CHARACTERISTICS BASED DETECTION OF INTERNET WORMS USING COMBINED MACHINE LEARNING METHODS AND WORM CONTAINMENT

CHAPTER 4

Detection and Containment of Self Propagating Monomorphic Characteristic worms based on Unknown Signatures using the Proposed PMR Method

4.1. Introduction

4.2. Steps of the Proposed Contribution One

- 4.2.1. Analysis and Preprocessing
  - 4.2.1.1. Principal Component Analysis
- 4.2.2. Detection and Classification of Internet Worms
  - 4.2.2.1. Multiclass Support Vector Machine
  - 4.2.2.2. Kernel Function
  - 4.2.2.3. Radical Basis Function Kernel
  - 4.2.2.4. Selective Sampling
- 4.2.3. Containment of Internet Worms
  - 4.2.3.1. Rabin Footprint Algorithm for Malcode Containment

4.3. Flow diagram of the Proposed Contribution One – PMR Method

4.4. Steps involved in the Proposed PMR Method

4.5. Pseudo code of PMR Method

4.6. Experimental Setup and Results

4.7. Chapter Summary

4.1. Introduction

In this chapter, the combined algorithm namely, **P**rincipal Component Analysis with **M**ulticlass Support Vector Machine and **R**abin Footprint Algorithm (**PMR**) is used to detect the self propagating Monomorphic characteristic worms and containment of Malcode programs. The propagation of Self carried worms is straight forward, where the payloads are transmitted within the program by itself. The payloads are the actual worm codes. The payloads that transfer the original worm codes without any change are Monomorphic worms. The payloads existing in the programs are detected by matching with the unknown signatures in the database. The infected programs are blocked.

Initially, all the downloaded executable programs are compared with the unknown signatures in the labeled dataset using Principal Component Analysis (PCA). The labeled dataset contains the signature that affects the Operating System, Hard Disk, Software and Web Browser. The program which has the Malcode lines as in the labeled dataset is detected as malicious. The detected Malcode lines are classified under different classes using Multiclass Support Vector Machine (MSVM).

The classified Malcode lines from the downloaded programs are counted at each round of periodic time set. A threshold (count limit) is set for each periodic time set. Once the Malcode lines reach or exceed the threshold count limit in a round, the detected Malcode program is blocked using Rabin Footprint Algorithm. The method used is explained below.

4.2. Steps of the Proposed Contribution One

The objective of contribution one is to detect and contain Monomorphic characteristic worms based on their unknown signatures with better accuracy. Containment of these Malcode appended programs blocks its malicious entry through network-level emulators. The proposed contribution consists of three steps and they are discussed below in the next section.

Figure.4.1 shows the three different steps of the proposed contribution one namely,

- Analysis and Preprocessing
- Detection and Classification of Internet Worms
- Containment of Internet Worms

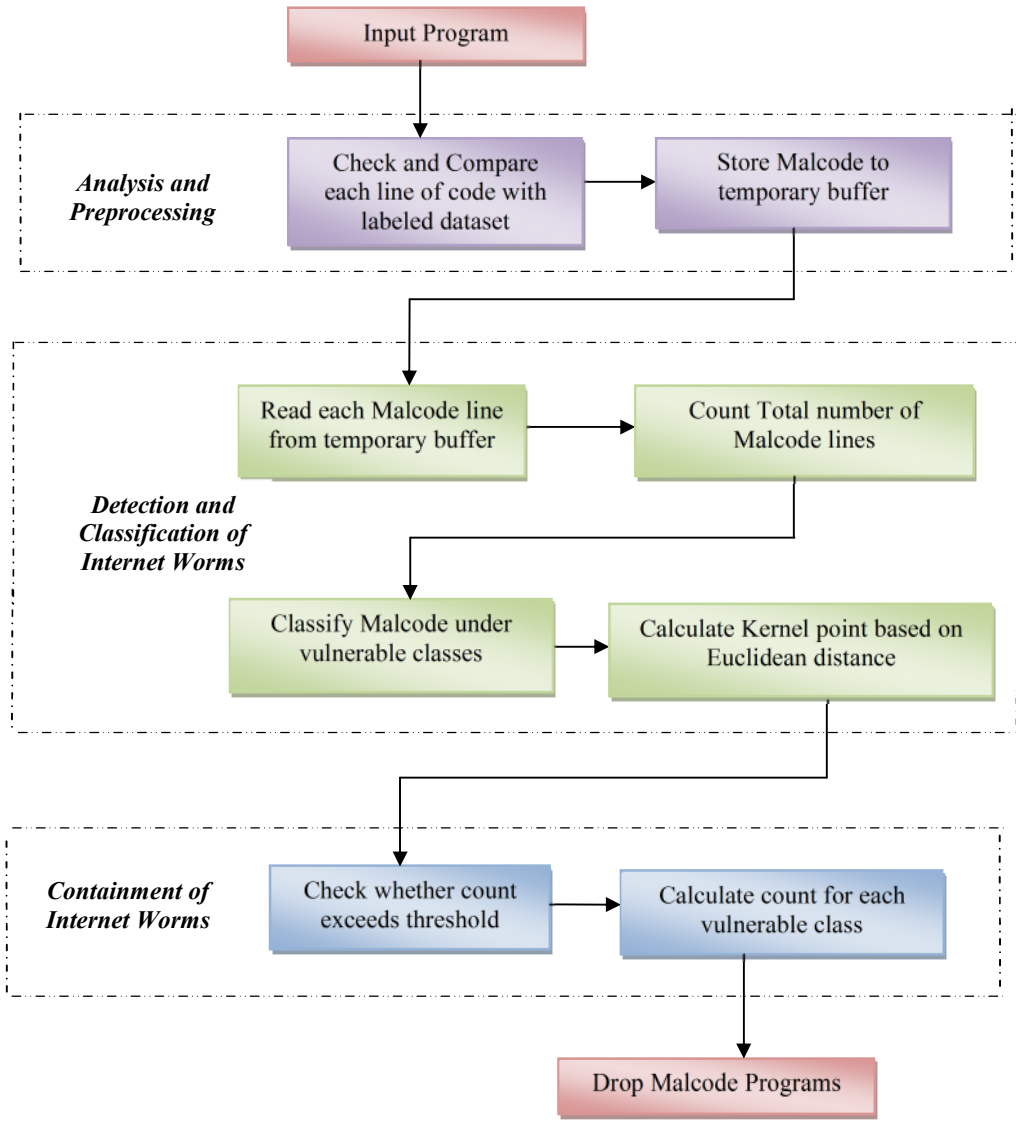

Figure.4.1. Block Diagram of the Proposed Contribution One

To detect the Malcode program and block its infection to the network, a combined method called PMR method is proposed. PMR is a combination of Principal Component Analysis (PCA), Multiclass Support Vector Machine (MSVM) and Rabin Footprint Algorithm (RFA).

4.2.1. Analysis and Preprocessing

To analyze and identify the existence of Malcode lines in the downloaded program, Principal Component Analysis (PCA) algorithm is applied. Here, each line of the program

is checked and compared with the labeled dataset containing unknown Malcode signatures. The analysis done using PCA method is explained below.

4.2.1.1. Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a statistical procedure that

- Uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components.
- Reduces the dimensionality of a sample by finding a new set of variables, smaller than the original set of variables that nonetheless retains most of the sample's information.

Let X indicate an N x P data, where N is the number of data samples, which can be regarded as N realizations of a P-dimensional random vector, which has been normalized to zero-mean and unit variance.

It is a linear transformation R^P from, to an M-dimensional vector space, where $M \leq P$. The optimal linear mapping in the least mean square sense is the one formed by the eigenvectors of the correlation matrix of $S_x.X,$ where

$$S_x = \left(\frac{1}{(N-1)}\right) X^T X$$

Let Z denote the $N \times M$ transformed data matrix. The PCA transforms X to Z by the following equation 4.1.

$$Z = XV_M \qquad - \qquad (4.1)$$

where V_M refers to the $P \times M$ weight matrix, consists of eigenvectors corresponding to the first largest eigen values of the correlation matrix S_x, or V_M corresponds to the first M column vectors of matrix, which is obtained through Singular Value Decomposition (SVD) of a scaled matrix

$$T=(1\sqrt{N-1})X = UDV^T$$

Here, both U and V are unitary matrices, and D is a $P \times P$ diagonal matrix with nonnegative diagonal elements d_i in decreasing order.

Let x denote a row of X (one of the samples), $Z_j = XV_j$ is referred to as the j^{th} principal component (or the PC score). The column vectors v_j of V are called the weight vector of z_j or the j^{th} feature vector. The minimum mean-square error due to dimension reduction is $\sum_i^P = M + 1\, d_i^2$.

Evaluating the statistical significance of a Principal Component (PC) is a significant part of choosing an accurate dimension for PCA to capture the most relevant features. As they are mutually uncorrelated, each coefficient is tested individually using only one random variable statistics to determine whether it is relevant or random.

After identifying the existence of Malcode lines in the downloaded program using PCA, the Malcode lines are classified in to different classes using the Multiclass Support Vector Machine method as in the discussion below.

4.2.2. Detection and Classification of Internet Worms

After identifying the Malcode lines from the downloaded program, they are classified under different classes. The different classes are Operating System, Hard Disk, Software and Web Browser. The classes are created based on their vulnerabilities they infect the network. The Radical Basis Function (RBF) kernel under Support Vector Machine (SVM) classification is applied for classifying the detected Malcodes. The methods like Multiclass Support Vector Machine, RBF kernel function and Selective Sampling are explained in detail in this section.

4.2.2.1. Multiclass Support Vector Machine (MSVM)

A Support Vector Machine is a discriminative classifier formally defined by a separating hyper plane. Multiclass Support Vector Machine classification implemented earlier is called as One-against-all approach, which constructs *' k'* SVM classifiers, in which each and every class is separated from others. The i^{th} SVM is trained with all the training examples of the i^{th} class with positive labels, and all the others with negative labels. It constructs 'k' SVM models where 'k' refers to the number of classes. The i^{th}

value in SVM is trained through all of the examples in the i^{th} class with positive labels, and the remaining with negative labels.

The given one training data $(x_1, y_1) \ldots\ldots, (x_l, y_l)$, where $x_i = R^N, i = 1 \ldots\ldots l$ is an N dimensional vector and $y_i \epsilon \{1, \ldots\ldots, N\}$ is the corresponding class labels. The i^{th} SVM solves the following equations 4.2 – 4.5:

$$min_{w^i, b^i, \xi^i} \frac{1}{2} (w^i)^T w^i + C \sum_{j=1}^{1} \xi_j^1 \left(w^i\right)^T \quad - \quad (4.2)$$

$$\left(w^i\right)^T \emptyset(x_j) + b^i \geq 1 - \xi_j^i, \ if\ y_j = i \quad - \quad (4.3)$$

$$\left(w^i\right)^T \emptyset(x_j) + b^i < -1 + \xi_j^i, \ if\ y_j \neq i \quad - \quad (4.4)$$

$$\xi_j^i \geq 0, j = 1 \ldots\ldots l \quad - \quad (4.5)$$

- x_i in training data is mapped with the high dimensional space by the function ϕ and
- C is the penalty parameter.
- ξ_j^i refers to the training errors
- b denotes biases
- w specifies weight and
- T refers to the term.

By minimizing $(1/2)(w^i)^T w^i$ means maximizing $2/\| w^i \|$ and the margin between two data groups. When grouped data are not separable linearly, there is a penalty term $C \sum_{j=1}^{1} \xi_j^i$ which can reduce the number of training errors. The fundamental conception behind SVM is to search for a balance between the regularization term $(1/2)(w^i)^T w^i$ and the training errors.

After solving the equation 4.6, there are 'k' decision functions

$$(w^1)^T \phi(x) + b^1$$

$$(w^k)^T \emptyset(x) + b^k$$

where w^1 and w^k are the weight of 1^{st} and the $k^{th}\ data$

$$class\ of\ x = \arg\ max_{i=1,\dots,} \left(\left(w^i\right)^T \emptyset(x) + b^i \right) \quad - \quad (4.6)$$

x is in the class that has the largest value of the decision function. Dual problem of equation 4.6 is solved and the number of variables is the same as the number of data in equation 4.6. Table.4.1 below presents the algorithm for Multiclass SVM.

Table.4.1. Algorithm of Multiclass SVM

Input: $\{(\bar{x}_1, y_1), \dots, (\bar{x}_m, y_m)\}$
Intialize: $\bar{T}_1 = \bar{0}, \dots, \bar{T}_m = \bar{0}$
Loop:
Step 1: Choose an example p.
Step 2: Calculate the constants for the reduced problem

- $A_p = K(\bar{x}_p, \bar{x}_p)$
- $\bar{B}_p = \sum_{i \neq p} K(\bar{x}_i, \bar{x}_p)\, \bar{T}_i - \beta \bar{1}_{yp}$

Step 3: Set $\bar{T}_p$ *to be the solution of the reduced problem*

$$min_{r_p} Q(\bar{T}_p) = \frac{1}{2} A_p(\bar{T}_p . \bar{T}_p) + \bar{B}_p . \bar{T}_p$$

Subject to:

$$\bar{T}_p \leq \bar{1}_{yp}\ and\ \bar{T}_p . \bar{1} = 0$$

Output:

$$H(\bar{x}) = \arg max_{r=1} \{\sum_i^k T_{i,r} K(\bar{x}, \bar{x}_i)\}$$

4.2.2.2. Kernel Function

The nonlinear classifiers are applied with the kernel trick to maximum-margin hyperplanes. Each dot product of the linear classifier is converted to a nonlinear kernel function, which allows the algorithm to fit the maximum-margin hyperplane in a transformed feature space. The conversion may be nonlinear and the converted spaces are high dimensional. Hence, the classifier is a hyperplane in the high-dimensional feature space, it may be nonlinear in the original input space. Among the various kernel functions, RBF provides better accuracy in classification and is explained below.

4.2.2.3. Radial Basis Function kernel

In SVM Classification, Radial Basis Function kernel is commonly used. The RBF kernel on two samples **x** and **x'**, represented as feature vectors in some input space, is defined as

$$K(x, x') = exp\left[-\frac{\| x - x' \|^2}{2\sigma^2}\right] + c, \| x - x' \|^2$$

- Identified as the squared Euclidean distance between the two feature vectors
- *C* is constant
- σ is a free parameter

An equivalent definition involves a parameter

$$\gamma = -\frac{1}{2\sigma^2}$$

$$K(x, x') = exp(\gamma \| x - x' \|^2)$$

Since the value of the RBF kernel decreases with distance and it ranges between zero (in the limit) and one (when **x** = **x'**). The kernel's feature space has an infinite number of dimensions; for $\sigma=1$, its expansion is:

$$exp\left[-\frac{1}{2} \| x - x' \|^2\right] = \sum_{j=0}^{\infty} \frac{x^T x'}{j!} exp\left[-\frac{1}{2} \| x \|^2\right] exp\left[-\frac{1}{2} \| x' \|^2\right]$$

In the kernel, Sigma plays a major role. If sigma is overestimated then the exponential behaves linearly and higher-dimensional projection will start to lose its non-linear power. Else if sigma is underestimated, the function will lack regularization and the decision boundary will be highly sensitive to noise in training data.

4.2.2.4. Selective Sampling

Selective sampling is an active variant of learning method in which the learner is allowed to adaptively query the label of an observed data. More misleading instances exist in the training set; the selective sampling is more contributive in filtering the worm instances that are very similar to the non-worm behavior.

The advantage of the approach is the automatic acquisition and maintenance of knowledge based on inductive learning. This avoids the need for a human expert who may not always be available or familiar with generating rules or signatures.

Selective Sampling is a branch of active learning mainly used

- To minimize the learning process
- To improve classifier accuracy by adding up supplementary labels to the dataset
- To achieve a good trade-off between prediction performance and the number of labels
- To reduce the cost of labeling supplementary training data
- To select the most informative instances from data that includes misleading instances

The downloaded programs are analyzed using PCA to check the Malcode lines appended with the programs. Once the Malcode signatures are detected in the programs when comparing each line of code with the labeled dataset containing Unknown signatures, they are classified. The classification is done using RBF kernel function under MSVM. After classifying the Malcodes, the programs with Malcode signatures are blocked from entering into the network level emulators using RFA and they are explained in detail below.

4.2.3. Containment of Internet Worms

The classified Malcode programs are blocked from entering into the network by the containment step. The Rabin Footprint Algorithm (RFA) is applied for blocking the malicious program and is discussed below.

4.2.3.1. Rabin Footprint Algorithm for Malcode Containment

The Rabin Footprint Algorithm is applied to block the malcode program from entering the network. The detected malcode programs are maintained in a list. The list *l,* contains the list of malcode programs and count is set at any time t for each round r. Then, the local prevalence *L(i,j)* is updated for content block of every *j* and *i* refers to the node index in the overlay. If *L(i,j)* is greater than local prevalence threshold and its payload occurrence is greater than local prevalence threshold, the overlay node starts to filter the malcode programs comparing with the unknown signatures.

The filtered Malcode programs are collectively stored as the prevalence content. Then at any time t, the node i maintains a list of (content k, count k) pairs. The content k characterizes the prevalence of the signatures and count k characterizes the sum of prevalence of the signatures. Index for the content blocks is referred as k. The containment method is applied as follows in the table 4.2 below.

Table.4.2. Rabin Footprint Algorithm for Malcode Containment

Maintain l, a list of payloads and their count at any time ,t

Let ($payload_{t-1,k}$, $count_{t-1,k}$) *be all pairs sent to node i in round t − 1*

Let $d_{t,l} = \sum_r count_{t-1,k}$ *represents the sum of the prevalence values of the signature* $payload_k$ *received by node i at round t for one particular payload block k*

Check each $d_{t,i}$ *with the threshold set*

If $d_{t,i}$ *greater than Threshold*

Then $payload_k$ *detected as Malcode worm*

Drop $payload_k$

End if

Repeat until end of payloads in the list

Block payloads detected in list, l

The various steps and the methods used for detection and containment of self propagating and monomorphic payload format worms are discussed in detail above.

4.3. Flow diagram of the Proposed Contribution One – PMR Method

The proposed method PMR detects the malicious codes in the downloaded program using Principal Component Analysis with Multiclass Support Vector Machine and containment using Rabin Footprint Algorithm. The flow diagram of the proposed contribution one is shown in figure.4.2 below.

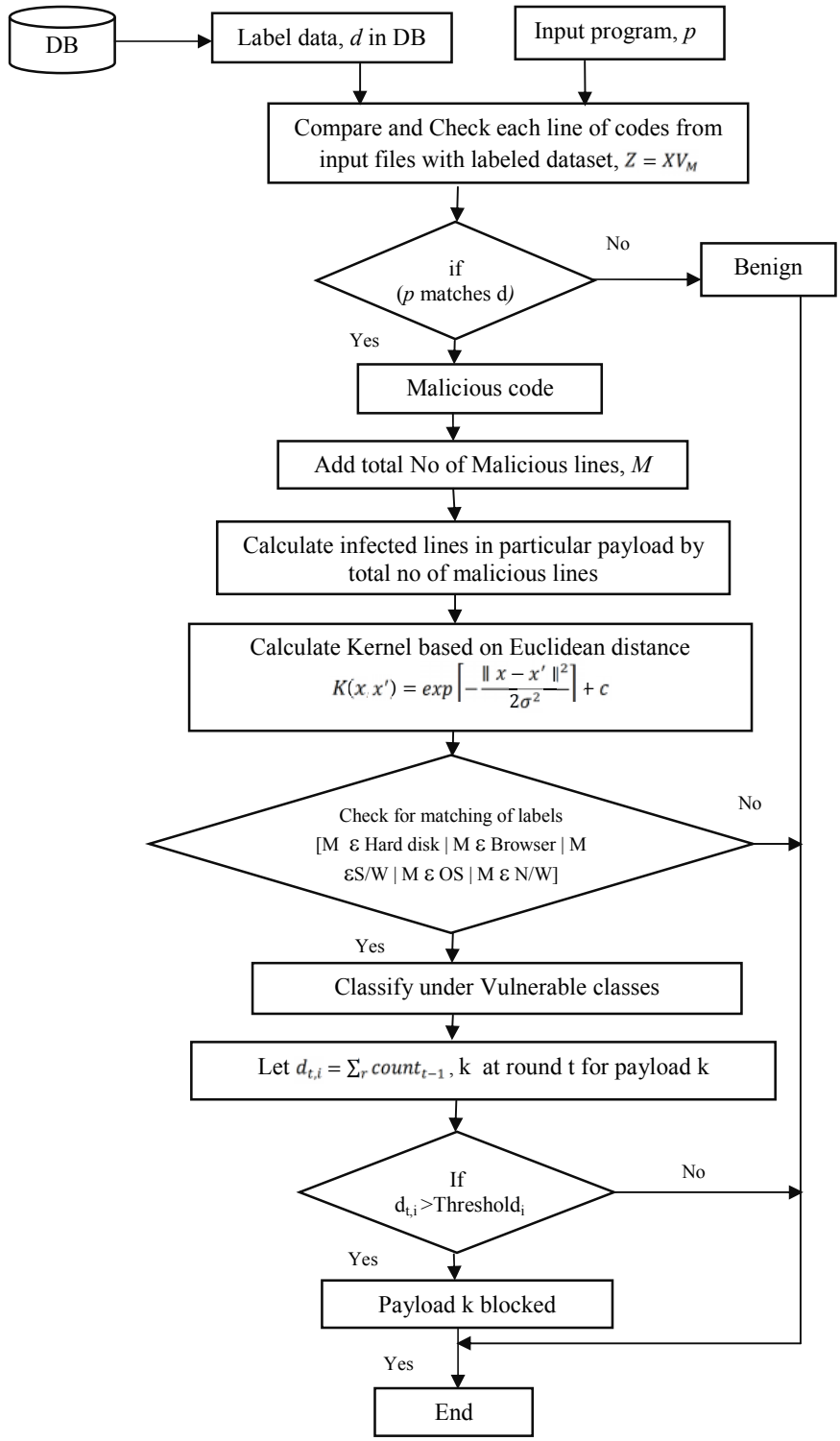

Figure.4.2. Flow diagram of the Proposed Contribution One – PMR Method

4.4. Steps involved in the Proposed PMR Method

The steps used for the detection of Malcode from the downloaded programs and containment of Malcode packets based on unknown signatures are discussed below:

Step 1: *Initiate and maintain I, a list of payloads and their count at any time t. Let* $payload_{t-1}$, *k and* $count_{t-1}$, *k be all pairs sent to node I in round t – 1*

Step 2: *Apply linear transformation and in it Z denote the N x M transformed data matrix and it transforms X to Z by the following equation 4.7:*

$$Z = XV_M \quad - \quad (4.7)$$

Step 3: *Construct 'k' SVM models where 'k' represents the number of classes. With the given training data* $(x_1, y_1) \ldots\ldots, (x_l, y_l)$, *where* $x_i = R^N, i = 1 \ldots\ldots l$ *is an N dimensional vector and* $y_i \epsilon \{1, \ldots\ldots, N\}$ *is the corresponding class labels, the* i^{th} *class solves the following equation 4.8 – 4.11.*

$$min_{w^i, b^i, \xi^i} \frac{1}{2} (w^i)^T w^i + C \sum_{j=1}^{1} \xi_j^1 \left(w^i\right)^T \quad - \quad (4.8)$$

$$\left(w^i\right)^T \emptyset(x_j) + b^i \geq 1 - \xi_j^i, \ if\ y_j = i \quad - \quad (4.9)$$

$$\left(w^i\right)^T \emptyset(x_j) + b^i < -1 + \xi_j^i, \ if\ y_j \neq i \quad - \quad (4.10)$$

$$\xi_j^i \geq 0, j = 1 \ldots\ldots l \quad - \quad (4.11)$$

Step 4: *Apply training data* x_i *with the high dimensional space by the function* ϕ *and C the penalty parameter, there are k decision functions*

$$(w^1)^T \phi(x) + b^1$$

$$(w^k)^T \emptyset(x) + b^k$$

Step 5: *Classify the malcodes based on two samples* ***x*** *and* ***x'****, represented as feature vectors in some input space defined as*

$$K(x, x') = exp\left[-\frac{\| x - x' \|^2}{2\sigma^2}\right] + c$$

Step 6: *Let* $d_{t,i}$ *represent the sum of the prevalence values of the signature* $payload_k$ *received by node i at round t for one particular payload k.*

Step 7*: Block the detected Malcode program, if* $d_{t,i}$ *is greater than Threshold* $payload_k$.

Seven steps discussed above are followed to detect and block the Malcode programs. The algorithm proposed is discussed below.

4.5. Pseudo code of PMR Method

The algorithmic procedures applied to detect the Malcode lines existing in the downloaded programs and to contain those detected payload programs is shown in table.4.3 below.

Table.4.3. Pseudocode of PMR Method

```
for each packet Pi transfer to node i in round t-1
  for i =1 to size(Pi)
       compare each line of code Ci in Pi with trained data S in labeled dataset
       j=1
       if(Ci matches S)
       Then Assign as Malicious code Ci
            B[i][j]= Ci
       else
            benign= Ci
       end if
       j=j+1
  until end of packet
   count =0
   for j=1 to end of temporary buffer T
         for i=1 to end of B[i][j]
               T=B[i][j]
               count= count+1
         until end of B[i][j]
         count[i]=count
         j=j+1
   until end of T
   E=Euclidean distance of first and last infected lines
   K=E/T
   Kernel =K+ constant
   Classify Malcode into different classes
       Apply d_{t,i}=∑_r count − 1 , p
       if(d_{t,i}>threshold_i)
       Then block the packet
       end if
until end of each packet
```

The section above briefly discussed the various steps involved in contribution one using the combined PMR method. The pictorial representation through flow diagram, steps involved and pseudo code of the proposed contribution one are also explained. The performance of the proposed PMR method is evaluated using the various parameters and the results obtained are explained in the section below.

4.6. Experimental Setup and Results

In experimentation, the downloaded programs in the system are scanned and analyzed using PCA comparing with the labeled dataset in SQL server. If the statement of the program matches with any of the signature stored in the labeled dataset, they are classified under various classes using MSVM. The count limit is set to count the number of Malcode lines classified and if that count limits exceeds the threshold limit then they are blocked from further infection. The proposed method is implemented using Java.

For validation of the proposed method, the real programs are downloaded from the Internet. The training dataset consists of 1008 unknown signatures. Those unknown signatures are classified into four labels namely operating system, hard disk, software and web browser. A sample of some unknown signatures used as labelled dataset for experimentation is shown in Annexure I.

The dataset used for validation contains 300 programs and in that 130 programs are benign programs and 170 programs contain Malcode lines. The training dataset is the trained unknown signatures and the test dataset is the programming language programs. The proposed PMR method is implemented. The experimentation methodology is shown in figure.4.3.

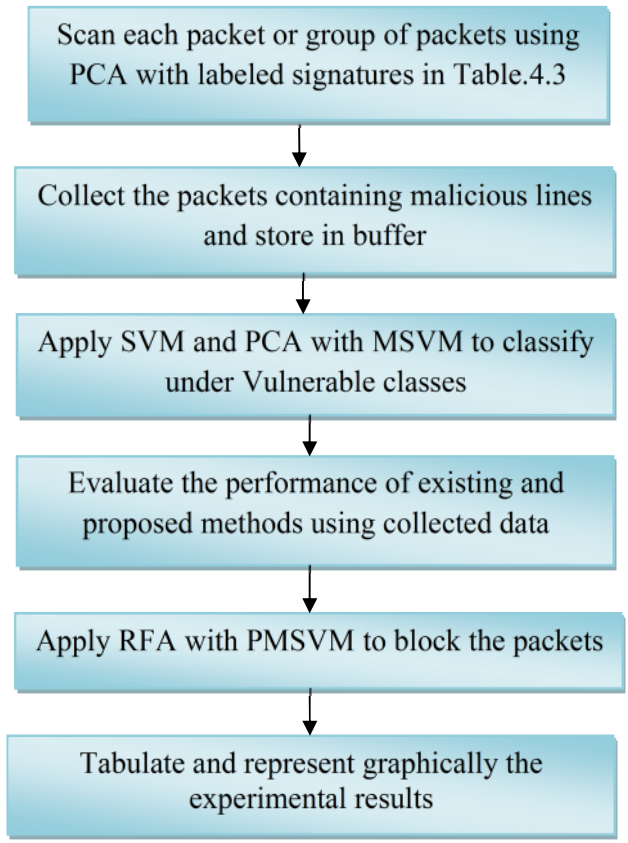

Figure.4.3. Experimentation Methodology for Contribution One

To evaluate the performance of the proposed method, the following five parameters listed below are used and defined:

(i) Memory Utilization
(ii) Time Consumption
(iii) Precision Value
(iv) Recall Value
(v) Accuracy

(i) Memory Utilization

The usage of CPU of the system is calculated for finding memory utilization.

Memory Utilization = CPU memory consumption after process completion –

CPU memory consumption before process beginning

(ii) Time Consumption

The time consumption is the period between the end of the detection system and the start of process execution.

Time Utilization = Finishing time of processing – Starting time of processing

(iii) Precision value

Precision refers to the retrieved document. This is calculated by the total number of relevant documents divided by the total number of resultant documents.

$$Precision\,Value = \frac{True\ Positive}{True\ Positive + False\ Positive}$$

(iv) Recall value

Recall value is referred to as the relevant documents that are related to the request search.

$$Recall\,Value = \frac{True\ Positive}{False\ Positive + False\ Negative}$$

(v) Accuracy

Accuracy provides the required related documents used for classification.

$$Accuracy = \frac{True\ Positive + True\ Negative}{True\ Positive +\ False\ Positive + True\ Negative + False\ Negative}$$

The parameters used for evaluating the blocking performance using the Rabin Footprint Algorithm are listed below:

(a) Detection Rate
(b) Containment Rate

(a) Detection Rate

Detection rate is defined as percentage of number of malicious codes correctly identified by total number of malcodes.

$$Detection\ Rate = \frac{Number\ of\ malcodes\ correctly\ identified}{Total\ number\ of\ malcodes\ detected} \times 100$$

(b) Containment Rate

Containment rate is defined as the percentage of number of malcodes blocked from number of malcodes correctly identified.

$$Blocking\ Rate = \frac{Number\ of\ malcodes\ blocked}{Number\ of\ malcodes\ correctly\ identified} \times 100$$

The proposed PMSVM is compared with the existing SVM [36]. The input programs are experimented using two different ways. First, the single programs are experimented and then the groups of programs containing five to ten programs stored in folders are experimented. The threshold is set to 1 and 2 respectively for the two types. The experiments are repeated for both the single programs and groups of programs varying the threshold values. It is further noted that irrespective of the types of programs executed, the average values remain the same. The average of the values obtained are tabulated and shown in table.4.4 and figures 4.4 to 4.8.

Table.4.4. Performance Comparison of Detection Results for Existing and Proposed PMSVM Method

Parameters	Existing SVM (Nir Nissim et al.)	Proposed PMSVM	% of Improvement
Memory Utilization (mb)	30	**26**	**13.33**
Time Consumption (ms)	11	**9**	**18.18**
Precision Value (%)	36.72	**47.45**	**22.61**
Recall Value (%)	52.57	**60.87**	**13.63**
Accuracy (%)	48.72	**56.37**	**13.57**

From the above table.4.4, it is observed that the proposed ***P***rincipal component analysis with ***M***ulticlass ***S***upport ***V***ector ***M***achine (***PMSVM***) provides better performance compared to that of the existing Support Vector Machine method.

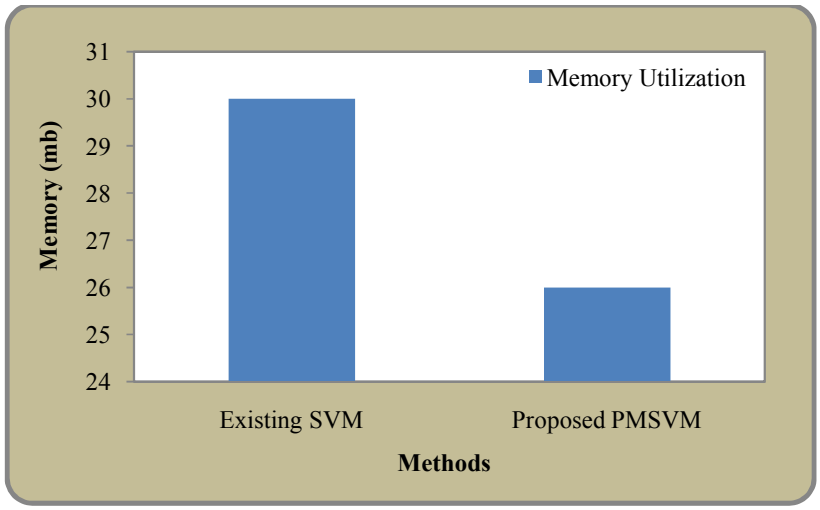

Figure.4.4. Comparison of Memory Utilization for Contribution One

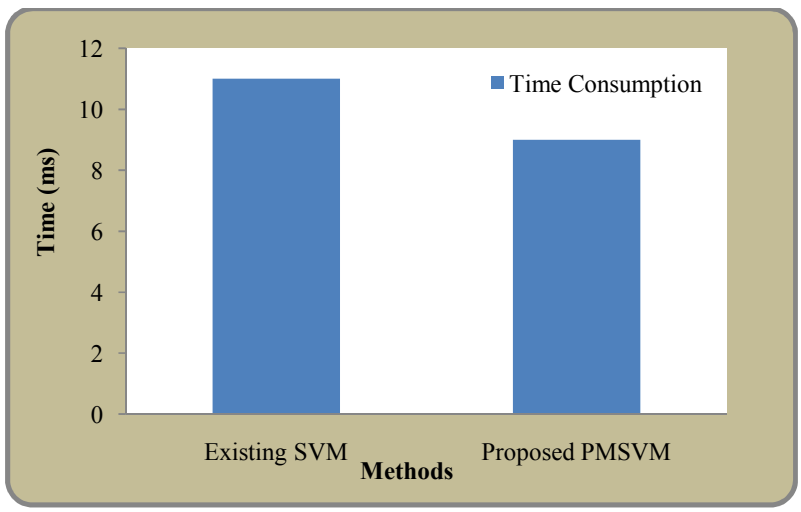

Figure.4.5. Comparison of Time Consumption for Contribution One

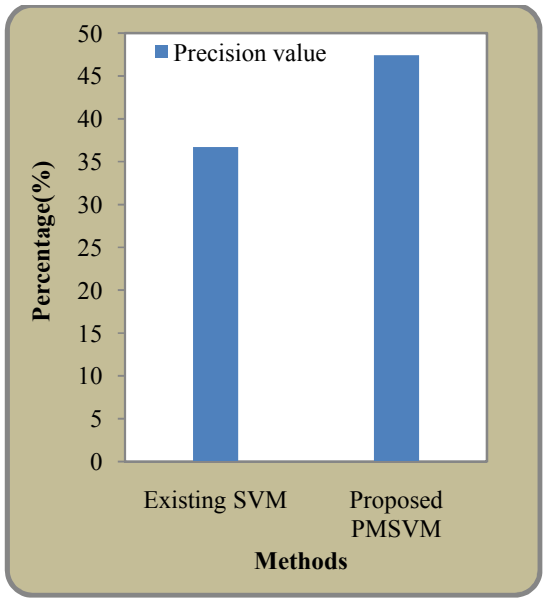

Figure.4.6. Comparison of results for Precision Value for Contribution One

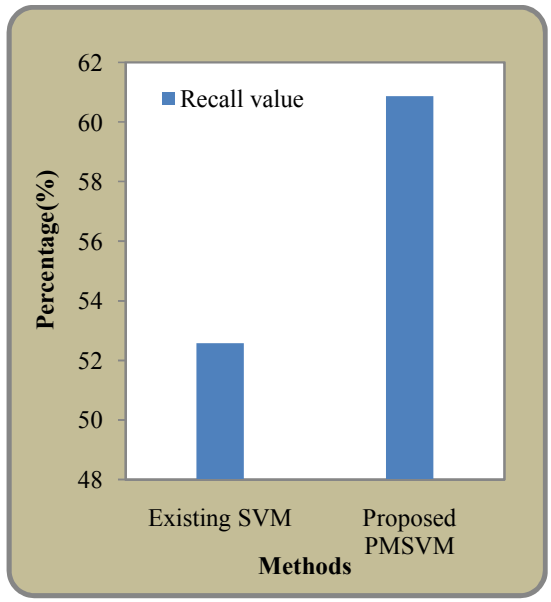

Figure.4.7. Comparison of results for Recall Value for Contribution One

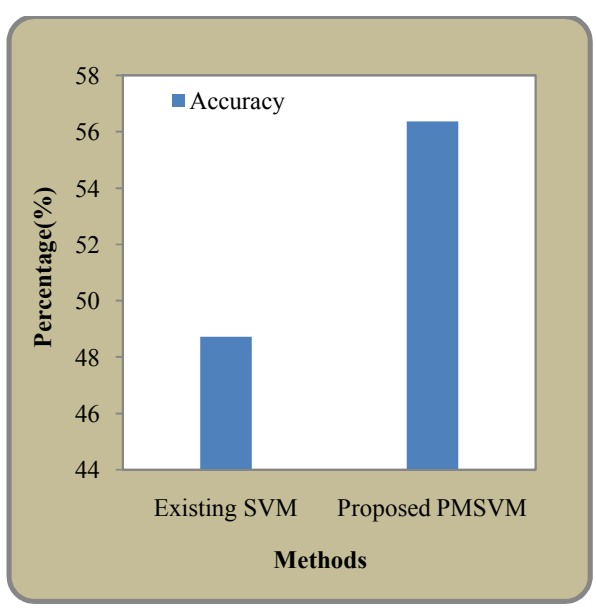

Figure.4.8. Comparison of results for Accuracy for Contribution One

It is observed from the figures.4.4 to 4.8, that the proposed method on the average has achieved minimum memory utilization with 26mb and time consumption of 9ms. The accuracy attained by the proposed method is also improved by 13.57%.

The proposed Containment technique is evaluated using Detection Rate and Containment Rate and the result is shown in figure.4.9. It is very important to note that when the detection rate is 66.66%, the containment rate is 66.66%. The detected Malcode files are blocked using the proposed PMR method. This shows that the containment rate is 100%, that is, all the detected malcodes are blocked.

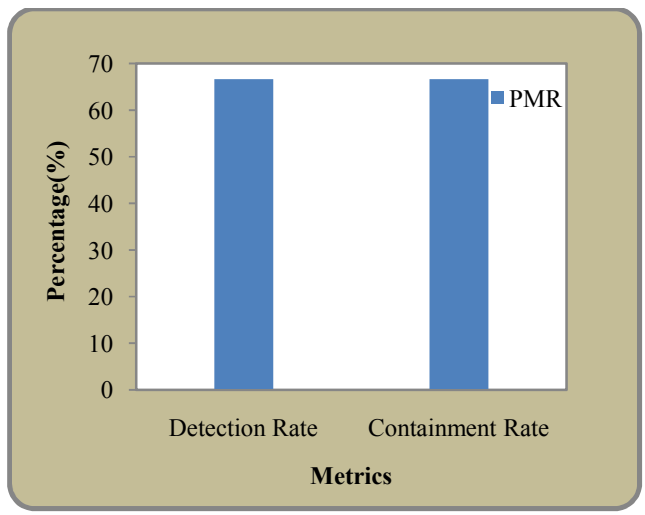

Figure.4.9. Results for Containment due to Contribution One

The Malcode programs detected are completely blocked using the ***PMR*** method over the time period of 200 ms.

4.7. Chapter Summary

In this chapter, self-carried propagation scheme and monomorphic payload format characteristics worms are detected and classified. The classified payloads are blocked using containment scheme to prevent the network from payload entry. The blocking is performed to overcome the challenges of payload worms that cause unnecessary traffic in the network. The proposed PMSVM provides better detection of payload programs and PMR provides better blocking of detected payloads. The experimental results show on the average, a detection accuracy of 56.37% during detection and containment rate of 100% over 200 ms of time. In other words all the detected Malcodes are blocked in containment step. Though unknown Malcodes are blocked, self-carried worms propagate and infect the network through packet payload information exploiting the vulnerable applications. To detect those packet content anomalies, Contribution two is proposed and explained in chapter 5.