Indian Journal of Science and Technology, Vol 9(41), DOI: 10.17485/ijst/2016/v9i41/86922, November 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Malicious Traffic Detection and Containment based on Connection Attempt Failures using Kernelized ELM with Automated Worm Containment Algorithm

S. Divya[1]* and G. Padmavathi[2]

[1]Faculty of Computer Science and Multimedia, Lincoln University College, Selangor; divya.phd.research@gmail com
[2]Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, University, Coimbatore - 641043, Tamil Nadu, India; ganapathi.padmavathi@gmail.com

Abstract

Objectives: In the world of Internet today, most of the communications are done through Internet applications. Rapidly with the growth of Internet, the security threat on Internet is also increasing. Internet worms are one of the serious dangerous threats heavy financial losses. To overcome these damages, the proposed methodology provide better defense mechanism through Internet worm detection and containment schemes based on connection attempt failures characteristic. **Method:** The Internet worm detection is done using the Machine Learning Method based on Anomaly detection schemes and containment based on blocking schemes. The proposed **k**ernelized **E**xtreme Learning Machine with **A**utomated Worm Containment Algorithm *(*kEA*)* method is used for detection and containment of malicious traffic from non-existing IP addresses based on connection attempt failures. **Findings:** Second channel based propagation through botnet worms propagates illegal traffic from malicious IP addresses through connection attempt failures. This traffic is transferred through TCP and UDP transmission schemes. The proposed work is used to identify the second channel propagating worms and containment of malicious traffic. **Improvement:** The proposed **k**ernelized **E**xtreme **L**earning **M**achine (kELM) method achieved detection accuracy improved by 23.67%. Then proposed kEA method blocks all the detected malicious IP addresses with 100% containment at the time span of 33 ms.

Keywords: Connection Attempt Failures, Kernelized ELM, Malicious IP

1. Introduction

The huge development of Internet, the data transaction process through network is growing highly in communication fields. Along with these rapid developments of Internet applications, the spread of attacks are also increasing. Among the various attacks, worm attack is one of the serious security threats that affect the data[1–3]. Worms are hazardous because their fast propagation. If worm contaminates network, it will automatically start to propagate which tends to affect network fully because of network congestion[4].

The worm itself acts as a computer programs by transmitting copies from one node to another throughout a network. Without some user involvement this transmission occurs, so permitting them to extend immediately and efficiently[5–7]. The worm that propagates in the network infects the vulnerable hosts exploiting the operating system and program[8]. Now-a-days, worms are causing dangerous security threats in the network through blocking packets, creating delays and loss of data during transfer[9].

The damages caused by Internet worms during the past 20 years created serious security threats and large

*Author for correspondence

Table 1. Potential damages caused by internet worms

Year	Worm Names	Infection
1998	Morris	Infected - 60,000 computers Financial Loss - $100 million
2001	Code Red	Infected - 3,50,000 systems Financial Loss - $2.5 billion
2003	Slammer	Infected: 75,000 Computers Damaged: MS SQL servers
2004	Witty	Infected: 12,000 hosts in 45 minutes
2007	Storm	Infected : Tens of Millions of hosts
2008	Conficker	Infected: 90% of susceptible hosts within minutes and controlled 6.4 million hosts
2012	Stuxnet	Created cyber war.

financial losses in the world[10–15]. Some of the infections and damages caused by Internet worms are listed in Table 1 below.

To avoid expensive damages caused by Internet worms, worm defense schemes are very important for the safety and security of the systems connected to the network.

The combined algorithm namely, **k**ernelized **E**xtreme Learning Machine with **A**utomated Worm Containment Algorithm (**kEA**) is used to detect the second channel propagating worms and containment of malicious traffic. Second channel based propagation through botnet worms propagates illegal traffic from malicious IP addresses through connection attempt failures. The illegal traffic is transmitted through TCP and UDP transmission schemes. The payload traffic transfers from Internet through transmission schemes without any changes in behavior are called monomorphic worms.

The proposed method checks each trace of incoming traffic and removes non-failures type of connections. With help of failure traces and feature vectors are derived. In the feature vectors for instance assured kind of failures, ratio of different destination port numbers, regular interval between failures, total amount of failures, and addresses for internet protocol and the average number of failures per destination IP address are trained using kernelized Extreme Learning Machine algorithm to distinguish normal and malicious connections in the training step.

Further the usual, Peer to Peer and Internet worm indications in detection step are classified using kernalized Extreme Learning Machine algorithm. The counter is set to count the output weight of malicious traces from the detected step. The counter checks whether it reach or exceed its threshold limit before the end of each cycle. Then block those malicious IP addresses. The experimentation is done with the different traffic traces collected from web sources. The results obtained for detection is compared with the existing C 4.5 method and 96.60% of detection accuracy achieved during detection and containment rate of 100% over 33 ms of time are presented.

Internet worm detection based on connection failures are detected by various proposed approaches and some of them detected through Machine Learning. Machine learning approaches detect the Internet worms quickly. Some of the existing approaches are discussed below:

In[16] machine learning classifiers are applied on static features for detect malicious type of code. The classifiers find out patterns in the binary code files that classify new unknown files are applied. SVM, random forest, logistic regression ANN, naive bayes, decision trees, BDT and BNB are the classification algorithms used for machine learning.

In[17] proposed an artificial neural network (ANN) to identify the occurrence of computer worms derived from amounts of computer activities. Trained the known worms. After training, the binary patterns of hidden neuron outputs are extracted. With the extracted output, behavior of new pattern is designed to detect new worms. The average accuracy of detecting the unknown worm is 90%.

In[18] proposed a worm detection scheme is based on machine learning techniques. The author mainly focuses accurately detecting unknown worm behavior in host feasibility that reducing the needed set of features accumulated from the monitored computer. The accuracy of mean detection is 90% and for the particular unknown worms the accuracy achieved above 99%, when managing a lower stage of false positive rate at 0.005.

In[19] proposed a Knowledge-based Temporal Abstraction for identifying previously escaped known malicious instance classes based on their temporal behavior. The periodically watching time stamped security data activities inside the target host after that performing processing. Temporal abstractions are observed to detect suspicious temporal patterns are automatically generated. These types of patterns are well matched with a group of predefined set of malware with a security specialist who employing value constraints and set of time. Scan rate is set as 5 probes or else 10 probes and the vulnerable size of population is 15% or 37%.

In[20] better solution is provided to detect both osts within a monitored local network using c4.5 algorithm. Both as a differentiable failure pattern using the botnet allocated design and implementation. Consequently, the host is to be removed or used by a single trained network depend on number of failure quantity produced in the single host in a short time. This method detects both osts with accuracy is more than 99% and 0.5% of false positive rate.

In[21] invaders scan blindly the network does not have any knowledge about the active host in the target network and produce a high ratio of connection failure in the form of Internet Common Message Protocol type 3 code1 and TCP-RST packets. The approach is used for TCP randomly and sequential scanning finding depends on link failure messages.

In[22] proposed a Botnet design that employs the HTTP based Central architecture model to devise a new Botnet C&C known as CBC2 which aware us about how cloud computing can be misused. Deploy CBC2 on a cloud computing platform by utilizing onion routing mechanism for message passing that offers a difficult and hardly-readable communication graph to stay main C&C center from defender.

The above section discussed various existing anomaly based approaches proposed by different authors. Moreover, the various Machine Learning approaches proposed by authors for Internet worm detection are also discussed in detail. The detection approaches with containment techniques provide effective defense mechanism for secure network. So, the study is made for the containment approaches also.

The containment approaches are used to block the detected Internet worms and to secure the network from further infection of detected attacks. Some of the proposed methods for containment of Internet worms are discussed below.

In[23] proposed a Cooperative Internet worm containment and gossip based aggregation using decentralized information sharing. It consists of epidemic algorithms to distribute the information about infection and to obtain the quasi global knowledge on attack behaviors. The model characterizes the associations between about the attack detection accuracy and the level of knowledge in the distributed system. The worm propagation authority is suitable, even for a gossip interval of 10 minutes.

In[24] proposed an Optimal Source-Based filtering to filter out the worm. It minimizes the collateral damage considerably that is 50%., even as communication overhead incremented only linearly with the whole number of filters presented.

In[25] proposed a malicious packets blocking algorithm to block the malicious packets. An anomaly detection system is proposed depends on the Filter ary Sketch, in that traffics are recoded online and the anomalies are detected, depends on one class SVM. While an anomaly is detected, malicious buckets known along with the KL distance between the forecasting sketch and the observed sketch. At last, packet blocking algorithm using malicious packets are blocked. The computational complexity of blocking is $O(M+1)$. It is high speed networks range but error occurs from some of the ordinary packets as it applies feature value known as malicious worms.

In[14] proposed a Cloud based benign Re-WAW model that is depends on the two factor model and the law of worm propagation. For cloud based benign worms it consists of two revised Worm Anti Worm (WAW) such as cloud based benign Re-WAW model and two stage Re-WAW propagation model. The cloud based benign Re-WAW model is to achieve effective worm containment. The two-stage Re-WAW model follows active and inactive switching protecting techniques depending on the benign worms to malicious worms ratio. This model is intended to protect the network from congestion and other risks basis by the actives can of benign worms. It slows down containment trend after the switching time of 31.8 seconds but does little effect on the overall containment.

From the above study, it observed that there are few limitations found in the existing Machine Learning detection approaches and containment techniques. When there are more failures found in the network from Internet, the detection accuracy in detecting large number of malicious IP addresses consume more time. To overcome the above observed limitations, the paper has proposed a three-step methodology.

2. Proposed Methodology

In this proposed methodology, Monomorphic characteristic worms based on the failures in connection attempts with better accuracy is detected and blocked. Containment of these illegal traffic blocks the malicious IP addresses. Figure 1 shows the three different steps namely,

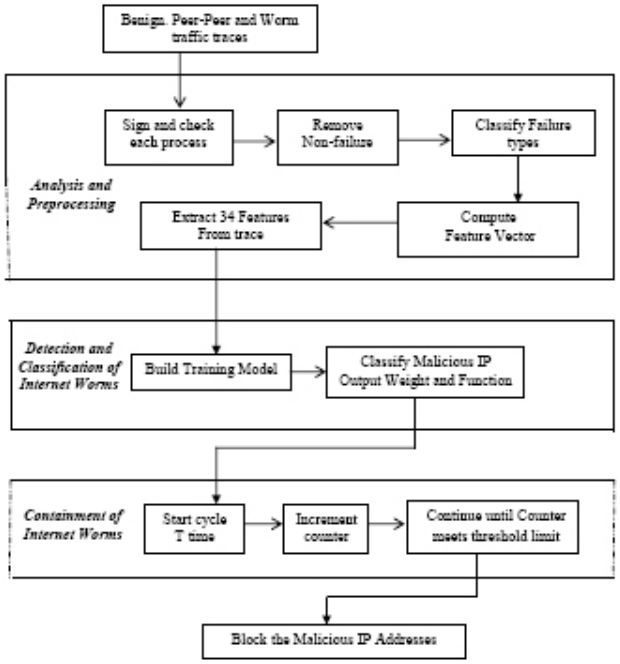

Figure 1. Block diagram of the proposed kEA method.

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

To detect the illegal traffic and block its infection to the network, a combined method called kEA method is proposed. ***kEA*** is a combination of ***k***ernelized ***E***xtreme Learning Machine with ***A***utomated Worm Containment Algorithm.

2.1 Analysis and Preprocessing

To analyze and identify the existence of malicious traffic created by connection failures, non-failure traces are sorted out. The feature vectors are computed based on the failures and the features are extracted.

2.1.1 Filter out Non-Failure Traces

In the proposed method, the failure traces are analyzed. The successful traces (i.e., non failure) are sorted out and therefore these traces are not utilized in this work. These traces are removed that reduce the storage, system loads, computation time and network flows quantity that the system needs to do. The Failures are classified into two different forms namely, immediate failures (***type-i***) and Time-out failures (***type-t***). The immediate failures are those found immediately when a trace is without address, whereas the Timed-out failures trace would have address without signal and are detected after a span of time.

2.1.1.1 TCP Failures

The TCP flow follows a three-way handshaking protocol to exchange any data. Hence identifying ***type-i*** failure is very easy. So to analyze ***type-t*** failure traces, it is necessary to verify the response to TCP SYN packets only. In the flow, if a TCP SYN packet is transmits without its TCP SYN ACK packet, the entering TCP RST packet in a particular span detects the ***type-t*** failure.

2.1.1.2 UDP Failures

The UDP is a bidirectional process. If the detected traffic is one way, within the given span of time, then it is assumed as ***type-t*** UDP failure. As the logical assumption, the sender should make sure that the receiver gets received data. Therefore assumption for every UDP flow "connection state" is maintained to keep track of failed and successful UDP requests. These requests are stored into the Pending List (PL) and the Connected List (CL). If the direction of UDP packet is determined in an outbound UDP packet with five-tuples, it will be added to PL with a timestamp of 120 seconds. Five tuples invoke the transport layer protocol, source IP, destination IP, source port and Destination port. Until the system receives a match edin-bound UDP packet, it will place the requests in PL. The five-tuples are changed from pending list to connected list once the system receives a matched tuples with two packets. It indicates that the UDP request is failed.

When data are transferred using these connections, the timestamps of five-tuples stored in the CL will be changed, connection will be active and it will prevent those five-tuples in the CL. If the timestamp of tuples stored in the pending list or the connected list is not changed for particular period of time, then it's considered as ended and terminated from the lists. If it is terminated from PL, then it is a ***type-t*** failure. If it is terminated from CL, then normal UDP flow is terminated. The failure types are given in Table 4.

2.1.2 Feature Vector Computation

The failures identified from analysis and preprocessing step are referred as features in this work. From the identi-

Table 2. Existing anomaly detection methods based on connection failures

Year	Authors	Methods used	Parametersused	Observations
2010	Asaf Shabtai et al	Knowledge-based Temporal Abstraction	Scanning Rate	The scan rate was set at 5 probes or 10probes, and the vulner able population size at 15% or 37%.
				Higher accuracy with almost 100% detection rates and very low false alarm rates.
2011	Syed Ali Khayam et al	Joint Network-Host Based Anomaly Detection Techniques	Detection Rate, False Alarm Rate	Detect bot hosts with more than 99% accuracy and false positive rate is lower than 0.5%
2014	Chun-Ying Huang et al	C 4.5 Algorithm	Accuracy	

Table 3. Existing containment approaches

Year	Authors	Methods	Parameters used	Observations
2010	Guangsen Zhang et al	Cooperative Internet worm containment	Time	Containment of worm propagation is acceptable even for a gossip interval of 10 minutes.
2012	Xufei Zheng etal	Cloud based benign Re-WAW model	Time	Slows down containment trend after the switching time of 31.8 seconds but does little effect on the over all containment.
2012	Liming Zheng et al	Malicious packets blocking algorithm	Time	The computational complexity of blocking is O(M+1). Scale to high-speed networks but error comes from that since some normal packets also uses the feature value identified as malicious.
2012	Fabio Soldo	Optimal Source-BasedFiltering	Containment Rate, Communication Overhead	This reduces the collateral damage significantly, i.e., by 50%., while the communication overhead increases only linearly with the overall number of filters available.

Table 4. Failure types

Protocol	Type	Description of the failure
TCP	i	TCP SYN sent, but got TCP resets (RSTs)
	i	TCP SYN sent, but got ICMP unreachable
	t	TCP SYN sent, but timed out
UDP	i	UDP sent, but got ICMP unreachable
	t	UDP sent, but timed out
DNS	i	A DNS server responds errors to a queried domain

fied failures in each host some features are extracted. In feature extraction[5], the terms snapshot and time window measurement (Δt) is explained first. A snapshot consists of a list of failures generated in network and is listed in Table 2. To decrease the snapshot length, failures identified between T – Δt are considered, where T is time. Feature vector are the failures collected from the snapshot. The features are grouped and features types are given below:

Failures types, Regular interval between failures, entire amount of failures, Ratio of specific destination port numbers, ratio of specific destination Internet

Protocol addresses and average amount of failures for each destination Internet Protocol address.

Labels are assigned to the collected feature vectors and are used to prepare and construct a classification model to distinguish the Benign, Peer to Peer and Internet traces. From the traces, features are extracted from the DNS, TCP and UDP failures. The features extracted are listed in Table 5.

2.2 Detection and Classification of Internet Worms

After the features are extracted and labeled under three different labels such as Normal, Peer-to-Peer and Internet Worm, the detected and classified traffic traces are stored under the defined labels. To detect and classify benign and malicious traffic traces with help of Kernelized extreme learning machine.

2.2.1 Kernelized Extreme Learning Machine

This learning algorithm is implements Single hidden layer feed forward network (SLFN), which select input weights

Table 5. Extracted features from a snapshot of failures

S.No.	Data type	Description
1	Real number	Average interval between two adjacent TCP failures
2	Real number	Average interval between two adjacent TCP RESETs
3	Real number	Average interval between two adjacent TCP unreachable failures
4	Real number	Average interval between two adjacent TCP timeout failures
5	Real number	Average interval between two adjacent UDP failures
6	Real number	Average interval between two adjacent UDP unreachable failures
7	Real number	Average interval between two adjacent UDP timeout failures
8	Real number	Average interval between two adjacent DNS failures
9	Boolean	Have TCP failures
10	Boolean	Have UDP failures
11	Boolean	Have DNS failures
12	Integer	Total number of TCP failures
13	Integer	Total number of TCP RESETs
14	Integer	Total number of TCP unreachable failures
15	Integer	Total number of TCP timeout failures
16	Integer	Total number of UDP failures
17	Integer	Total number of UDP unreachable failures
18	Integer	Total number of UDP timeout failures
19	Integer	Total number of DNS failures
20	Integer	Total number of all failures
21	Real number	Ratio of distinct destination ports to all destination ports
22	Real number	Ratio of distinct destination TCP ports to all destination TCP ports
23	Real number	Ratio of distinct destination UDP ports to all destination UDP ports
24	Real number	Ratio of distinct destination IP addresses to all destination IP addresses (count all network flows)
25	Real number	Ratio of distinct destination IP addresses to all destination IP addresses (count only TCP network flows)
26	Real number	Ratio of distinct destination IP addresses to all destination IP addresses (count only UDP network flows)
27	Real number	Average number of failures per destination IP addresses
28	Real number	Average number of TCP failures per destination IP addresses
29	Real number	Average number of TCP RESETs per destination IP addresses
30	Real number	Average number of TCP unreachable failures per destination IP addresses
31	Real number	Average number of TCP timeout failures per destination IP addresses
32	Real number	Average number of UDP failures per destination IP addresses
33	Real number	Average number of UDP timeout failures per destination IP addresses
34	Real number	Average number of DNS failures per destination IP addresses

(w_i), hidden nodes (Ñ), hidden layer biases (x_i) randomly output weights is produced. The SLFN uses $\tilde{N}$ hidden neurons and activation function f (a), Amount of training set (N) and X [$x_1, x_2, \ldots\ldots x_n$] as input features set.

Inputs: Training set and testing set as $\mathbf{N} = \{(\mathbf{a_i}, \mathbf{b_j}), \mathbf{a_i} \varepsilon\ \mathbf{R^n}, \mathbf{b_i} \varepsilon\ \mathbf{R^m}, \mathbf{i = 1, 2, 3 \ldots\ N}\}$

f(a) - kernel function, $\tilde{\mathbf{N}}$ – hidden neuron, ($\mathbf{w_i}$) - input weights, ($\mathbf{x_i}$) - biases

Outputs: Internet worm

Step 1: Define the hidden layer node $\tilde{\mathbf{N}}$, randomly allocate input weights $\mathbf{w_i}$, & $x_{i,}$ (i= 1, 2,…$\mathbf{N}$) is hidden Layer biases.

Step 2: Compute the output matrix of hidden layer h(a).

Step3: Sigmoid function using computes hidden layer and output weight $\tilde{\mathbf{N}}$

$$\mathbf{f(a)} = \frac{\mathbf{1}}{\mathbf{1 + e^{-a}}} \quad (1)$$

Step4: To rectify linear equation $\mathbf{H\beta = T}$, the output weight β is compute.

$$\beta = \mathbf{H^{+}T} \quad (2)$$

If h(a) hidden layer feature mapping is unknown to users, kernel function for ELM is illustrate users. KELM is specified by,

$$\text{KELM}\ (\mathbf{a_i}, \mathbf{a_j}) = 1/\text{H}\ \text{f}\ (\mathbf{a_i},).\ \text{f}\ (\mathbf{a_j}) \quad (3)$$

After classifying the malicious traffic created by connection failures, the classified failure connections made through anomalous IP address is blocked using automated worm containment steps and they are explained below.

2.3 Containment of Internet Worms

The containment algorithm is based on Automated Host-related Internet worm containment approach. Control the unwanted scans to the malicious IP address with use of containment. The algorithm applied is shown in Table 6.

Table 6. Containment algorithm

Step 1:	Construct the containment cycle and set counter to zero at the starting of each cycle T.
Step 2:	When a new IP address is scanned, then the counter is incremented.
Step 3:	If the counter value reaches its threshold limit before the end of cycle T, then that particular IP address is blocked.
Step 4:	Else scan continues until end of IP address and reset the counter to zero.

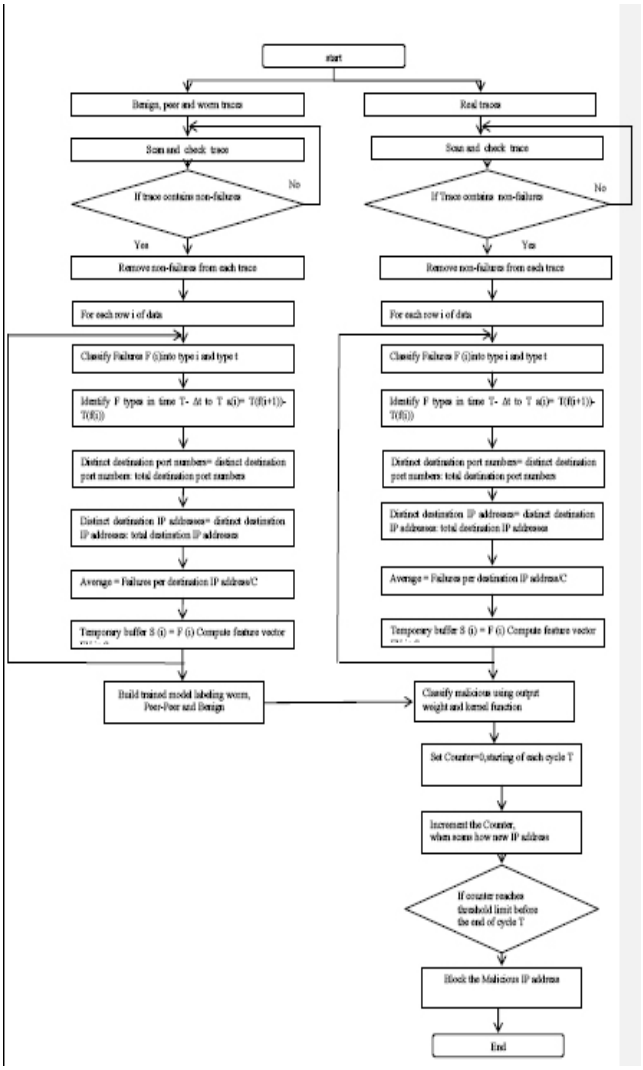

Figure 2. Flow diagram of the proposed kEA method.

The flow diagram contains two flows parallel, the left side flow shows the training steps and the right side shows the detection steps. The left side flow stops after it built the training model. The training model is used as training dataset for classifying there all traffic traces. The proposed kEA method flow diagram is shown in Figure 2 below.

The various steps and the methods used for detection and containment of second channel worms propagating through botnet propagation using the transmission schemes such as TCP and UDP schemes and monomorphic payload format worms are discussed in detail above.

3. Experimental Result

For experimentation, three different traffic traces are scanned and is stored in SQL server. The non-failures are removed. The feature vectors are computed and thirty four features are extracted. Build the trained model, classify and detect the normal, peer-to-peer and Internet Worms traffic traces using KELM. Set the counter at the starting cycle of time T and set its threshold limit to 2. If the counter reach or exceed threshold limits before the end of cycle time T, block the IP addresses.

The three traces such as Normal (Benign), Peer-to-Peer and Internet Worm traffic traces datasets are used for experimentation. The Real traces are gathered from https://traces.simpleweb.org/traces/netflow/netflow1. Benign, Internet worm and Peer-to-Peer traces are collected from web sources such as http://www.caida.org/data/passive http://wisnet.seecs.nust.edu.pk/downloads.php and http://www.fukuda-lab.org/mawilab/v1.0/2003/04/11/20030411.html. The data set contains the traces of peer-to-peer failures, Sasser and Slammer worm traces. The JAVA programming using implemented proposed kEA method. The Figure 3 shows the experimentation methodology.

The proposed kernelized Extreme Learning Machine (kELM) is evaluated with the existing C 4.5 method. The programs are experimented using the collected dataset. All the three datasets extracted through Wire shark and is evaluated. The experiments are performed and compared with existing work. The results obtained are tabulated

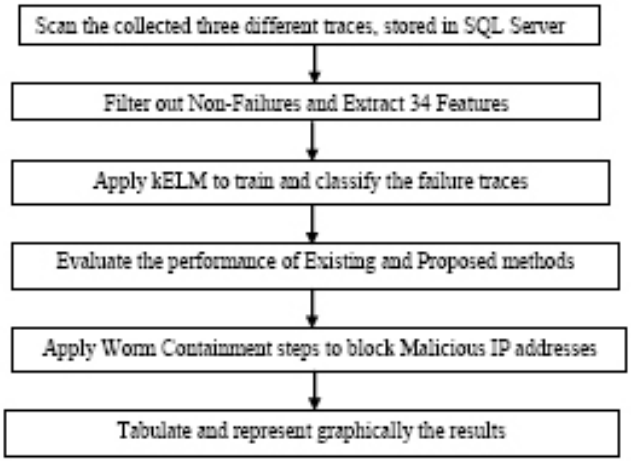

Figure 3. Experimentation methodology.

Table 7. Performance comparison of detection results for existing and proposed methods.

Parameters	Existing C 4.5	Proposed kELM	% of Improvement
Memory Utilization (mb)	161	**100**	**37.88**
Time Consumption (ms)	123	**44**	**64.22**
Precision Value (%)	96.69	**98.49**	**1.82**
Recall Value (%)	77.00	**94.91**	**18.87**
Accuracy (%)	73.73	**96.60**	**23.67**

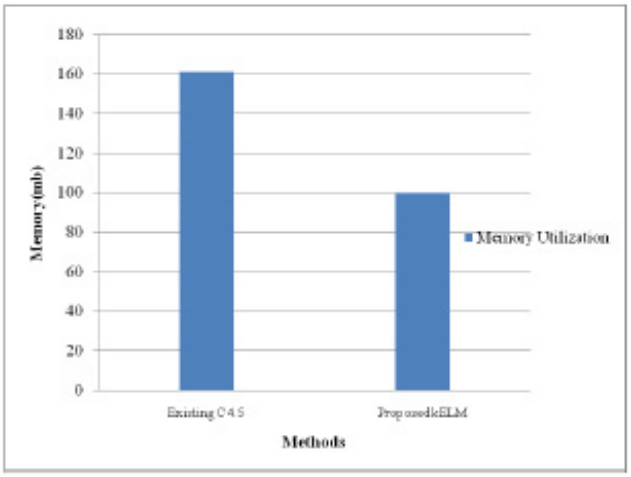

Figure 4. Comparison of memory utilization.

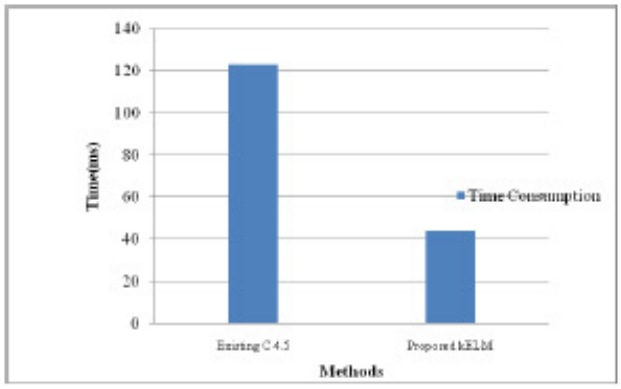

Figure 5. Comparison of time consumption.

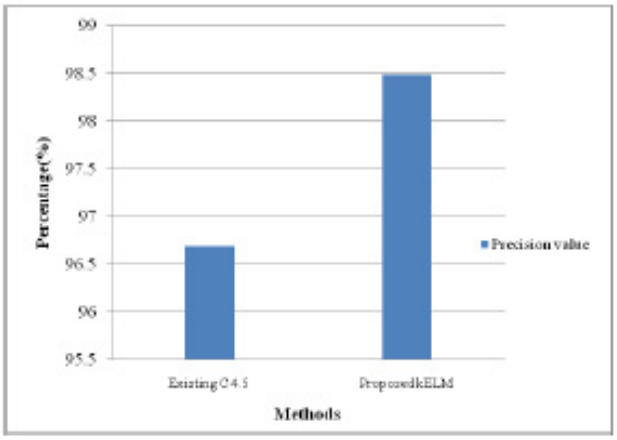

Figure 6. Comparison of results for precision value.

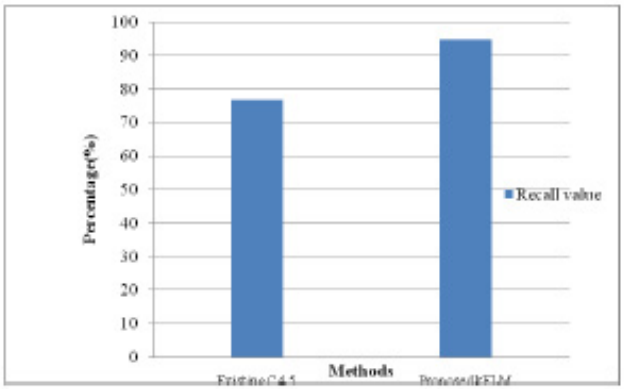

Figure 7. Comparison of results for recall value.

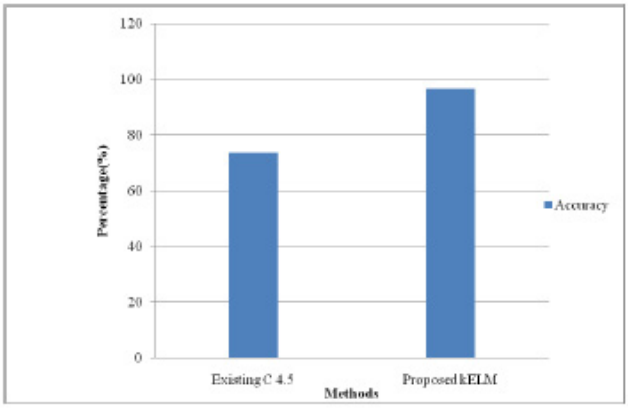

Figure 8. Comparison of results for accuracy.

which is represented in Table 7 and shown in the Figures 4 to 8.

Proposed **k**ernelized **E**xtreme **L**earning **M**achine (**kELM**) method for detection is evaluated based on the parameters like as time consumption, memory utilization, recall value, precision value, and accuracy.

From the above Table 7, it is observed that the proposed **k**ernelized **E**xtreme **L**earning **M**achine method provide better performance compared to that of the C 4.5 algorithm.

It is observed from the Figures 4 to 8, that the proposed CPC method on the average has achieved minimum memory utilization with 100 mb and time consumption of 44 ms. The accuracy attained by the proposed method is also improved by 23.67%.

The proposed Containment technique is evaluated applying containment rate and detection rate and result is shown in Figure 9. When the detection rate is 99.05% and containment rate is 99.05% very essentially notified that rates. The detected malicious IP addresses are blocked using the proposed kEA method. This shows that the containment rate is 100%, that is, all the detected IP addresses are blocked.

The Malicious IP addresses detected are completely blocked using the ***kEA*** Method over the time period of 33 ms.

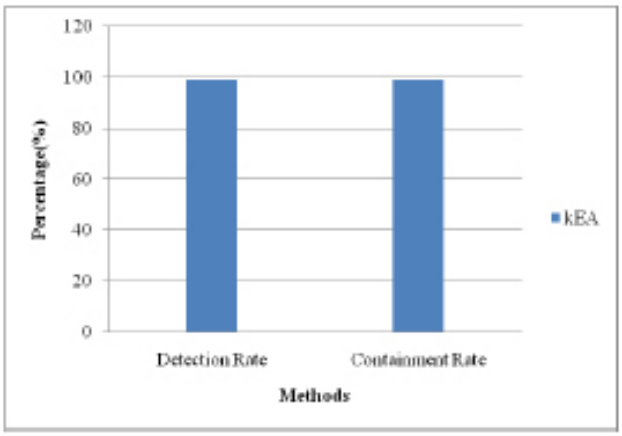

Figure 9. Results for containment.

4. Conclusion

Second channel botnet propagating worms creating malicious traffic due to connection failures are detected and blocked in this paper. The transmission media is analyzed and the failures are traced. The features are extracted from the failure traces and labeled namely, Benign, Peer-to-Peer and Internet Worm. The detection is performed using kELM and the real traffic traces are classified under

labels. Automated containment steps are applied and the detected malicious IP addresses are blocked. The proposed contribution provides better detection of malicious traffic caused by connection failures and better containment of detected malicious traffic. The experimental results show on the average, a detection accuracy of 96.60% during detection and containment rate of 100% over 33 ms of time. In other words all the detected malicious IP addresses are blocked in containment step. The detection accuracy is improved by 23.67% compared to the existing C 4.5 method.

5. References

1. Pratama A, Rafrastara FA. Computer worm classification. International Journal of Computer Science and Information Security. 2012 Apr 1; 10(4):21.
2. Yu W, Wang X, Champion A, Xuan D, Lee D. On detecting active worms with varying scan rate. Computer Communications. 2011 Jul 15; 34(11):1269–82.
3. Choi YH, Li L, Liu P, Kesidis G. Worm virulence estimation for the containment of local worm outbreak. Computers and Security. 2010 Feb 28; 29(1):104–23.
4. Anbar M, Abdullah R, Manasrah A, Munther A, Manickam S. BADUW: Behavioural based approach for detecting UDP worm. Indian Journal of Science and Technology. 2015 Dec 11; 8(35).
5. Zheng X, Li T, Fang Y. Strategy of fast and light-load cloud-based proactive benign worm countermeasure technology to contain worm propagation. The Journal of Supercomputing. 2012 Dec 1; 62(3):1451–79.
6. Khule M, Singh M, Kulhare D. Enhanced worms detection by Netflow. International Journal of Engineering and Computer Science. 2014 Mar; 3(3):5123–7.
7. Yang X, Shi Y, Zhu H. Detection and location algorithm against local-worm. Science in China Series F: Information Sciences. 2008 Dec 1; 51(12):1935–46.
8. Moskovitch R, Elovici Y, Rokach L. Detection of unknown computer worms based on behavioral classification of the host. Computational Statistics and Data Analysis. 2008 May 15; 52(9):4544–66.
9. Rasheed MM, Norwawi NM, Ghazali O, Kadhum MM. Intelligent failure connection algorithm for detecting internet worms. IJCSNS. 2009 May; 9(5):280.
10. Mohammed A, Nor SM, Marsono MN. Analysis of internet malware propagation models and mitigation strategies. Analysis. 2012; 2(1).
11. Khouzani MH, Altman E, Sarkar S. Optimal quarantining of wireless malware through reception gain control. IEEE Transactions on Automatic Control. 2012 Jan; 57(1):49–61.
12. Chen S, Liu L, Wang X, Zhang X, Zhang Z. A host-based approach for unknown fast-spreading worm detection and containment. ACM Transactions on Autonomous and Adaptive Systems (TAAS). 2014 Jan 1; 8(4):21.
13. Yu W, Wang X, Calyam P, Xuan D, Zhao W. Modeling and detection of camouflaging worm. IEEE Transactions on Dependable and Secure Computing. 2011 May; 8(3):377–90.
14. Fan X, Xiang Y. Defending against the propagation of active worms. The Journal of Supercomputing. 2010 Feb 1; 51(2):167–200.
15. M. Zaki, Hamouda AA. Design of a multi agent system for worm spreading reduction. Springer, Journal of Intelligent Information System. 2010; 35:123–55.
16. Shabtai A, Moskovitch R, Elovici Y, Glezer C. Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey. Information Security Technical Report. 2009 Feb 28; 14(1):16–29.
17. Stopel D, Moskovitch R, Boger Z, Shahar Y, Elovici Y. Using artificial neural networks to detect unknown computer worms. Neural Computing and Applications. 2009 Oct 1; 18(7):663–74.
18. Moskovitch R, Elovici Y, Rokach L. Detection of unknown computer worms based on behavioral classification of the host. Computational Statistics and Data Analysis. 2008 May 15; 52(9):4544–66.
19. Shabtai A, Fledel Y, Elovici Y, Shahar Y. Using the KBTA method for inferring computer and network security alerts from time-stamped, raw system metrics. Journal in Computer Virology. 2010 Aug 1; 6(3):239–59.
20. Huang CY. Effective bot host detection based on network failure models. Computer Networks. 2013 Feb 4; 57(2):514–25.
21. Anbar M, Ramadass S, Manickam S, Al-Wardi A. Connection failure message-based approach for detecting sequential and random TCP scanning. Indian Journal of Science and Technology. 2014 May 28; 7(5):628–36.
22. Torkashvan M, Haghighi H. CBC2: A cloud-based botnet command and control. Indian Journal of Science and Technology. 2015 Sep 22; 8(22).
23. Zhang G, Parashar M. Cooperative detection and protection against network attacks using decentralized information sharing. Cluster Computing. 2010 Mar 1; 13(1):67–86.
24. Soldo F, Argyraki K, Markopoulou A. Optimal source-based filtering of malicious traffic. IEEE/ACM Transactions on Networking (TON). 2012 Apr 1; 20(2):381–95.
25. Zheng L, Zou P, Jia Y, Han W. Traffic anomaly detection and containment using filter-ary-sketch. Procedia Engineering. 2012 Dec 31; 29:4297–306.