Anomaly Based Worm Detection Using Deterministic Finite Automata and Efficient Keygraph Technique

Leelavathi B
Sri Jayendra Saraswathy CAS
India
researchsolutions62@gmail.com

Rajesh Babu M
Karpagam College of Engineering
India

ABSTRACT: *Nowadays, detecting unknown computer worms is a challenging task. Anti-virus is the feasible solution for this problem but anti viruses that contains signature based methods are clueless against new unknown worms and it cannot detect new worms until it is updated with new worm signatures. This paper focuses on improving the accuracy of detecting the unknown worm while reducing the feature set extracted using DFA. Gain ratio method is implemented to reduce the amount of features. Keygraph and data crystallization classifier is proposed for classifying the unknown attacks. The dataset is collected from the VX Heaven virus collection website which contains the sources and samples of the virus collection. Extensive experiment was performed for testing the feasibility of detecting unknown worms. The evaluation results shows that by applying classification algorithms, the detection ratio obtained for specific unknown worms exceeds 98%.*

Keywords: Frequency Weight Extraction, Gain Ratio, Classification Algorithms, Worm Detection

Received: 26 October 2017, Revised 25 November 2017, Accepted 8 December 2017

DOI: 10.6025/jdp/2018/8/2/63-73

1. Introduction

The ever increasing rate of malcode attacks on computers, computer networks, Internet and smart phones has prioritized the need for hostile measures to counter the malware threat. As a result, malcode analysis and detection has been an active area of research lately, and a large group of methods have been proposed in this field using concepts such as graph theory (Elhadi et al, 2014 and Hu et al, 2009), machine learning (Rieck et al, 2011 and Eskin et al, 2001), and information visualization (Nataraj et al, 2011 and Saxe et al, 2012), etc.

Among the above-mentioned methods, machine learning based algorithms have gained special attention of the area of malware

research. The main reason behind using these techniques is that they enable detection of a previously unknown threat using models leaned from known malcode. This is a key requirement in today's cyber world where millions of new malcode are reported every day and a large number of the new malcode is obfuscated from existing malcode (Elhadi et al, 2014).

The machine learning based techniques generally require a feature vector representation of malware, where a feature represents a particular malcode behavior that can play a discriminatory role in the classification process. Extracting discriminatory attributes pertaining to malcode and representing them in a way to be effectively used in a machine learning setting is a major challenge in the domain of malcode analysis and detection. Feature extraction in the machine learning paradigm is a prerequisite to every malware detection technique proposed in the literature that employs a machine learning algorithm.

In this paper, feature extraction methods that are commonly used for malcode detection and classification tasks are presented, using a comprehensive dataset of real malcode. Specifically, fixed length feature vectors are obtained by applying different feature extraction methods, and are evaluated by performing malcode classification using KeyGraph and machine learning based classifier on the feature vectors. This paper also adds value by evaluating a novel approach of combining the features extracted through different methods for the purpose of malcode classification.

This paper proceeds as follows. Section 2 briefly discusses the related works. Section 3 describes the methodology adopted for the experiments. Section 4 discusses the results due to experimentation and finally section 5 discuss about conclusion and future enhancement**.**

2. Related Works

S. Divya et al. (2014) proposed Deterministic Finite Automata (DFA) with Delayed Dictionary Compression (DDC) that scans every incoming packet in depth to detect the payload occurrence of those affecting the network. Regular expressions are analyzed and for matching regular expressions, DFA-based pattern matching is developed to detect payload. To achieve better matching speed, DDC is combined withDFA.DDC algorithm provides better increased speed links and minimizes decoding latency.

Liu et al. (2005) performed a study about eleven years ago in which three feature extraction methods were addressed. A recent comparison of feature extraction methods can be found in (Ranveer et al, 2015) but it focuses mainly on a survey of such methods and falls short of evaluating them on some benchmark data.

One of the few approaches employing hidden Markov model as a feature extraction method was proposed (Annachhatre et al, 2014) who used opcode sequences extracted from various compilers and virus generators to train HMMs. Opcode sequences from malcode samples were then scored against the HMMs to generate the feature vectors which were then used for clustering of malcode samples. Imran et al. (2015) varied this method by modeling malicious behavior instead of compiler behavior. In their approach, malcode behavior was represented as sequence of system calls which were used to train separate HMMs for different malware families. Known malcode were evaluated against the malcode group HMMs and the resulting score vectors were used to classify unknown malcode using discriminative classifier.

Robert Moskovitch et al. (2008) detect the unknown worms using machine learning methods based on their behaviors on the host. Worm malcodes are detected and feature set is reduced using feature selection methods. Classification accuracy is achieved by minimizing the false positive rate. The learning algorithms are applied for the feature subset reduction.

Nir Nissim et al. (2012) applied Support Vector Machines (SVM) for classifying the unknown Internet worms, which combines active learning selective sampling for its effective performance. Different kernel functions are applied with the SVM classifier to detect unknown worms based on their behavior in the host environment. Mean detection accuracy is achieved by the technique, sustaining the false positive rate at low level.

3. Proposed Methodology

3.1 Behavioral Signatures – PCA

This subsection briefly describes the principal component analysis (PCA) for feature analysis, which has been widely used in diverse applications (C.M. Bishop, 2006). Let the training data sets be organized as an $(M \times N)$- dimensional data matrix, where

M denotes the number of training data sets and N represents the length of each data set.

Let X be the centered version of the original $(M \times N)$ data matrix, where each row of X denotes data sample x with zero mean. In behavior analysis application, M is often smaller than N, i.e., the number of training samples is lesser than the length of time-series. Hence, it is numerically efficient to analyze the $(M \times M)$ matrix $(1/M)\, X.X^T$ that has the same nonzero eigenvalues as the $(N \times N)$ computed covariance matrix $(1/M)\, X^T X$ (C.M.Bishop, 2006).

Let v_i be the normalized eigenvectors of the real symmetric matrix $(1/M)\, X.X^T$ corresponding to the (real positive) eigenvalues λ_i that are arranged in the descending order of magnitude, $\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_M$. Let '$m$' be the smallest integer such that $\Sigma_{i=1}^{m} \lambda_i > \eta\, \Sigma_{i=1}^{m} \lambda_i$ where $1 \leq m \leq M$, where η is a threshold parameter that is a real positive fraction nearer to one. The respective (normalized) eigenvectors u_i in the original data space are obtained in terms of v_i and λ_i as follows (C.M.Bishop, 2006 and Mallapragada et al, 2012):

$$u_i = \frac{1}{\sqrt{M\lambda_i}} (X^T)\, V_i, i = 1, 2, ...m \tag{1}$$

The PCA projection matrix $W \in R^{N \times m}$ is then obtained by grouping the computed eigenvectors as follows:

$$W = [u_1 u_2 ... u_m] \tag{2}$$

The PCA-based feature vector $p \in R^{1 \times m}$ for a given (train or test) $x \in R^{1 \times N}$, is then computed by projection as follows:

$$p = xW \tag{3}$$

Algorithm for PCA-based feature analysis is presented below.

Algorithm: Principal component analysis for feature analysis

Input: Training data sets $xj \in R^{1 \times N}, j = 1 \ldots M$; Threshold parameter $\eta \in (0, 1)$; and test data set $x \in R^{1 \times N}$

Output: Feature vector $p \in R^{1 \times m}$ for x

Step 1: Build the "centered version" training data matrix $X \in R^{M \times N}$, where each row x_j has zero mean.

Step 2: Calculate the matrix $S = (1/M)\, X.X^T$

Step 3: Calculate the normalized eigenvectors $\{v_i\}$ of S with their respective eigenvalues $\{\lambda_i\}$ in the decreasing magnitude order

Step 4: Calculate the normalized eigenvectors $u_i = \frac{1}{\sqrt{M\lambda_i}} (X^T)\, V_i$

Step 5: Determine the smallest $m \leq M$ such that $\Sigma_{i=1}^{m} \lambda_i > \eta\, \Sigma_{i=1}^{m} \lambda_i$

Step 6: Build $(N \times m)$ projection matrix $W = [u_1, u_2 ... u_m]$

Step 7: Create $(1 \times m)$ reference patterns $p = xW$

3.2 Feature Extraction using DFA

The natural technique used for regular expressions are finite automata and it is categorized into two types such as, Deterministic Finite Automata (DFA) and Non-deterministic Finite Automata (NFA). All transitions are deterministic in DFA and each transition leads to only one state. The analysis of REs and establishing memory-adept DFA-based solutions that support high speed processing are reviewed. Transitions are non-deterministic in NFA and each transition results in subsets of states.

DFA consists of a definite set of i) input symbols, represented as Σ, ii) definite set of states and iii) a transition function, denoted

as δ. Σ consist of 2^8 symbols from extended ASCII code. δ gets the start state q0 and argument denoted by input symbol and enters the state. Every single transition leads to only one active state. REs compare the packet with the pattern in the list. When the pattern is matched, they split as states.

For the malcode scanning, regular expressions and automata theory are directly applied. In malcode scanning, input codes or substrings of the input that enter the network are compared with regular expression patterns. DFA faces complexity in recognizing all substring matches as it don't have any knowledge of start and end positions of substrings. Exhaustive and nonoverlapping matching styles can be performed to complete the matching process with DFA for all substrings.

In exhaustive matching, pattern compares all the input substrings taken for comparing and determines a set of results for the complete regular expression pattern and given input stream. Consider for a given pattern *cb*"and input *cbbb*, the result will be three matches such as *cb, cbb,* and *cbbb.*

Let *M* be a function from a pattern *P* and a string *S* to a power set of *S* such that

$$(P, S) = \{\text{substring } S\text{' of S} \mid S\text{'is accepted by the DFA of P}\} \quad (4)$$

Exhaustive matching is costly and it is unnecessary to match every substring. To overcome this drawback, nonoverlapping matching is used.

For the matching process in nonoverlapping approach, let *M* be a function from a pattern *P* and a string *S* to a power set of *S* such that

$$(P, S) = \{\text{substring } S_i \text{ of } S \mid \forall S,\ S\,j \text{ accepted by the DFA of } P,\ S_i \cap S_j = \varphi\} \quad (5)$$

From the input strings, this matching process reports all nonoverlapping substrings that match the pattern appear in various locations of the input. Consider, given pattern *cb*" and input *cbbb*, the report generated is only one and even the prefix "*cb*" overlapped three times. Nonoverlapping matching for malcode provides better analyzing of worms found in the code. This matching results in lack of memory-efficient DFAs.

In order to handle pattern substring matching, DFA creates one pass search execution model. DFA created explicitly for extended patterns that match the pattern with the input anywhere. Instead of scanning from first to last, DFA can begin its substring matching at various positions of the input.

3.3 Frequency Feature Extraction

In this feature extraction method, the count of occurrence of a dictionary term in the sequence is used instead of just its presence or absence (Santos et al, 2013). Mathematically, this vector can be denoted as $VSf = (fs1, fs2 \ldots fsn)$, where fsi is the frequency of the ith element of $\mathbb{S}$ in S.

3.4 Frequency Weight Feature Extraction

Frequency weighting methods such as *term frequency-inverse document frequency* (TF- IDF) have also been employed to generate feature vectors from sequences (Marian et al, 2012). TF-IDF is a statistic widely used in information retrieval and text mining domains for calculating frequency based weights for terms in a document in order to assess their relative importance.

Frequency weighting vector for *S* will be $VSfw = (fws1, fws2, \ldots, fwsn)$, where $fwsi$ represents the frequency weight for the ith element of $\mathbb{S}$ as computed over S.

3.5 Gain Ratio

Gain ratio (GR) was proposed by Quinlan in the framework of decision trees (Mitchell, 1997). It was designed to overcome intolerance in the information gain (IG), and quantify the expected reduction of entropy occurred due to the partitioning of examples to a chosen feature.

Given the measure entropy E(S) of the impurity in collected codes, it is likely to measure the feature effectiveness in classifying

the training data.

Equation 7 denotes the entropy of a set of items S, depending on C subsets of S, denoted by S_c. IG assess the expected reduction of entropy induced according to attribute A, where V denotes the set of possible values of A, and is shown in Equation 6. These equations refer to discrete values and can be extended to continuous valued attribute.

$$IG(S, A) = E(S) - \sum_{v \in V(A)} \frac{|S_v|}{|S|} xE(S_v) \tag{6}$$

$$E(S) = \sum_{c \in C} - \frac{|S_c|}{|S|} . log_2 \frac{|S_c|}{|S|} \tag{7}$$

The IG measure favors only features with high variety of values. Gain Ratio overcomes this by considering how the feature splits the data (Equations. 8, 9). S_i are d subsets of examples developed from dividing S by the d-valued feature A.

$$GR(S, A) = \frac{IG(S, A)}{SI(S, A)} \tag{8}$$

$$SI(S, A) = - \sum_{i=1}^{d} \frac{|S_i|}{|S|} . log_2 \frac{|S_i|}{|S|} \tag{9}$$

3.6 Feature Ensemble Ranking

Instead of selecting features based on of the feature selection methods, one can use the ensemble strategy (Lior et al., 2007 and Rokach et al., 2008) which combines the features subsets that are produced from various features selection methods.

Particularly, we incorporate several methods by averaging the features ranks as shown in Equation 10:

$$Rank(f_i) = \frac{\sum_{j=1}^{k} rank^j(f_i)}{k} \tag{10}$$

Where f_i denotes a feature, *filter* is one of the k filtering (feature selection) methods.

4. Classification using KeyGraph and Data Crystallization

The proposed Keygraph and data crystallization mechanisms are used to classify the known and unknown worms in the network. KeyGraph can extract key points of the data and map the relations among them as an intuitionistic graph. The lines between nodes in KeyGraph denote the relations among the data and quantify the degree of tightness between the nodes.

Based on the frequency feature extraction and frequency weight extraction, the following key terms are extracted.

$$key(w) = 1 - \prod_{g \subset G} \left[1 - \frac{based(w, g)}{neighbors(g)}\right]$$

$$based(w, g) = \sum_{s \in D} |w|_s \, |g - w|_s$$

$$neighbors(g) = \sum_{s \in D} \sum_{w \in s} |w|_s \, |g - w|_s$$

where $|g|_s$ is the number of times a cluster g occurs, $key(w)$ gives a measure of how often a term w occurs near a cluster of high-frequency terms.

Keygraph mechanism classifies the known attacks from the trained dataset based on the following classification steps.

4.1 Classification

1. Build a set of keygraphs whose vertices are the detected keypoints.

2. Extract feature vectors from each keygraph.

3.Apply the classifier to the feature vectors in order to decide if each tested keygraph is sufficiently similar to a trained keygraph.

4.2 Data Crystallization

KeyGraph can only state known features of a data set; it cannot state unknown data sets. In fact, if there is an unknown attack, KeyGraph detection cannot classify it accurately. A data crystallization algorithm can state the problem of unknown dataset and make accurate decisions. Implementing this method, unknown data can be classified based on the analysis of known data.

However, KeyGraph is used for calculating the unknown data set. A KeyGraph can be obtained, and "virtualized item" can be integrated into KeyGraphas a vertex thatis related to the unknown data. Then, the relation between the "virtualizeditem" and existing vertexes can be established. By inserting unknown data into the known data, the relation between them can be obtained. Confined vertexes from the KeyGraph can be connected in the KeyGraph by iteratively inserting them as "virtualized items".

For normal access, assume that 'μ' is the mean and S is the covariance matrix of the leafnode feature vector. For the current KeyGraph, assume 'θ' is a matrix of the feature data of a novel attack. Let the covariance matrix of the feature vector is represented as H.

Based on the calculation of the Mahalanobis-distance (MD), virtual items should be added into the original dataset. The MD can be calculated based on the following formula:

$$MD = \sqrt{(\theta - \mu)^T . H^{-1} . (\theta - \mu)}$$

where the minimum MD is the threshold L_1, which is the distance from the usual known attack vertex to the normal access central vertex.

After adding the virtual items, the KeyGraph can be reconstructed. Once the iteration is completed, the added virtual items should be confirmed by the existing attack data set, which can optimize the KeyGraph. If it does not match the existing attack data set the virtual item will be removed. Finally, based on the final KeyGraph, the results can be obtained. The flow of data crystallization based classification is shown in the following figure.

5. Experimental Results and Discussion

The performance evaluation of the proposed methodology is discussed in this section. This work is developed in the net beans IDE, the dataset is collected from the VX Heaven virus collection website which contains the sources and samples of the virus collection. The performance of proposed approach is evaluated with the parameter such as precision, recall and detection accuracy. The efficient performance of the proposed work is shown below.

Precision	94.099
Recall	94.44444
Accuracy	98.225

Table 1. Performance evaluation of proposed classifier

Table 1 shows the proposed work performance result. The values obtained for various performance metrics are given.

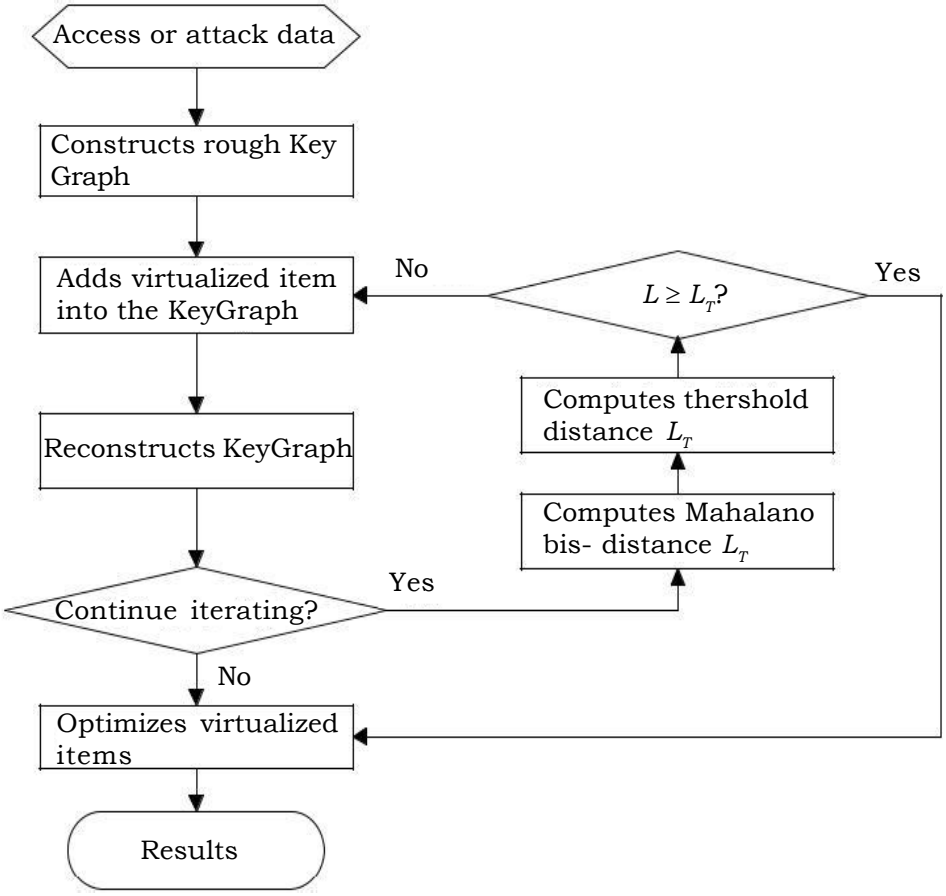

Figure 1. Process of Classification of Unknown attacks

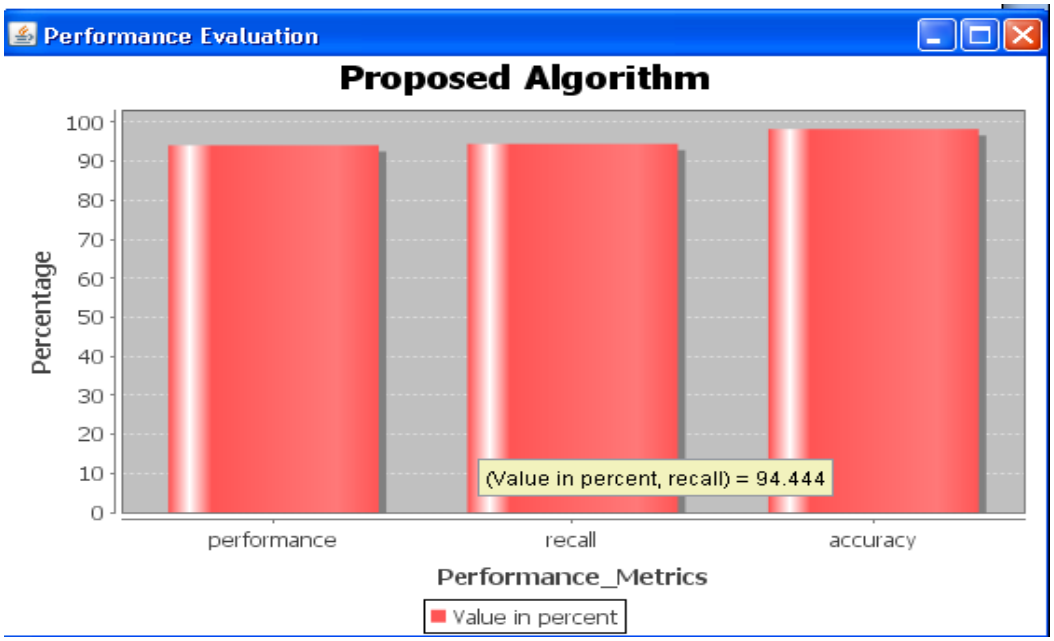

Figure 2. Proposed classifier report

The proposed approach evaluation result is shown in figure 2. The performance metric such as precision, recall value of the proposed keygraph classifier are 94.1% and 94.44%. The unknown worm detection accuracy is 98.225%.

Precision	78.12
Recall	93.01
Accuracy	78.1634

Table 2. Performance Evaluation of Existing SVM classifier

Table 2 shows the performance of existing SVM classifier. The values obtained for various performance metrics are given.

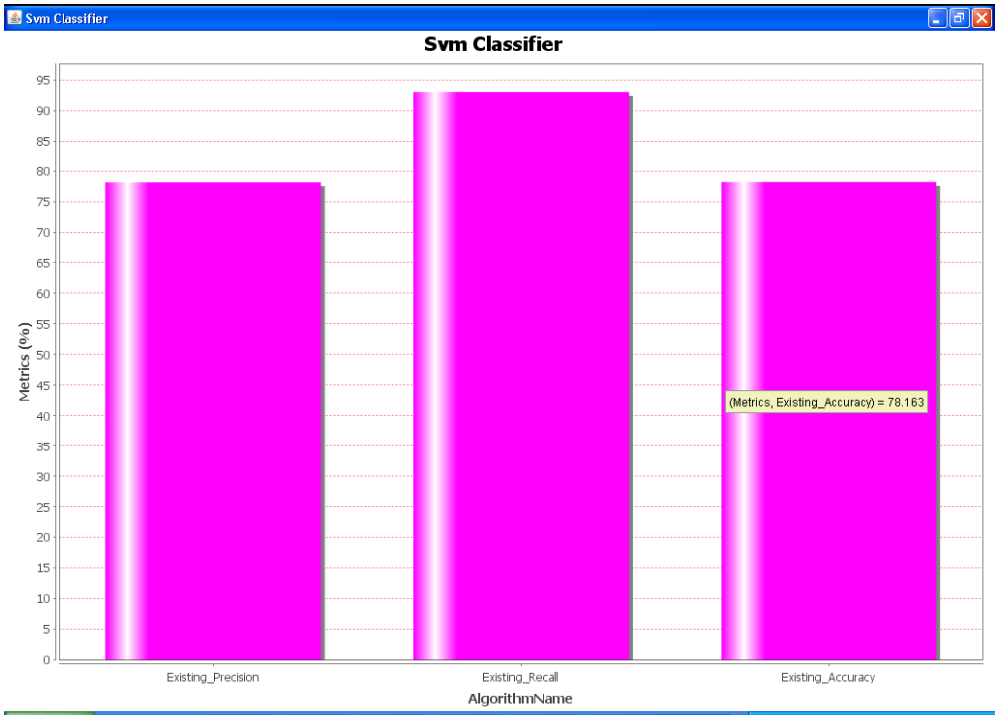

Figure 3. Existing SVM classifier report

The precision, recall value obtained for SVM are 78.12% and 93.01%. The unknown worm detection accuracy is 78.163. The graphical representation of this result is shown in figure 3.

Precision	94.1
Recall	93
Accuracy	98.025

Table 3. Performance Evaluation of Existing Improved ELM classifier

Table 3 shows the performance of Improved ELM classifier. The values obtained for various performance metrics are given.

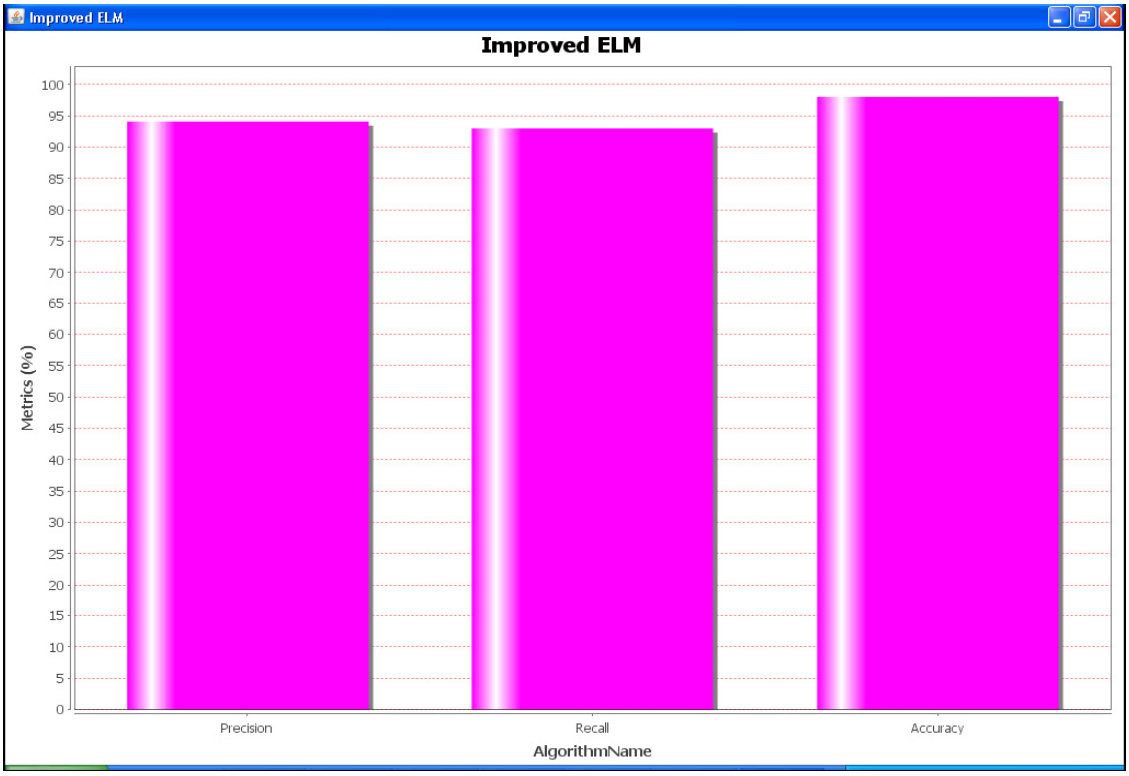

Figure 4. Existing Improved ELM report

The performance evaluation of existing Improved ELM classifier is shown in figure 4. The performance metric such as precision, recall value of the Improved ELM are 94.1% and 93%. The unknown worm detection accuracy is 98.025%.

Metrics	Proposed classifier	SVM	Improved ELM
Precision	94.099	78.12	94.1
Recall	94.44444	93.01	93
Accuracy	98.225	78.163	98.025

Table 4. Performance comparison of Existing and proposed methods

Table 4 shows the performance comparison report of proposed keygraph classifier with existing SVM and Improved ELM classifier. The values obtained for various performance metrics are given.

Figure 5 shows the performance comparison of existing SVM and Improved ELM classifier with the proposed keygraph classifier. The result shows that the proposed classifier achieves better results with respect to the performance evaluation metrics such as precision, recall and accuracy.

6. Conclusion

The objective of the proposed research work is to detect all types of computer worms based on their characteristics. Initially traffic abnormalities of worm characteristics are monitored using Principal Component analysis (PCA) and the feature is extracted using DFA with Frequency feature extraction. Then the frequency weight is extracted and Gain ratio is computed for selecting

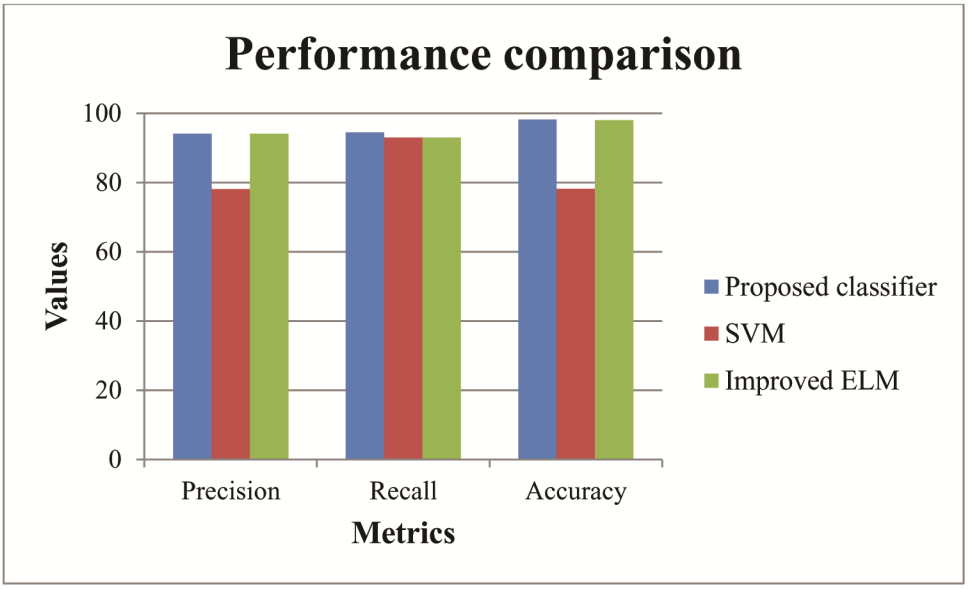

Figure 5. Performance comparison report

the features that are used in classifying worms. The keygraph classifier is used along with data crystallization to detect the known and unknown attack and the final result is obtained. The proposed model produces good results in worm detection. The proposed system produces perfect result with accuracy of 98.23% in detecting unknown worms in the network.

References

[1] Elhadi, A. A. E., Maarof, M. A., Barry, B. I., Hamza, H. (2014). Enhancing the detection of metamorphic malware using call graphs. *Comput & Sec*; 46. 62–78.

[2] Hu, X., Chiueh, T., Shin, K. G. Large-scale malware indexing using function-call graphs, *In:* Proc of CCS'09 2009; 611–620.

[3] Rieck, K., Trinius, P., Willems, C., Holz, T. (2011). Automatic analysis of malware behavior using machine learning. *J Comput Sec*; 19. 639–668.

[4] Schultz, M. G., Eskin, E., Zadok, E., Stolfo, S. J. (2001). Data mining methods for detection of new malicious executables, *In*: IEEE Proc of S&P. 38–49.

[5] Nataraj, L., Karthikeyan, S., Jacob, G., Manjunath, B. (2011). Malware images: visualization and automatic classification. Proc of VizSec'11. 4.

[6] Saxe, J., Mentis, D., Greamo, C. (2012). Visualization of shared system call sequence relationships in large malware corpora. Proc of VizSec'12. 33–40.

[7] Selvaraj, Divya., Ganapathi, Padmavathi. (2014). Packet Payload Monitoring for Internet Worm Content Detection Using Deterministic Finite Automaton with Delayed Dictionary Compression. *Journal of Computer Networks and Communications*, Article ID 206867, 9 p. Hindawi Publishing Corporation.

[8] Liu, A., Martin, C., Hetherington, T., Matzner, S. (2005). A comparison of system call feature representations for insider threat detection, *In:* Proc of IAW'05 2005. 340–347.

[9] Ranveer, S., Hiray, S. (2015). Comparative Analysis of Feature Extraction Methods of Malware Detection. *Int J Comput App*. 120.

[10] Annachhatre, C., Austin, T., Stamp, M. (2014). Hidden Markov models for malware classification, *J Comput Virol Hack Tech*. 1–15.

[11] Imran, M., Afzal, MT., Qadir, MA. (2015). Similarity-based Malware Classification using Hidden Markov Model. Proc of

CyberSec. 133–138.

[12] Moskovitch, R., Elovici, Y., Rokach, L (2008). Detection of unknown computer worms based on behavioral classification of the host, ELSEVIER, *Computational Statistics & Data Analysis*, vol. 52. 4544–4566.

[13] Nissim, N., Moskovitch, R., Rokach, L., Elovici, Y. (2012). Detecting unknown computer worm activity via support vector machines and active learning, Springer, *Pattern Analysis and Applications*, 15 (4) 459-475.

[14] Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer, New York, NY, USA, 2006.

[15] Mallapragada, G., Ray, A., Jin, X.(2012). Symbolic dynamic filtering and language measure for behavior identification of mobile robots, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 42 (3) 647–659.

[16] Santos, I., Brezo, F., Ugarte-Pedrero, X., Bringas, PG. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Info Sci* 231. 64–82.

[17] Marian, T., Weatherspoon, H., Lee, K-S, and Sagar A. (2012). Fmeter: Extracting indexable low-level system signatures by counting kernel function calls. *In*: Middleware 2012 Springer; p. 81–100.

[18] Ross Quinlan, J (1993) C4.5. Programs for machine learning. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

[19] Mitchell, TM .(1997). Machine learning. McGraw-Hill, New York.

[20] Lior, R,. Barak, C., Oded, M (2007). A methodology for improving the performance of non-ranker feature selection filters. IJPRAI 21 (5) 809–830.

[21] Rokach, L., Romano, R., Maimon, O (2008). Negation recognition in medical narrative reports. *Inf Retrieval* 11 (6) 499–538.