

Fisher exact Boschloo and polynomial vector learning for malware detection

Sheelavathy Veerabhadrapa Kudrekar, Udaya Rani Vinayaka Murthy

Department of Computer Science and Engineering, REVA University, Bengalore, India

Article Info

Article history:

Received Apr 8, 2022

Revised Sep 27, 2022

Accepted Oct 1, 2022

Keywords:

Behavior-based network feature

Content-based network feature

Malware detection

Polynomial regression

Support vector learning

ABSTRACT

Computer technology shows swift progress that has infiltrated people's lives with the candidness and pliability of computers to work ease shows security breaches. Thus, malware detection methods perform modifications in running the malware based on behavioral and content factors. The factors are taken into consideration compromises of convergence rate and speed. This research paper proposed a method called fisher exact Boschloo and polynomial vector learning (FEB-PVL) to perform both content and behavioral-based malware detection with early convergence to speed up the process. First, the input dataset is provided as input then fisher exact Boschloo's test Bernoulli feature extraction model is applied to obtain independent observations of two binary variables. Next, the extracted network features form input to polynomial regression support vector learning to different malware classes from benign classes. The proposed method validates the results with respect to the malware and the benign files. The present research aimed to develop the behaviors to detect the accuracy process of the features that have minimum time speeds the overall performances. The proposed FEB-PVL increases the true positive rate and reduces the false positive rate and hence increasing the precision rate using FEB-PVL by 7% compared to existing approaches.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Sheelavathy Veerabhadrapa Kudrekar

Department of Computer Science and Engineering, REVA University

Bengalore, India

Email: Sheela.kv@reva.edu.in

1. INTRODUCTION

In recent years, the computer system is retained to secure from malware infection which is a substantial ultimatum [1]. There is rapid growth and also intricacy for malware as it is creating a big issue for network and computer security [2]. A hypervisor content-based malware detection method uses the hypervisor [3]. The comparison is done for prevailing the hypervisor content based on the proposed method as it used a new model for a sophisticated crafted monitoring element which was injected into the guest operating system address space [4]. Moreover, the hypervisor in the proposed content-based malware detection method also safeguarded the monitoring component from detection and modification [5]. The performances are evaluated and handled by the guest context finds negligible contributes to the overhead of minimum [6]. However, the developed model focused on detecting malware based on content [7]. A novel behavior-based malware detection method was proposed in [8].

To design this method, a dynamic analysis environment with various behaviors artifacts is extracted such as printable string information (PSI), calls, application programming interface (API) calls, registry changes, operations, and network activities, with the help of count vectorization, string tokens of the vocabulary were created [9]–[12]. Next, a singular value decomposition model was utilized that in turn

minimized the feature matrix dimension. Finally, malware classifiers were trained and therefore results in moderate accuracy [13]–[15]. Despite improvement observed in terms of accuracy, efforts were not made in addressing content-based malware detection [16], [17]. The creation and curating of a large set of useful features consume significant amounts of time [18], [19]. The motivation for malware detection is that the malware detection model concentrates mainly on content and behavior aspects [20].

Leon *et al.* [21] performed a hypervisor-assisted dynamic malware analysis. The developed model detected the malware and the latter significantly showed improvement in performance. The developed model analyzed the malware with respect to the context of the operating system (OS) which made it completely transparent for running OS. The OS component was camouflaged by a hypervisor and analyzed the model to run OS and its applications. However, the model has a hypervisor that failed to meet the transparency. Singh and Singh [22] analyzed the behavioral artifacts using machine learning algorithms for detecting malicious software. The runtime features were extracted to set the dynamic analysis in an environment by Cuckoo sandbox. The primary features were processed to develop a malware classifier that overcame the problem of the existing research problem. The model experimentally tested the inclusion of features that showed improvement in accuracy for the malware classifiers. Vinayakumar *et al.* [23] developed a deep learning model for malware detection using a robust intelligent model of an advanced type. However, the most significant direction for malware detection was an important step to improve safety.

Lu *et al.* [24] developed a hybrid deep-learning model for Android-based malware detection. The android malware detection uses a deep learning model as it combines deep belief network (DBN) and gate recurrent unit (GRU). However, the separate model and traditional machine learning algorithm were used to improve and optimize to reduce the time cost. Xiao *et al.* [25] utilized a deep learning model for behavior graphs (BDLF) to detect malware. However, the stacked autoencoders (SAEs) had inserted one another, and the last layer required classifiers to some extent. Semi-supervised technique for recognizing distributed denial of service in an SD-honeypot network environment has been developed by Sumadi *et al.* [26]. Faeiq and Mijwil [27] proposed the support vector machine (SVM) and artificial neural network (ANN), two techniques for the early identification of cardiac disease. The medical information was obtained from a database maintained by the University of California, Irvine (UCI), and it includes 170 individuals' reports. The analysis' findings validated the SVM technique is ideal use, which yields highly accurate prediction outcomes.

In [28], various machine-learning approaches were put to the test to evaluate how well they identified ransomware attacks. The gain ratio was used as a feature selection approach to choose the top 1,000 features from raw bytes. To obtain considerable ransomware detection accuracy, three separate classifiers that were accessible in the Waikato environment for knowledge analysis (WEKA) based machine learning platform has been investigated. Security concerns and potential cyber-attacks were mentioned in [29] as a result of the utilization of the main prominent VM apps by academic institutions. Additionally, a thorough comparison of the various VM programs from the viewpoint of cybersecurity was done. In contrast, Ojugo and Oyemade [30] suggested using a string match method as part of a deep learning ensemble on a hybrid spam filtering method to normalize noisy features, expand text, and use semantic dictionaries of categorization to train underlying learning algorithms and accurately categorize SMS into legitimate and spam classes. The objective of the survey by Nordin *et al.* [31] was to do a comparative examination of the metaheuristic fuzzy modeling algorithms for phishing attack detection. Apache spark machine learning library (MLlib), that serves as an open source, independent, scalable, and distributed learning library, has been proven by Ali *et al.* [32]. However, the Boruta feature selection method was used in [33] to identify the most crucial features while building a machine learning model.

Convolutional neural networks (CNN) and large short-term memories were combined to provide a method for identifying cross-site scripting (XSS) attacks while taking the cross-site scripting attack into consideration [34]. Long short-term memory (LSTM) decoding, generalization, and tokenization were first used to pre-process the XSS data set, after which word2vec was used to turn the phrases in the XSS payloads into word vectors. After that, train and test word vectors using CNN and LSTM to develop an algorithm that could be applied to a web-based application. By recommending a system that employs feature selection based on an ensemble extra tree classifier technique and machine learning classifier, Alkaaf *et al.* [35] have supplied exploring permission in android applications utilizing ensemble-based extra tree feature selection to investigate the sequence of malware apps interpreting authorizations. A malicious uniform resource locator (URL) detection method employing optimization and machine learning classifiers has been developed by Lee *et al.* [36]. To identify applications with dangerous URLs, the static analysis combined with machine learning has been employed. This integration with other important properties leads to encouraging results and increased detection accuracy in this work. Particle swarm optimization (PSO), a bio-inspired technique has been utilized to optimize the properties of URLs. It demonstrates that naive Bayes and SVM may achieve excellent detection accuracy when detecting dangerous URLs.

From the overall analysis of the above-stated literature, the existing model has a hypervisor that failed to meet the transparency, because it requires the extension for the layer to improve the precision. In order to overcome those issues, the fisher exact Boschloo and polynomial vector learning (FEB-PVL) model is proposed. The major contributions of this research are i) that this research develops the fisher exact Boschloo and the polynomial vector learning model for the detection of content, and behavior-based features; ii) additionally, the integration method is performed for the selection of network features based on the dissimilar features among the benign and malware classes; and iii) the network features utilized in benign and malware classes showed classification among benign and malware classes.

This research paper is structured as: section 2 explains the proposed FEB-PVL model for malware detection. The results evaluated are provided in section 4. The comparative analysis for the precision performance is stated in section 4. And the conclusion of this research work is presented in section 5.

2. FISHER EXACT BOSCHLOO AND POLYNOMIAL VECTOR LEARNING (FEB-PVL)

The FEB-PVL is proposed to design both content and behavioral-based malware detection. The integrated method selects the network features by identifying the dissimilarity between malware and benign classes. As the network features are normally utilized in malware and hardly utilized in benign classes that are more paramount to different malware classes from benign classes an objective function of the proposed method is split into two parts. Figure 1 shows the structure of the FEB-PVL method. As shown in Figure 1, the network anomaly dataset is provided with input at first and then fisher exact Boschloo's test Bernoulli feature extraction model is used. The design of the contingency table extracts both content-based network features and behavior-based network features.

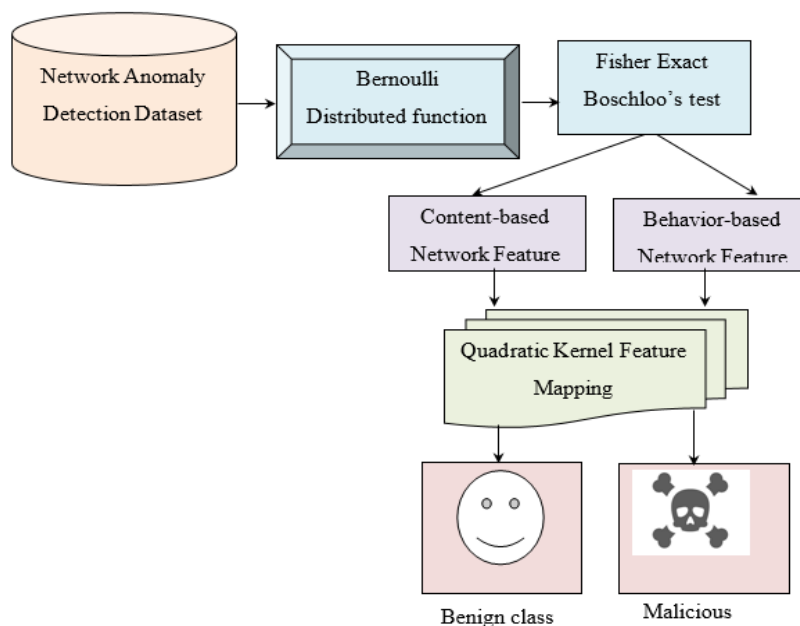


Figure 1. Structure of FEB-PVL method

2.1. Fisher exact Boschloo's test Bernoulli feature extraction model

The procedure of feature extraction plays a very significant role in deciding the cost-effectiveness and accuracy of the malware detection process. It focuses on influencing the feature subset that assists in significantly differentiating between malicious and benign files. In this section, fisher exact Boschloo's test Bernoulli feature extraction model is designed with the purpose of extracting both content-based and behavior-based network features in a timely and efficient manner.

The fisher exact Boschloo's test Bernoulli feature extraction being a statistical significance test is valid for all sample sizes. The exact tests are owed with the significance of divergence from the null hypothesis estimated. Instead of this, an approximation has become an exact limit that shows an increase in sample size with statistical tests arranged numerously. With this advantage, absolute features are extracted in a timely and accurate manner. Comprising of three different features F as shown in Table 1, i.e., basic

features BF , content-related features C and time related traffic features TTF , the objective remains in extracting the relevant features promptly and ensuring an early convergence rate. Let us consider a sample of 2×2 contingency as given in Table 1.

Table 1. Sample contingency table

Row	Columns		Total
	C_1	C_2	
R_1	F_{11}	F_{12}	$F_{11} + F_{12}$
R_2	F_{21}	F_{22}	$F_{21} + F_{22}$
<i>Total</i>	$F_{11} + F_{21}$	$F_{12} + F_{22}$	n

As given in the sample contingency Table 1, F_{ij} represents the feature in row i and column j , $F_{11} + F_{12}$ represent the sum of two features in row i and $F_{12} + F_{22}$ denotes the sum of two features in column j , respectively. Then, the probability of obtaining the values as in the above contingency table based on the hypergeometric distribution function is mathematically expressed in (1)-(3).

$$Prob(R_1 C_1 \rightarrow R_2 C_1) = \frac{\binom{F_{11}+F_{12}}{F_{11}} \binom{F_{21}+F_{22}}{F_{21}}}{\binom{n}{F_{11}+F_{21}}} \quad (1)$$

$$Prob(R_1 C_2 \rightarrow R_2 C_2) = \frac{\binom{F_{11}+F_{12}}{F_{12}} \binom{F_{21}+F_{22}}{F_{22}}}{\binom{n}{F_{12}+F_{21}}} \quad (2)$$

$$Prob(R_1 C_1 \rightarrow R_2 C_1 \cup R_1 C_2 \rightarrow R_2 C_2) = \frac{(F_{11}+F_{21})!(F_{21}+F_{22})!(F_{11}+F_{21})!(F_{12}+F_{22})!}{F_{11}!F_{12}!F_{21}!F_{22}!} \quad (3)$$

From (1)-(3), $\binom{n}{l}$ denotes the network connection vector binomial coefficients and exclamation mark! represents factorial representing with l denoting a 2×2 contingency table, respectively. Then, the probability that network feature vectors are positive in a random selection is based on the significance level from a larger network feature vector set. It is containing elements in total out of which are positive using the hypergeometric distribution function that is mathematically stated as given in (4) and (5).

$$Z_p(F_1, F_0) = \frac{p'_i - p'_0}{\sqrt{p'(1-p) \left(\frac{1}{n_1} + \frac{1}{n_0} \right)}} \quad (4)$$

$$p'_i = \frac{F_i}{n_i}; \quad p' = \frac{F_1 + F_0}{n_1 + n_0} \quad (5)$$

From (4), and (5), p'_i represents the group network connection vector event rates and p'_i denotes the pooled network connection vector event rate p' respectively. Moreover, p_0 and p_1 denote the expected and variance random values of the Bernoulli distribution. The higher the significance level obtained based on hyper-geometric distribution function, the higher is the probability of extracting the features. The pseudo-code representation of hyper-geometric fisher exact Bernoulli feature extraction is given in Algorithm 1.

Algorithm 1. Hyper-geometric fisher exact Bernoulli feature extraction

Input: Features $F = \{BF \cup CF\}$, basic features $BF = \{bf_1, bf_2, \dots, bf_9\}$, content features $CF = \{cf_1, cf_2, \dots, cf_{13}\}$

Output: Precise content-based and behavior-based significant feature $SF = SF_1, SF_2, SF_3, \dots, SF_n$

1: Begin

2: For each Features F

3: For each basic features BF , content features CF

4: Obtain contingency table based on hyper-geometric distribution function as in (1), (2) and (3)

4: Estimate significance level from a larger network feature vector set as in (4) and (5)

5: Return (high significance feature SF)

6: End for

7: End for

8: End

As given in the above hyper-geometric fisher exact Bernoulli feature extraction model, an objective function for the extraction of highly précised information is provided. At the first, a contingency table is used

for the hyper-geometric distribution function for modeling it. With the employment of this hyper-geometric distribution function, the probability of obtaining a certain number of successes for the corresponding network connection vector without the replacement of specific basic feature, content-related, and time-related traffic feature size. Also, it explores the relationship of two Bernoulli distributed random variables by employing fisher exact Boschloo's test via contingency table. The weights are evaluated based on fisher exact Boschloo's test that utilized a table for the behavior-based and content-based features that have to be extracted precisely to ensure a convergence rate better.

2.2. Polynomial regression support vector learning model

With the extracted content-based and behavior-based network features, malware is detected using the polynomial regression support vector learning model. Malware detection refers to the procedure of detecting the malware presence by differentiating whether a software code is malicious or benign. Early malware detection helps the hackers from malicious users to intercept the data packets to compromise the information. By using the polynomial regression support vector learning model, the grouping of significant network features is performed via polynomial kernel, therefore it maximizes the margin of detection and classification. Let us consider training data (a_i, b_i) for $i = 1, 2, 3, \dots, n$, where each a_i is a real-valued significant feature vector of length n and each $b_i \in \{-1, +1\}$ refers to the equivalent class label. This is mathematically formulated as (6).

$$TD = \{(a_i, b_i) | a_i \in SF, b_i \in \{-1, +1\}\} \quad (6)$$

In (6), a_i represents the training sample consisting of n samples with b_i representing the binary label related to the i^{th} vector. Then, for a degree d polynomials, the polynomial kernel is mathematically formulated as given in (7), where SF and c refer to the vectors in the input space (i.e., significant features) and the parameter p trading of higher order versus lower order. On the other hand, as a kernel, K refers to the innermost consequence in a feature space based on the mapping parameter φ as formulated in (8).

$$K(SF, c) = (SF^T c + p)^d \quad (7)$$

$$K(SF, c) = \langle \varphi(a), \varphi(c) \rangle \quad (8)$$

The weight mapping between input space and feature space for each significant feature is then updated as given in (9), where SF represents the significant feature, RS is the random sample selected from the set of significant features, $NN(SRS)$ denotes the nearest neighbor from the same random sample whereas $NN(DRS)$ represents the nearest neighbor from different samples respectively, the weight update of each feature is obtained. Therefore, the polynomial regression support vector learning is proposed for estimating the relationship between two network features using quadratic kernel feature mapping and a statistical pattern.

$$W(SF) = W(SF) - \frac{Diff[SF, RS, NN(SRS)]}{n} + \frac{Diff[SF, RS, NN(DRS)]}{n} \quad (9)$$

Therefore, to find the benign classes and attack classes, speeding up the process is performed for malware detection using (10) and (11). Based on the weight update and regrouping for a parameter p regression kernel results are obtained from (10) and (11). Then, for a hyperplane parametrized by vector w and a parameter p the classification rule between malicious class and benign class for early malware detection is mathematically expressed as given in (12).

$$K(SF, c) = (\sum_{i=1}^n SF_i c_i + p)^2 \quad (10)$$

$$= \sum_{i=1}^n (SF_i^2)(c_i^2) + \sum_{i=1}^n \sum_{j=1}^n (\sqrt{2SF_i SF_j})(\sqrt{2c_i c_j}) + \sum_{i=1}^n (\sqrt{2p SF_i})(\sqrt{2p c_i}) \quad (11)$$

$$CR = h_w(SF) = \text{Sign}(w^T SF + c) \quad (12)$$

From (12), the function $h_w(SF)$ precisely classifies the training data or the significance feature, therefore, improving the margin of detection and classification. The given polynomial quadratic support vector learning for malware detection algorithm has the objective to detect the malware at an early stage. In a timely manner, a polynomial regression function is first modeled via quadratic kernel feature by mapping the

input space with the feature space. By performing this mapping, the relationships between two network features (i.e., content-based and behavior-based) are determined with which highly significant results (i.e., network features) are considered as a malicious class and less significant results (i.e., network features) are contemplated as benign classes. Finally, the vector kernel results, hyperplane is parameterized therefore improving the margin of detection and classification. The pseudo code representation of polynomial quadratic support vector learning for malware detection is given in Algorithm 2.

Algorithm 2. Polynomial quadratic support vector learning for malware detection

Input: Features $F = \{BF \cup CF\}$, basic features $BF = \{bf_1, bf_2, \dots, bf_9\}$, content features $CF = \{cf_1, cf_2, \dots, cf_{13}\}$, Threshold TP
Output: Timely and early malware detection
1: Initialize high significance feature SF
2: Initialize threshold values $ThreshHErr$, $ThreshDur$, $ThreshNFile$, $ThreshFLogins$
3: Begin
4: For each high significance feature SF as input
5: Formulate polynomial kernel as in (7)
6: Perform quadratic kernel feature mapping as in (8) and (9)
7: Evaluate weight update and regrouping as in (10) and (11)
8: Classify between malicious class and benign class as in (12)
9: Return (classified results)
10: If $CR < ThreshHErr$ then no attack
11: Else attack type is DDoS
12: End if
13: If $CR < ThreshDur$ then no attack
14: Else attack type is Probing
15: End if
16: If $CR < ThreshNFile$ then no attack
17: Else attack type is U2R
18: End if
19: If $CR < ThreshFLogins$ then no attack
20: Else attack type is R2L
21: End if
22: End for
23: End

3. EVALUATION OF PERFORMANCE PARAMETERS

In the results section of the research, the underlying definitions of the performance parameters employed assess the proposed method for malware detection. The proposed method is conducted for fisher exact Boschloo and polynomial vector learning to detect the malware attack. Based on the behavior and content of the network features, the implementation is done in Java.

Fair comparisons for a similar number of connections or counts are utilized in the network for anomaly detection [3]. An in-depth analysis is proposed to be made for malware detection performance using FEB-PVL with respect to three different parameters, attack detection time, attack detection overhead, and precision for estimating the malware detection performance. Attack detection time symbolizes the time consumed in detecting different types of attacks, namely, DDoS, probing, U2R, and R2L, respectively. This is mathematically formulated as given in (13).

$$ADT = \sum_{i=1}^n Count_i * Time (AD[C_{based} + B_{based}]) \quad (13)$$

From (13), the attack detection time ADT is measured based on the number of connections taken into consideration for simulation $Count_i$ and the time consumed in detecting the content-based C_{based} and behavior-based B_{based} network feature, respectively. It is measured in terms of milliseconds (ms). The second parameter of significance is the attack detection overhead. A small portion of memory is said to be consumed during the attack detection process and this is mathematically formulated as given in (14).

$$ADO = \sum_{i=1}^n Count_i * Mem (AD[C_{based} + B_{based}]) \quad (14)$$

From (14), the attack detection overhead ADO is measured based on the connections involved $Count_i$ and the memory consumed in detecting content-based C_{based} and behavior-based network features B_{based} . It is measured in terms of kilobytes (KB). Finally, precision quantifies the number of positive class predictions that belong to the positive class. This is mathematically formulated as given in (15), where the precision rate P is estimated based on the number of malware cases identified as malware Num_{M-M} and the number of benign cases identified as malware Num_{B-M} respectively.

$$P = \frac{Num_{M-M}(TP)}{Num_{M-M}(TP) + Num_{B-M}(FP)} \quad (15)$$

3.1. Dataset details

In this section, the detailed analysis of the network anomaly dataset [37] considered for detecting different types of malwares is discussed. The majority of anomaly-based detection techniques have been the subject of extensive research in the intrusion detection field for a very long period, whereas anomaly detection is frequently thought of as a more potent technique in academic research because of its theoretical potential for addressing various attacks. Three different types of network connection vectors are called basic features of each network connection vector; content-related features of each network connection vector are analyzed.

3.2. Performance analysis of attack detection time

Table 2 shows the results of attack detection time using three methods, FEB-PVL, hypervisor content-based malware detection [21], and behavior-based malware detection [22] respectively. From the results, it is evident that FEB-PVL consumes comparatively minimum attack detection time than hypervisor content-based malware detection [21] and behavior-based malware detection [22]. For a detailed comparison of the methods, the indicator line chart is shown in Figure 2.

Table 2. Attack detection time using FEB-PVL, hypervisor content-based malware detection [21] and behavior-based malware detection [22]

Count	Attack detection time (ms)		
	FEB-PVL	Hypervisor content-based malware detection	Behavior-based malware detection
15	6.075	6.375	7.275
30	7.085	10.115	13.315
45	9.325	13.255	19.255
60	11.515	15.125	20.215
75	15.355	18.315	25.325
90	19.215	25.215	34.135
105	21.535	29.315	40.55
120	28.595	40.235	49.215
135	35.455	45.155	53.585
150	41.255	51.255	60.215

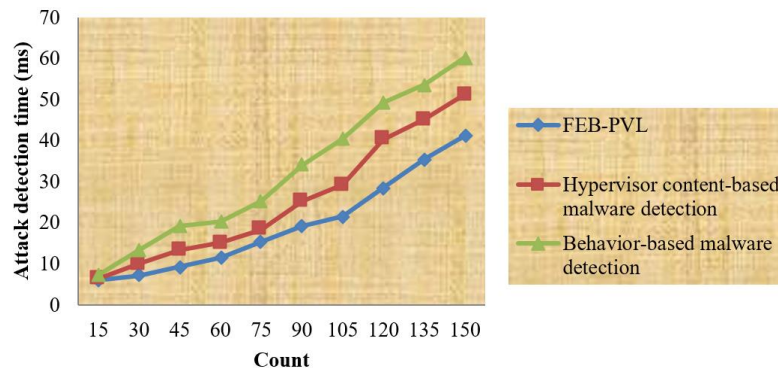


Figure 2. Graphical representation of attack detection time

From Figure 2, performances were compared in terms of average attack detection time for distinct numbers of connections (i.e., count) ranging from 15 to 150. The results obtained show that the attack detection time of the proposed method FEB-PVL is better and rise more steadily than the other methods, [21], [22]. Moreover, it can also be inferred from Figure 2 that the proposed FEB-PVL method consumes less time to detect both the content and behavior types of attacks. Both of the models outperform [21], [22] in terms of both qualities of services and convergence speed. Also from the figure, when the overall network connection vector (i.e., count) increases, the proposed method exhibits more stability. But in the case of [21], [22], the probability of identifying attack detection time becomes robust. Hence, the FEB-PVL method discloses adequate convergence features. The attack detection time is minimized considerably upon the increase in the number of connections.

3.3. Performance analysis of attack detection overhead

Table 3 lists the results of attack detection overhead using three methods, FEB-PVL, Hypervisor content-based malware detection [21] and behavior-based malware detection [22], respectively. From the results, it is evident that FEB-PVL is better than, Hypervisor content-based malware detection [21] and behavior-based malware detection [22] in terms of attack detection overhead. Figure 3 illustrates the average attack detection overhead of the FEB-PVL, hypervisor content-based malware detection [21], and behavior-based malware detection [22].

Table 3. Attack detection overhead using FEB-PVL, hypervisor content-based malware detection [21] and behavior-based malware detection [22]

Count	Attack detection overhead (KB)		
	FEB-PVL	Hypervisor content-based malware detection	Behavior-based malware detection
15	45	60	75
30	60	90	120
45	90	135	180
60	120	180	240
75	225	300	375
90	270	360	450
105	315	420	525
120	360	480	600
135	405	540	675
150	450	600	750

Attack detection overhead is an essential performance parameter as it denotes the detection overhead and also measures to which extent the network connection vectors were busy to detect the malware. The result represents that the attack detection overhead of FEB-PVL is maintained at a high level which means that it has the lowest detection overhead compared with hypervisor content-based malware detection [21] and behavior-based malware detection [22]. Moreover, hypervisor content-based malware detection [21] shows good performance in comparison with behavior-based malware detection [22]. However, the divergence between the proposed method and these state-of-the-art methods, [21], [22] is predominant and hence corroborates the significance of the proposed method based on learning. Therefore, the FEB-PVL method has the adaptability to probe, explore and detect both content-based and behavior-based malware intelligently. The comparison between the proposed method FEB-PVL shows that [22] exhibit less performance than [21]. Also, the proposed FEB-PVL detects both the types of malwares.

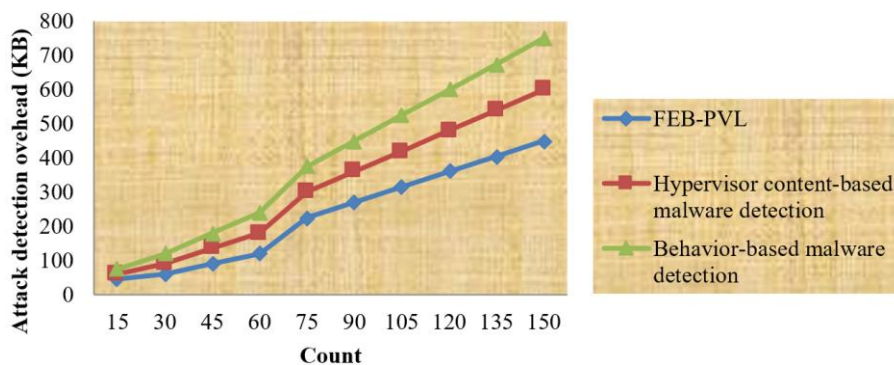


Figure 3. Graphical representation of attack detection overhead

4. COMPARATIVE ANALYSIS OF PRECISION

Finally, Table 4 lists the results of precision rate using three methods, FEB-PVL, hypervisor content-based malware detection [21] and behavior-based malware detection [22] respectively. From the results, it is inferred that with the deployment of the FEB-PVL method, precision is comparatively increased than enhanced heterogeneous earliest finish time based on rule (EHEFT-R) [21] and quantitative risk assessment system (QRAS) [22]. Figure 4 illustrates the precision rate for 150 different network connections.

From Figure 4, it is inferred that the precision rate is neither directly proportional nor inversely proportional to the number of network connection vectors provided as input. In other words, increasing the

number of network connection vectors neither increases the precision rate nor decreases the precision rate. However, the precision rate was higher for the existing methods [21], [22] when compared to the proposed FEB-PVL method. The hyper-geometric fisher exact Bernoulli feature extraction algorithm can efficiently control the drawback of [21], [22], therefore extracting the precise content-related and behavior-related features via hyper-geometric distribution function. This in turn increases the true positive rate (i.e., the number of malicious is identified as malicious and the number of benign is identified as benign cases) and reduces the false positive rate and hence increasing the precision rate using FEB-PVL by 7% compared to [21] and 14% compared to [22] respectively.

Table 4. Precision using FEB-PVL, hypervisor content-based malware detection [21] and behavior-based malware detection [22]

Count	Precision (%)		
	FEB-PVL	Hypervisor content-based malware detection	Behavior-based malware detection
15	0.86	0.8	0.73
30	0.85	0.79	0.73
45	0.83	0.77	0.72
60	0.81	0.76	0.72
75	0.81	0.76	0.72
90	0.81	0.77	0.73
105	0.82	0.78	0.74
120	0.83	0.78	0.74
135	0.85	0.79	0.75
150	0.87	0.79	0.75

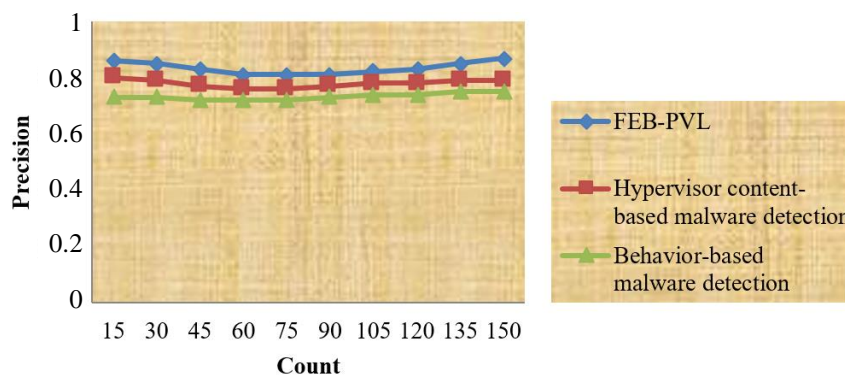


Figure 4. Graphical representation of precision

5. CONCLUSION

Detection of malicious code or malware at an early stage saves the time and money involved in the software engineering process. Malicious code detection in recent years has been designed using learning techniques. In this work, a fisher exact Boschloo and polynomial vector learning (FEB-PVL) performs both content and behavioral-based malware detection with early convergence and speed. First, fisher exact Boschloo's test Bernoulli feature extraction model is designed based on a contingency table and hyper-geometric distribution function. The polynomial regression support vector learning for early malware detection is proposed by employing quadratic kernel feature mapping for significant classification between benign classes and attack classes. The simulation results demonstrate the proposed FEB-PVL method achieved a 7% improvement in precision when compared with existing hypervisor-assisted dynamic malware analysis. Also, comparison simulation results disclosed that the proposed method outperforms recent state-of-the-art malware detection methods in terms of attack detection time, attack detection overhead, and precision rate.

REFERENCES




- [1] X. Gao, C. Hu, C. Shan, B. Liu, Z. Niu, and H. Xie, "Malware classification for the cloud via semi-supervised transfer learning," *Journal of Information Security and Applications*, vol. 55, Dec. 2020, doi: 10.1016/j.jisa.2020.102661.
- [2] X. Huang, L. Ma, W. Yang, and Y. Zhong, "A method for windows malware detection based on deep learning," *Journal of Signal Processing Systems*, vol. 93, no. 2–3, pp. 265–273, Mar. 2021, doi: 10.1007/s11265-020-01588-1.

- [3] V. Syrris and D. Geneiatakis, "On machine learning effectiveness for malware detection in Android OS using static analysis data," *Journal of Information Security and Applications*, vol. 59, Jun. 2021, doi: 10.1016/j.jisa.2021.102794.
- [4] T. Aishwarya and V. Ravi Kumar, "Machine learning and deep learning approaches to analyze and detect COVID-19: a review," *SN Computer Science*, vol. 2, no. 3, May 2021, doi: 10.1007/s42979-021-00605-9.
- [5] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "DL-Droid: Deep learning based android malware detection using real devices," *Computers and Security*, vol. 89, Feb. 2020, doi: 10.1016/j.cose.2019.101663.
- [6] D. Gibert, C. Mateu, and J. Planes, "The rise of machine learning for detection and classification of malware: Research developments, trends and challenges," *Journal of Network and Computer Applications*, vol. 153, Mar. 2020, doi: 10.1016/j.jnca.2019.102526.
- [7] J. Hemalatha, S. Roseline, S. Geetha, S. Kadry, and R. Damaševičius, "An efficient DenseNet-based deep learning model for malware detection," *Entropy*, vol. 23, no. 3, Mar. 2021, doi: 10.3390/e23030344.
- [8] J. Singh and J. Singh, "A survey on machine learning-based malware detection in executable files," *Journal of Systems Architecture*, vol. 112, Jan. 2021, doi: 10.1016/j.sysarc.2020.101861.
- [9] D. Yuxin and Z. Siyi, "Malware detection based on deep learning algorithm," *Neural Computing and Applications*, vol. 31, no. 2, pp. 461–472, Feb. 2019, doi: 10.1007/s00521-017-3077-6.
- [10] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi, "Linear SVM-based android malware detection for reliable IoT services," *Journal of Applied Mathematics*, pp. 1–10, 2014, doi: 10.1155/2014/594501.
- [11] O. Aslan and R. Samet, "A comprehensive review on malware detection approaches," *IEEE Access*, vol. 8, pp. 6249–6271, 2020, doi: 10.1109/ACCESS.2019.2963724.
- [12] Z. Liu, R. Wang, N. Japkowicz, D. Tang, W. Zhang, and J. Zhao, "Research on unsupervised feature learning for android malware detection based on restricted Boltzmann machines," *Future Generation Computer Systems*, vol. 120, pp. 91–108, Jul. 2021, doi: 10.1016/j.future.2021.02.015.
- [13] V. Kouliaridis and G. Kambourakis, "A comprehensive survey on machine learning techniques for android malware detection," *Information*, vol. 12, no. 5, Apr. 2021, doi: 10.3390/info12050185.
- [14] H. Cai, "Assessing and improving malware detection sustainability through app evolution studies," *ACM Transactions on Software Engineering and Methodology*, vol. 29, no. 2, pp. 1–28, Apr. 2020, doi: 10.1145/3371924.
- [15] A. Pektaş and T. Acarman, "Deep learning for effective Android malware detection using API call graph embeddings," *Soft Computing*, vol. 24, no. 2, pp. 1027–1043, Jan. 2020, doi: 10.1007/s00500-019-03940-5.
- [16] G. D'Angelo, M. Ficco, and F. Palmieri, "Malware detection in mobile environments based on Autoencoders and API-images," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 26–33, Mar. 2020, doi: 10.1016/j.jpdc.2019.11.001.
- [17] S. Jeon and J. Moon, "Malware-detection method with a convolutional recurrent neural network using Opcode Sequences," *Information Sciences*, vol. 535, pp. 1–15, Oct. 2020, doi: 10.1016/j.ins.2020.05.026.
- [18] R. U. Khan, X. Zhang, and R. Kumar, "Analysis of ResNet and GoogleNet models for malware detection," *Journal of Computer Virology and Hacking Techniques*, vol. 15, no. 1, pp. 29–37, Mar. 2019, doi: 10.1007/s11416-018-0324-z.
- [19] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K.-K. R. Choo, "An opcode-based technique for polymorphic internet of things malware detection," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 6, Mar. 2020, doi: 10.1002/cpe.5173.
- [20] E. Amer and I. Zelinka, "A dynamic windows malware detection and prediction method based on contextual understanding of API call sequence," *Computers and Security*, vol. 92, May 2020, doi: 10.1016/j.cose.2020.101760.
- [21] R. S. Leon, M. Kiperberg, A. A. Leon Zabag, and N. J. Zaidenberg, "Hypervisor-assisted dynamic analysis," *Cybersecurity*, vol. 4, no. 1, Jun. 2021, doi: 10.1186/s42400-021-00083-9.
- [22] J. Singh and J. Singh, "Detection of malicious software by analyzing the behavioral artifacts using machine learning algorithms," *Information and Software Technology*, vol. 121, May 2020, doi: 10.1016/j.infsof.2020.106273.
- [23] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, and S. Venkatraman, "Robust intelligent malware detection using deep learning," *IEEE Access*, vol. 7, pp. 46717–46738, 2019, doi: 10.1109/ACCESS.2019.2906934.
- [24] T. Lu, Y. Du, L. Ouyang, Q. Chen, and X. Wang, "Android malware detection based on a hybrid deep learning model," *Security and Communication Networks*, pp. 1–11, Aug. 2020, doi: 10.1155/2020/8863617.
- [25] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Mathematical Problems in Engineering*, pp. 1–10, Feb. 2019, doi: 10.1155/2019/8195395.
- [26] F. D. S. Sumadi, C. S. K. Aditya, A. A. Maulana, S. Syaifuddin, and V. Suryani, "Semi-supervised approach for detecting distributed denial of service in SD-honeypot network environment," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 3, pp. 1094–1100, Sep. 2022, doi: 10.11591/ijai.v11.i3.pp1094-1100.
- [27] A. K. Faieq and M. M. Mijwil, "Prediction of heart diseases utilising support vector machine and artificial neural network," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 26, no. 1, pp. 374–380, Apr. 2022, doi: 10.11591/ijeecs.v26.i1.pp374-380.
- [28] B. M. Khammas, "Comparative analysis of various machine learning algorithms for ransomware detection," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, vol. 20, no. 1, Feb. 2022, doi: 10.12928/telkomnika.v20i1.18812.
- [29] N. A. Karim and A. H. Ali, "E-learning virtual meeting applications: A comparative study from a cybersecurity perspective," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 24, no. 2, pp. 1121–1129, Nov. 2021, doi: 10.11591/ijeecs.v24.i2.pp1121-1129.
- [30] A. A. Ojugo and D. A. Oyemade, "Boyer Moore string-match framework for a hybrid short message service spam filtering technique," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 519–527, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp519-527.
- [31] N. S. Nordin *et al.*, "A comparative analysis of metaheuristic algorithms in fuzzy modelling for phishing attack detection," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 23, no. 2, pp. 1146–1158, Aug. 2021, doi: 10.11591/ijeecs.v23.i2.pp1146-1158.
- [32] A. H. Ali, M. N. Abbod, M. K. Khaleel, M. A. Mohammed, and T. Sutikno, "Large scale data analysis using MLlib," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, vol. 19, no. 5, Oct. 2021, doi: 10.12928/telkomnika.v19i5.21059.
- [33] C. Aroef, Y. Rivani, and Z. Rustam, "Comparing random forest and support vector machines for breast cancer classification," *Telecommunication Computing Electronics and Control (TELKOMNIKA)*, vol. 18, no. 2, Apr. 2020, doi: 10.12928/telkomnika.v18i2.14785.
- [34] R. W. Kadhim and M. T. Gaata, "A hybrid of CNN and LSTM methods for securing web application against cross-site scripting attack," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 21, no. 2, pp. 1022–1029, Feb. 2021, doi: 10.11591/ijeecs.v21.i2.pp1022-1029.




- [35] H. A. Alkaaf, A. Ali, S. M. Shamsuddin, and S. Hassan, "Exploring permissions in android applications using ensemble-based extra tree feature selection," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 19, no. 1, pp. 543–552, Jul. 2020, doi: 10.11591/ijeecs.v19.i1.pp543-552.
- [36] O. V. Lee *et al.*, "A malicious URLs detection system using optimization and machine learning classifiers," *Indonesian Journal of Electrical Engineering and Computer Science (IJECS)*, vol. 17, no. 3, pp. 1210–1214, Mar. 2020, doi: 10.11591/ijeecs.v17.i3.pp1210-1214.
- [37] Anush, "Network anomaly detection," Kaggle. <https://www.kaggle.com/datasets/anushonkar/network-anomaly-detection?select=Train.txt> (accessed Jan. 15, 2022).

BIOGRAPHIES OF AUTHORS



Sheelavathy Veerabhadrappa Kudrekar    pursued M.Tech. degree in computer science and engineering from Jain University. She has 12 years of teaching experience. Her areas of interest and teaching include big data and data analytics, data warehousing, data mining, and advanced database management systems. She is pursuing her Ph.D. at REVA University. She has presented 1 paper at a national conference and published 1 paper in international journals. Her qualifications are B. E., M. Tech., (Ph.D.). She can be contacted at Sheela.kv@reva.edu.in.



Udaya Rani Vinayaka Murthy    received her Ph.D. from the Mother Teresa University, Kodaikanal, Tamil Nadu, India, in 2014, her M.Tech degree from MSRIT, Bangalore, India, in 2009 under VTU, and her B.E. degree from SJCE, Mysore, India, in 1994 under VTU. She started her career as a lecturer in Govt. CPC Polytechnic, Mysore for one year. She served as a lecturer in various colleges during her career such as NIE-Mysore, SMSG JAIN COLLEGE-Bangalore, RBANM's College-Bangalore, Nagarjuna College of Management-Bangalore for five years, as HOD and assistant professor in Sambhram Institute of Engineering-Bangalore for two years in the meanwhile she got BEST Lecturer Award. Now she has been serving at REVA University as a senior associate professor since 2011. She is having a total of 22 years of experience. She has published more than 60 journal papers in national and international journals. Her research area includes data mining and warehousing, machine learning, big data analytics, big data and Hadoop, and genetic algorithms. She is currently guiding 8 students through their research at VTU and REVA University. Currently, she is working as a senior associate professor in School of C&IT, as Head of IPR, and as a PG coordinator at REVA University. She can be contacted at udayarani.v@reva.edu.in.