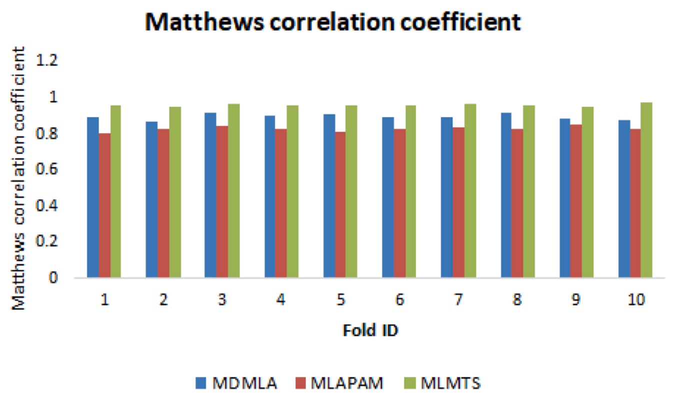


Fig. 8. Graphical representation of the Matthews correlation coefficient at 10-folds for the proposed MLMTS method and contemporary MDMLA and MLAPAM methods.

significance of the proposed model. The qualitative factors considered to boost the significance of the experimental study are, the records pooled up as a set to perform the testing process shall evince the least correlation with the records undertaken for the training process, which helps to reflect the zeroth-day attacks. As stated in the literature review, contemporary methods resulted in poor specificity, sensitivity, and detection accuracy. Whereas the proposed method, MLMTS, is using all three possible formats of the features, which are n-gram patterns of call sequences, arguments, and fallouts. Hence, the MLMTS can detect the malware with maximal accuracy, specificity, and sensitivity that manifested due to the rules defined under the correlation of all these input formats.

5. Conclusion

A machine learning approach is to identify the malware scope of a portable executable that has been proposed in this manuscript. Unlike contemporary models, this method performs the detection process in three phases. Among these, the first and second phases are using call sequences, and arguments passed to these call sequences, and the third phase uses the fallouts of these calls. According to this, the first and second phases fall into the category of static analysis, and the third phase falls into the dynamic analysis category. The execution of the second and third phases initiates if the predecessor phase is not confident to classify the given portable executable. The proposal is a multi-level linear regression technique that defines a triad scale from the input given for each phase. The empirical study has carried on the benchmark malware dataset. The performance significance of the proposed model is scaled by comparing the results obtained from the contemporary models MLAPAM and MDMLA, which have been executed on the same dataset. The significance of the proposal has been scaled by comparing the results obtained from the metrics such as Precision, Specificity, Sensitivity, Accuracy, F-measure, False Alarming, and MCC.

Regarding resultant values of these metrics, the proposed method MLMTS outperformed the contemporary methods MLAPAM and MDMLA with minimal false alarming and maximal specificity, sensitivity, and accuracy. The other two metric values F-measure, and MCC, also evincing the significance of the proposed method. The contribution of the proposed work motivates future research towards the development of fuzzy guided malware detection by using the triad scales as a member function.

Author statement

All authors confirm that we have participated sufficiently in the work to take public responsibility for the content, including participation in the concept, design, analysis, writing, or revision of the manuscript. Furthermore, each author confirms that this material or similar material has not been and will not be submitted to or published in any other publication before its appearance in the Computers and Electrical engineering.

Declaration of Competing Interest

All author states that there is no conflict of interest.

References

[1] Zimba Aaron. malware-free intrusion: a novel approach to ransomware infection vectors. Int J Comput Sci Inform Secur 2017;15(2):317–25.
[2] Sood Aditya K, Enbody Richard J. Targeted cyberattacks: a superset of advanced persistent threats. IEEE Secur Priv 2012;11(1):54–61.
[3] Lindsay Jon R. Stuxnet and the limits of cyber warfare. Secur Stud 2013;22(3):365–404.
[4] Cesare Silvio, Xiang Yang, Zhou Wanlei. Malwise an effective and efficient classification system for packed and polymorphic malware. IEEE Trans Comput 2012; 62(6):1193–206.
[5] Yan Jinpei, Qi Yong, Rao Qifan. "Detecting malware with an ensemble method based on deep neural network. Secur Commun Netw 2018;2018:16. Article ID 7247095.
[6] Rudra Kumar M, Kumar Gunjan V. Review of machine learning models for credit scoring analysis. Revista Ingeniería Solidaria 2020;16(1):1–16.
[7] Baldangombo Usukhbayar, Jambaljav Nyamjav, Horng Shi-Jinn. A static malware detection system using data mining methods. Int J Artif Intell Appl 2013;4(4): 113.
[8] Alazab M, Huda S, Abawajy J, Islam R, Yearwood J, Venkatraman S, Broadhurst R. A hybrid wrapper-filter approach for malware detection. J Netw 2014;9(11): 2878–91.
[9] Alam Shahid, Nigel Horspool R, Traore Issa. MARD: a framework for metamorphic malware analysis and real-time detection. In: 2014 IEEE 28th International Conference on Advanced Information Networking and Applications. IEEE; 2014. p. 480–9.
[10] Mehra Vishakha, Jain Vinesh, Uppal Dolly. DACOMM: detection and classification of metamorphic malware. In: 2015 Fifth International Conference on Communication Systems and Network Technologies. IEEE; 2015. p. 668–73.
[11] Rieck K, Trinius P, Willems C, Holz T. Automatic analysis of malware behavior using machine learning. J Comput Secur 2011;19(4):639–68.
[12] Ravi Chandrasekar, Manoharan R. malware detection using windows API sequence and machine learning. Int J Comput Appl 2012;43(17):12–6.
[13] Van Nhuong N, Nhi VTY, Cam NT, Phu MX, Tan CD. Semantic set analysis for malware detection. In: IFIP International Conference on Computer Information Systems and Industrial Management. Springer; 2015. p. 688–700.
[14] Liu L, Wang BS, Yu B, Zhong QX. Automatic malware classification and new malware detection using machine learning. Front Inform Technol Electron Eng 2017;18(9):1336–47.
[15] Yajamanam S, Selvin VRS, Di Troia F, Stamp M. Deep Learning versus gist descriptors for image-based malware classification. Icissp 2018:553–61.
[16] Pektaş Abdurrahman, Acarman Tankut. "malware classification based on API calls and behavior analysis. IET Inf Secur 2017;12(2):107–17.
[17] Cai H, Meng N, Ryder B, Yao D. Droidcat: effective android malware detection and categorization via app-level profiling. IEEE Trans Inf Forensics Secur 2018;14 (6):1455–70.
[18] Kilgallon Sean, Rosa Leonardo De La, Cavazos John. "Improving the effectiveness and efficiency of dynamic malware analysis with machine learning. In: 2017 Resilience Week (RWS). IEEE; 2017. p. 30–6.
[19] Yousefi-Azar Mahmood, Hamey Leonard GC, Varadharajan Vijay, Chen Shiping. Malytics: a malware detection scheme. IEEE Access 2018;6:49418–31.
[20] Saracino A, Sgandurra D, Dini G, Martinelli F. Madam: effective and efficient behavior-based android malware detection and prevention. IEEE Trans Dependable Secure Comput 2016;15(1):83–97.
[21] Ding Yuxin, Xia Xiaoling, Chen Sheng, Li Ye. A malware detection method based on family behavior graph. Computers & Security 2018;73:73–86.
[22] Udayakumar N, Anandaselvi S, Subbulakshmi T. Dynamic malware analysis using machine learning algorithm. In: 2017 International Conference on Intelligent Sustainable Systems (ICISS). IEEE; 2017. p. 795–800.
[23] Jerlin MAsha, Marimuthu K. "A new malware detection system using machine learning techniques for API call sequences. J Appl Secur Res 2018;13(1):45–62.
[24] Salehi Zahra, Sami Ashkan, Ghiasi Mahboobe. MAAR: robust features to detect malicious activity based on API calls, their arguments, and return values. Eng Appl Artif Intell 2017;59:93–102.
[25] Mehmet Barış, Yaman. A machine learning approach to predict advanced malware. pp, (2019): 1–5.
[26] Bahtiyar Şerif, Yaman Mehmet Barış, Altıniğne Can Yılmaz. A multi-dimensional machine learning approach to predict advanced malware. Comput Netw 2019; 160:118–29.

[27] Budak Hüseyin, Taşabat Semra Erpolat. A modified t-score for feature selection. Anadolu Üniversitesi Bilim Ve Teknoloji Dergisi A-Uygulamalı Bilimler ve Mühendislik 2016;17(5):845–52.
[28] Ki Youngjoon, Kim Eunjin, Kim Huy Kang. A novel approach to detect malware based on API call sequence analysis. Int J Distrib Sens Netw 2015;11(6):659101.
[29] Malicia Project, http://malicia-project.com/dataset.html. Volume 14, Issue 1, February Pages 15-33, 2015.
[30] VirusTotal, https://www.virustotal.com.2017.

Dr Saud S. Alotaibi received the Bachelor of Computer Science degree from King Abdul Aziz University, Jeddah, KSA, in 2000, the Master's degree in Computer Science from King Fahd University, Dhahran, KSA, in May 2008, the Ph.D. degrees in Computer Science from Colorado State University, Fort Collins, USA, in August 2015. From January 2009 to 2010, he worked as a Deputy of the IT Center for eGovernment and Application Services at Umm Al-Qura University, Makkah, KSA. Currently, he is an assistant professor with the Department of Information Systems, College of Computer and Information Systems. His-current research interests include Emotional Intelligence, Data Mining, Natural Language Processing, Machine Learning, Deep Learning, Computer Networks, Wireless Sensor Networks, and Network Security.