Employing Machine Learning Algorithms to Detect Unknown Scanning and Email Worms

Shubair Abdulla[1], Sureswaran Ramadass[2], Altyeb Altaher[2], and Amer Al-Nassiri[3]
[1]Instructional and Learning Technologies Department, Sultan Qaboos University, Oman
[2]NAV6 Center of Excellence, Universiti Sains Malaysia, Malaysia
[3]Faculty of Computer Engineering and Computer Science, Ajman University of Science and Technology, UAE

Abstract: *We present a worm detection system that leverages the reliability of IP-Flow and the effectiveness of learning machines. Typically, a host infected by a scanning or an email worm initiates a significant amount of traffic that does not rely on DNS to translate names into numeric IP addresses. Based on this fact, we capture and classify NetFlow records to extract feature patterns for each PC on the network within a certain period of time. A feature pattern includes: No of DNS requests, no of DNS responses, no of DNS normals, and no of DNS anomalies. Two learning machines are used, K-Nearest Neighbors (KNN) and Naive Bayes (NB), for the purpose of classification. Solid statistical tests, the cross-validation and paired t-test, are conducted to compare the individual performance between the KNN and NB algorithms. We used the classification accuracy, false alarm rates, and training time as metrics of performance to conclude which algorithm is superior to another. The data set used in training and testing the algorithms is created by using 18 real-life worm variants along with a big amount of benign flows.*

Keywords: *IP-Flow, netflow, NB, KNN, scanning worms, email worms.*

Received September 27, 2011; accepted May 22, 2012; published online January 29, 2013

1. Introduction

Worms' inventors are continuously inventing malicious codes. They caused loss of millions of dollars to businesses around the world. During the past 10-15 years, there have been many instances of worm that were able to penetrate the defense systems in the Internet, such as the Code Red worm in 2001 [34], the Slammer worm in 2003 [15], the Sasser worm in 2004 [1], the Strom botnet in 2007 [30], the Conficker [21] in 2008, and the Stuxnet worm in 2010 [14]. A computer worm is a self-replicating program which spreads stealthily over the network nodes by exploiting the software vulnerabilities. After sneaking into the network, the worm attacker remotely controls the infected hosts to launch its malicious tasks, such as stealing sensitive information, sending spam emails, and generating Distributed Denial-of-Service (DDoS) attacks.

Since the damage of worms is steadily growing, worm detection research has become vital to the field of Network Security. The security community has adopted Network Intrusion Detection (NID) systems to defend worms which can be classified into two categories: content-based systems and behavior-based systems. Relying on a pre-compiled database of signatures, content-based systems explore the network traffic to catch ongoing attack's signature. Without discovering signatures, these systems cannot defend against attack. Therefore, content-based systems are not effective against unknown attacks. Considering the significant amount of human involvement and time consuming in discovering the worm's signature, many research efforts [6, 9, 18, 22] have been devoted to develop behavior-based systems which are more sophisticated. They directly analyze the network traffics, and consequently, detect the worms. However, reducing the false-positive and false-negative rates is still a challenging problem. Initially, the behavior-based systems detect network statistical normals and anomalies to measure a "baseline", an alarm system can trigger when there is a deviation from this "baseline".

To collect statistical information and build a "baseline", the behavior-based NID systems inspect the payloads of every network packet to find known or unknown attacks [23, 26]. This task is now hard or even impossible due to high-speed lines, large number of packets, and the huge volume of packet information makes it too difficult to analyze. One option that has been recently attracted the attention of the researchers is IP-Flow-based technique. The IP-Flow is unidirectional chains of IP packets of TCP/UDP protocol travelling between a pair of IP addresses within a certain period of time. The flow information can be exported by using an export mechanism such as CISCO NetFlow [3] and sFlow [20]. CISCO NetFlow has become an industry standard for network traffic monitoring and one of the most commonly used NetFlow versions is NetFlow 5. A NetFlow record contains wide variety of information that can be used for a variety of purposes including data warehousing, network monitoring, security analysis, etc. Although, the information carried by flows is limited to the network nodes interactions, the volume of flow records is extremely huge. However, the flow information can be used to build an effective NID system after being sampled and normalized.

Recently, there has been a significant amount of research that uses IP flow information to train learning machine for the purpose of combating worms. The applications of machine learning techniques involve identifying a number of features of worms. The features are attributes of worms calculated over multiple packets. Then these features are used to train a classifier to build a classification model. Most research efforts in this area have been directed towards the following aspects: Removing the redundancy and noise from the data collected; performing efficient training for the classifier by using real variants of worms; identifying the most optimum classifier among the data mining classification algorithms.

Our contributions to this research work are as follows: First we investigated whether using machine learning can detect variants of unknown worms. We extracted unique features from the NetFlow records for each host connected in a certain period of time. To avoid the redundancy and noise, the features were classified as: DNS request, DNS response, DNS normal, or DNS anomaly. (18) the most dangerous scanning and email worms have been used to train and test the classifiers. Second, we compared the individual performance between Naive Bayes (NB) and K-Nearest Neighbors (KNN) machine learning classifiers to show any of the classifiers performs better. Our work is inspired by the work of Mesud *et al.* [13] to detect email worms. However, our approach is more comprehensive, it tries to detect both email and scanning worms. In addition, we have extracted different features which are unique and present in both types of worms. To show which algorithm is superior to another, we used the cross-validation and the paired t-test to track the performance of NB and KNN by using the classification accuracy, false alarm rates, and the training and prediction times as performance metrics.

The reminder of the paper is organized as follows: Section 2 describes related work. Section 3 provides background information on scanning and email worms. Section 4 identifies briefly NB and KNN classifiers and details the feature set worm variants along with the system implementation. Section 5 explains the experimental approach. Results and discussion are presented in sections 6. Finally, section 7 concludes the paper and gives directions for future work.

2. Related Work

Although machine learning has already been used for detecting malicious attacks, the authors in [8] mentioned that there have been few attempts to use data mining techniques for the purpose of identifying unknown worms. However, the authors in [4] mentioned some research that has been done to compare the performance of machine learning methods in malware detection. A look at the literature reveals that the most of the machine learning-based research have been focusing on payload features to classify the malicious codes [10, 24, 25, 29]. The payload features that are extracted to train the classifiers in these approaches could be the variable length of instruction sequences, some strings, or the JUMP address. Several learning methods have been applied by the researchers such as: NB, Support Vector Machines (SVM), Instance-Based k (IBk), KNN, and Term Frequency-Inverted Document Frequency (TFIDF). It is obvious that these methods are not suitable for high speed networks as they require high processing to analyze the network packet payloads online. One of the most relevant to our work is [33] where the authors found that the worm actions grouped into 3 categories: Registry, file system, and network. They used SVM to classify malicious programs classification. However, our approach differs from their in that we consider the network activities to avoid installing the system on each network node to watch the Registry and file system.

Using IP addresses by worms obviates the need for a DNS query [31]. Based on this idea, some research are devoted to detect scanning worms by studying the DNS traffic. The method proposed by Whyte *et al.* [31], relies on the correlation of DNS queries with outgoing connections from an enterprise network to detect scanning worms. In 2008, Binsalleeh *et al.* [2] proposed a system that followed the same architecture in [31] but the new system performs online processing of TCP dump. In most of the above approaches, as in [2], the system inspects the network packet to find the DNS anomalies. This task is hard or even impossible due to high-speed lines, large number of packets, and the huge volume of packet information makes it too difficult to analyze. In fact, the main difference that distinguishes our work is our proposed system relies on IP-flow rather than network packets.

The email worms also attracted the attention of the researchers. The methods in [16, 17] are straight-forward with focus on the volume of DNS queries for Mail eXchange (MX) to detect email worm infections. Ishibashi *et al.* [7], proposes an approach for detecting worm based on prior knowledge of worm signature DNS queries. However, the problem of legitimate traffic that does not rely on DNS queries has not been resolved completely. This legitimate traffic could be originated by either normal users or network applications. The authors in [31] suggested whitelist to address those clients that legitimately do not rely on DNS. The disadvantage of using whitelist is that it needs to be updated regularly to reflect changes to the network.

3. Scanning Worms and Email Worms

A worm is a malicious program that self-propagates across the networks by exploiting the software vulnerabilities. According to the way that is followed in finding new host to infect, the worms are categorized into four groups: Scanning Worms, Email Worms, P2P Worms, and Instant Messaging Worms. To limit our scope, we will consider two types of worms, scanning worms and email worms, and since we concentrate on the IP flows, the discussion will be limited to the flows generated during the life-cycle of these two types. Readers who are interested in worms and their categories can refer to [12, 27].

- *Scanning Worms:* The life-cycle of scanning worm consists of four phases: victim finding, transferring, activation, and infection. The scanning worm is active over the network in victim finding and transferring phases, while its activities are limited to local hosts in the other phases. Most worms use either blind or hit-list scanning strategies. In the former strategy, the worm has no knowledge about the targets while in the later strategy the worm knows where the victims are. In both strategies, the worms scan a TCP or UDP port on the targets to find a host that runs vulnerable software and penetrate it. This scanning process causes a dramatic increase in anomaly traffic rate which makes it possible for a vigilant NID system to catch the worm [27]. Table 1 shows examples of scanning worms along with their scanning ports and the software vulnerabilities that they utilize.

Table 1. Examples of scanning worm.

Worm	Scanning Port	Software Vulnerability
CodeRed 2003	TCP 80	Buffer overflow in ms index server or MS IIS
Slammer 2003	UDP 1434	Buffer overflow in MS SQL server
Sasser 2004	TCP 445	Local security authority subsystem service LSASS
Witty 2004	Uses A UDP port to scan randomly generated list of UDP ports	Internet security systems (ISS) protocol analysis module (PAM)
Doomjuice 2005	TCP 3127	This worm spreads by entering systems through a backdoor created by the mydoom worm.

- *Email Worms:* This kind of worms spread by using email messages. They spread through HTML links or an attachment. If the HTML link or the attachment is opened, the worm will infect the computer and propagates by emailing itself using the user's address book. Email worms, such as Sobig, NetSky, and MyDoom, are programmed to drop backdoors, launch DoS, and send documents via email. Recently, most of the email worms' victims become part of Botnet: a group of computers (bots) infected by malicious programs that cause them to operate against the owners' intention and without their knowledge [11]. A bot becomes active over the network when it launches DoS or SPAM attacks, and when it tries to contact other bots searching for an update. In all cases, it generates a significant amount of anomalous traffic. Storm and Conficker worms are among the most recent severe examples of this kind of worms. When a host infected by Storm worm, it receives an initial list of 290 possible "peer nodes" from the botnet and attempts to contact each peer node to obtain more updated list of "peer nodes" [19]. The Storm's body contains a special function to turn the victim machine to a TCP/IP client to specify a TCP connection to each peer node. These connections evidently will generate significant amount of anomalous traffic in a few seconds.

Almost the same behavior found in conficker worm. Within its life-cycle, the conficker generates randomly a list of 250 domain names (rendezvous points), and then it attempts to contact these domains [21]. When the contacted domain is available, conficker will send a URL request to TCP port 80 of the target IP. The aim of this request is to download a malicious Windows executable. If the domain is not connected to the Internet, conficker will check for connection every 60 second. Based on our experiments, more than 1000 TCP port 80 requests could be generated by an infected host within 20 minutes. Such number of requests for same port in a short period indicates that the host behaves abnormally.

4. The Methodology

4.1. Using Machine Learning (ML) to Detect Worms

A learning machine is a computer program based on a specific mathematical concept designed to learn automatically from experience. The ML technique has been applied in a various applications including search engines, medical diagnosis, text and handwriting recognition, and so on. In 1994 ML was utilized for internet flow classification in the context of intrusion detection [32]. Most of the existing ML algorithms are fall into two categories: supervised and unsupervised learning algorithms. In the supervised learning the algorithm is provided with a collection of instances that are pre-classified into their desired classes. While the desired classes for all of the input instances are not known in the unsupervised learning. In this research, we investigate the performance of the KNN along with the performance of the NB. These two algorithms are among the most influential data mining algorithms in the research community [28].

We will briefly define the KNN and NB algorithms, readers interested in more information can refer to [28]. KNN algorithm is a straight-forward algorithm. It finds a group of K labels in the training set that are closest distance (nearest neighbors) to the unknown label. Then, the unknown label is classified by the majority vote in the KNN. The number of nearest neighbors, K, is considered as one of the most influential factors in the accuracy of the classification. For any given case, we need to set K to a large value in order to minimize the probability of misclassification and, at the same time, small value so that the K nearest labels are close enough to the right class. NB algorithm is based on the Bayesian theorem which finds the probability of an event occurring giving the probability on another event that has already occurred. The NB classifier is specially designed for binary classification problems. It is very easy to construct and in many cases it outperforms some sophisticated methods when dealing with huge data.

Figure 1 shows the overall structure of our system that employs the KNN and NB classifiers. It contains three modules: data collecting, data sampling, and the classifier. The data collecting module collects the raw data and extracts the NetFlow information fields and then inserts these fields into a database. Table 2 shows the columns of the database that is used to store the NetFlow information. The data sampling module categorizes every database entry according to special rules into four categories: DNS requests, DNS responses, DNS normals, and DNS anomalies. Based on these four categories, the classifier will decide whether the traffic is benign or malicious.

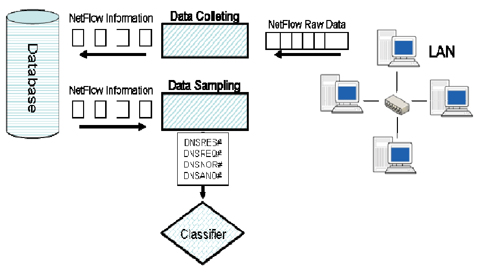


Figure 1. System structure.

Table 2. Database columns used to store NetFlow records.

Column Name	Data Stored
Timestamp	The time of sending the packets
SrcIP	Source IP
DstIP	Destination IP
SrcPort	Source port
DstPort	Destination port
Pckts	Number of packets
Bytes	Number of bytes
Srvc	Service
L4 Pro	Layer 4 protocol (TCP, UDP, ICMP, etc)
TCPFlag	TCP flags

4.2. Feature Set

Our set of features is based on our observations of scanning and email worms' behavior when they become active over the networks. As discussed in Section 3, a host infected by either type of worms initiates a significant amount of traffic that does not rely on DNS queries. Based on this fact, we extracted the features shown in Table 3 from the NetFlow records for each host connected to the network in a certain period of time.

Table 3. The set of features we used to train SVM.

Feature	Explanation
DNSREQ#	Number of DNS requests initiated
DNSRES#	Number of DNS responses to DNS requests
DNSNOR#	Number of flows sent based on previous DNS resolve. In other words, number of flows sent by using fully qualified domain names. However, any flows sent after 500 milliseconds of the last DNS resolve or they are not a DNSNOR tail are considered as DNS anomaly.
DNSANO#	Number of flows sent by a host without a DNS resolve. In other words, number of flows sent by using IP address rather than fully qualified domain name

The features are extracted explicitly by the sampling module which operates on regular basis. The operation time periods can be customized according to the sensitivity of the network information and the possibility of being attacked. Throughout our experiments, we used setting between 60-120 seconds for the operation time period. The process of features extraction includes two steps: classifying and calculating. During the classifying step, for all the NetFlow records captured, sampling module investigates chronologically every record to classify it as: DNSREQ, DNSRES, DNSNOR, or DNSANO. The rules that were followed during the classification are described in Table 4.

Table 4. The rules followed to classify every Netflow record.

NetFlow Record	The Condition
DNSREQ	IF SrcPort is greater than 1023 & DstIP equals DNS server IP & DstPort equals 53
DNSRES	IF SrcIP equals DNS server IP & SrcPort equals 53 & DstIP equals IP that made a DNSREQ & DstPort equals port# that a DNSREQ was sent from
DNSNOR	IF SrcIP = IP that made a DNSREQ and has received DNSRES or it is a tail of DNSNORs & DNSREQ last time is less than DNS Age (DNS Age = 500 milliseconds)
DNSANO	Otherwise

According to Microsoft, the standard port for DNS server is 53 and the DNS request should be initiated from port number greater than 1023. The value of DNS Age 500 millisecond has been selected carefully after investigating about 500 DNS requests from different IPs. However, we used the DNS age as a variable to suit the different settings of networks. To facilitate the process of feature extraction, we cite some examples of NetFlow records in Table 5 along with their category.

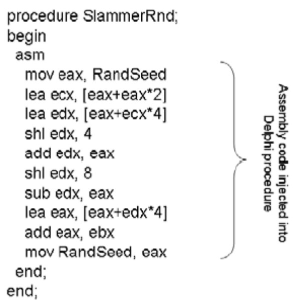
Table 5. Examples of NetFlow records captured, it shows an explanation of feature extraction process.

Type	Timestamp	SrcIP	DstIP	SrcPort	DstPort	.....	Justification
DNSANO	4/1/2011 12:17.320	192.168.1.11	192.168.1.56	1105	5358	.....	DNSANO because of no previous DNSREQ
DNSREQ	4/1/2011 12:56.140	192.168.1.11	192.168.0.2	2115	53	.....	DNSREQ because it has been sent from port# greater than 1023 to destination port of 53 on the server
DNSRES	4/1/2011 12:56.165	192.168.0.2	192.168.1.11	53	2115	.....	DNSRES because there was a DNSREQ within last 500 milliseconds from the same IP and port#
DNSNOR	4/1/2011 12:56.210	192.168.1.11	192.168.1.13	138	230	....	DNSNOR because there was a DNSRES within last 500 milliseconds from the same IP
DNSNOR	4/1/2011 12:56.610	192.168.1.11	192.168.1.13	138	230	....	DNSNOR because it is a tail for the previous DNSNOR within the last 500 milliseconds

After finishing the classifying step, the calculating step starts. This step is very simple and straightforward. It involves one function to calculate the number of DNSREQ, DNSRES, DNSNOR, and DNSANO records and create the Feature Pattern (FP) which represents the traffic activities initiated by a host within 120 seconds. The fp consists of four columns as in the following:

$$fp(1)=DNSREQ\# \qquad fp(2)=DNSRES\#$$
$$fp(3)=DNSNOR\# \qquad fp(4)=DNSANO\#$$

Two different sets of worms' variants were used throughout the research. First, we created semi-real variants of Slammer and Storm worms for using them in developing and testing the software system. The semi-real variants of these two worms were created by injecting the Assembly language codes that are responsible for activating the worm into a customized program. The result was worms without harmful parts. Figure 2 gives an example of creating semi-slammer worm. The second set was 18 real-life variants obtained from a publicly available malware databases [5, 28]. We used these variants in training and testing KNN and NB classifiers.



```
procedure SlammerRnd;
begin
 asm
   mov eax, RandSeed
   lea ecx, [eax+eax*2]
   lea edx, [eax+ecx*4]
   shl edx, 4
   add edx, eax
   shl edx, 8
   sub edx, eax
   lea eax, [eax+edx*4]
   add eax, ebx
   mov RandSeed, eax
 end;
end;
```

Assembly code injected into Delphi procedure

Figure 2. Assembly code that represents slammer's pseudo-random number generator injected into delphi procedure.

5. The Experiments

5.1. Experiment Setup

Before approaching the experiments, we built a virtual network environment by using GNS 3 0.7.1 and VMW are workstation 7.0.0 software to simulate the functions of 3600 CISCO router and two switched segments of network. The using of virtual environment aims at programming and testing the software system. After the completion of programming the system, we put the same design of the network on the ground to conduct the experiments.

The real network design is used in training and testing the KNN and NB classifications. Figure 3 depicts the network design used in our experiments. To make our setup closer to reality, we setup the domain server to act as: proxy server, DHCP server, and DNS server. We also allocated one PC to function as Oracle database server, one PC to function as web server, and one PC to provide a shared folder that hosts different software and documents. We inherited the system more comprehensiveness by installing Debian GNU/Linux 4.0 on one PC. Furthermore, (5) students from IT specialty have been engaged to act as real network users.

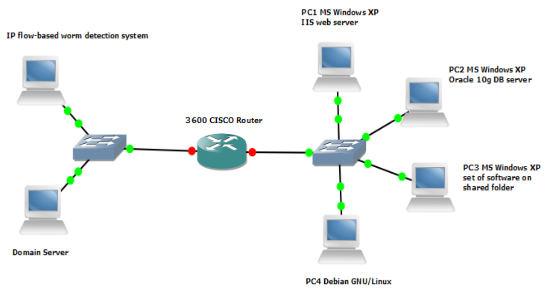


Figure 3. The experiment setup, two switched segments connected by a NetFlow enabled CISCO rout.

5.2. Data Set

Two different data sets were used, one for training and one for testing. The training data set was created by capturing 15 days NetFlow records. By more than 120 hours, we captured 125661 NetFlow records and created 6017 feature patterns 50% of which were benign and 50% were malicious. On each capturing day, there were two working sessions. During the first session, we clean the network to ensure non-existence of worms, and then we capture the benign traffic. In the second session, we inject 10 worms to capture malicious traffic: CodeRed, Slammer, Sasser, Witty, Doomjuice, Sobig, NetSky, MyDoom, Storm and Conficker. To create a consistent worm model, we apply a special filter to exclude the unneeded traffic.

The main theme of this filter is relying on monitoring the DHCP server log file. According to Microsoft, typically, the DHCP logs are saved as text files in the folder of "C:\WINNT\System32 \DHCP" on DHCP servers with naming format of "DhcpSrvLog-*.txt". From this log file we created a dynamic "node-living list" by collecting the event IDs: 10 (new lease) and 11 (renew a lease) periodically. Any traffic initiated by a node that is not listed in "node-living list" will be excluded. For testing, we used a data set of 3000 feature patterns created by capturing three days NetFlow records. First, we cleaned the network and captured 1500 benign feature patters. Then, we injected 11 real-life worms to create 1500 feature patterns: Welchia, Dabber, BlueCode, Myfip, Nimda, Sober, Bagle, Francette, Sasser, MyDoom, and Conficker.

6. Results and Discussion

This section discusses the results of comparing individual performance between KNN and NB classifiers. The comparison will be in terms of classification accuracy, false alarm rates, and training and prediction times. Initially, we establish the optimal KNN configuration to use in this comparison.

6.1. Optimal Number of Nearest Neighbors

As we mentioned in section 4-1, choosing K value is essential in building the KNN model. To estimate the value of K accurately, we used the n-fold cross validation technique. First, we divided the data set into 10 folds of randomly selected instances, each fold contains 600 instances. Consequently, we performed 10 iterations of training and testing such that 9 folds are used in each iteration for training and a different fold is held-out for testing. We repeated these 10 iterations 7 times on 7 different K values, 5, 7, 11, 15, 23, 25, and 29. Figure 4 shows how the classification accuracy is affected by the number of neighbors. The results show that the optimal value of K is 11 or 15 for a training data size of 6000 instances.

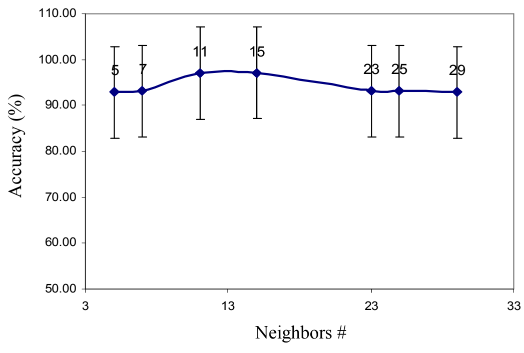


Figure 4. Impact of the number of neighbors (K) on the accuracy of KNN for a training data size of 6000 instances.

6.2. Comparison of KNN and NB Algorithms

Equipped with the optimal value for K=11, we compare the KNN and NB classifiers in terms of classification accuracy, false alarm rates, and training and prediction times. We performed 6 phases of training by using different data set sizes ranged from 1000-6000 instances. After each training phase, we tested the model by using dataset of 3000 instances, 1500 benign instances and 1500 malicious instances. Table 6 shows classification accuracies that resulted from these tests.

Table 6. The classification accuracy of KNN and NB on different dataset sizes.

Training Data Size	Classification Accuracy (%)	
	KNN	NB
1000	89.04	89.17
2000	88.97	85.84
3000	88.97	85.67
4000	88.97	85.64
5000	98.07	94.94
6000	98.07	96.97

To confirm which algorithm performs better, we conducted a two-sample statistical hypothesis test to compare the mean of above results. Initially, we put a null-hypothesis assumption that the two algorithms perform evenly. Then, we started gathering facts against this assumption. Using the "classification accuracy" as a predetermined performance metric, we used the paired t-test to show which algorithm, KNN or NB, is superior to another. The data shown in Figure 5 represents the results of the paired t-test.

	KNN	NB
Mean:	92.011	89.700
Variance:	22.011	25.643
Observations:	6	6
Pearson Correlation:	0.96	
Hypothesized Mean Difference:	0	
Df:	5	
t Stat:	3.857	
P(T<=t) one-tail:	0.006	
t Critical one-tail:	2.015	
P(T<=t) two-tail:	0.012	
t Critical two-tail:	2.571	

Figure 5. The results of t-test: Paired 2 samples for the means of classification accuracy.

As shown in Figure 5, the value of "t Stat" is greater than "t Critical two-tail". Also, the probability that our hypothesis is true, "P (T<=t) two-tail", is smaller than Alpha (0.05). Therefore, we reject the hypothesis and simply we can say that there is a significant difference between the classification accuracies, and by comparing the mean values, 92.011 and 89.700, we conclude that the KNN algorithm performs better than NB on our particular data set. Figure 6 compares graphically the KNN and NB classification accuracy. The graph confirms that the KNN being slightly more accurate than the NB. The graph in the Figure 7 compares the false alarm rates for the KNN and NB. For each training data size, the number of false alarms is the average of false-positives and false-negatives.

The results confirm that the NB suffers from a considerably higher number of false alarms for our particular data. Arguably, the training time is not a key performance measure since it can be done offline. While the prediction time is essential since we tend to have a prompt classifier that has ability to set an alarm in the very first moment of worm attack. The KNN is constantly outperformed by NB in terms of prediction time as it is much slower in prediction. In each prediction attempt, the KNN performs a calculation of the distance between the unknown class and all of the instances in the date set. However, the training and prediction times highly depend on the number of features for each instance as well as the training data set size.

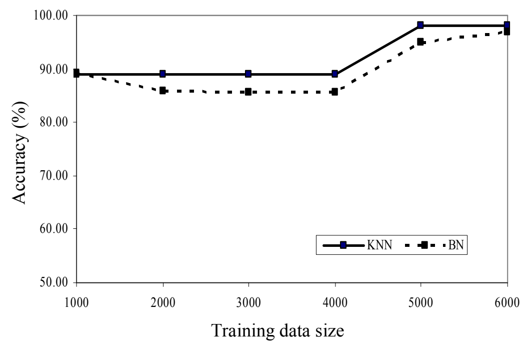


Figure 6. Comparing the KNN and NB classification accuracy on different sizes of data set.

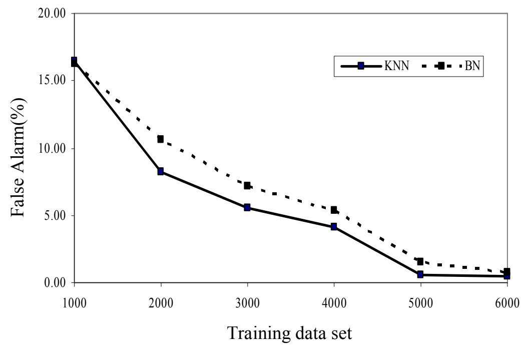


Figure 7. Comparing the KNN and NB false alarm rates on different sizes of data set.

7. Conclusions and Future Directions

In the worm detection research field, the ML and IP flow techniques have showed encouraging outcomes. We have investigated the optimal leveraging of the effectiveness of machine learning and the reliability of NetFlow data captured. We have demonstrated that the NetFlow data captured, after being properly sampled and analyzed, can be used for setting an alarm for a worm attack and consequently identifying the source of the suspicious payloads. For the classification purpose, we have employed two learning machines, KNN and NB. These two algorithms are among the most influential data mining algorithms in the research community. Both algorithms have demonstrated high performance and good accuracy rates. However, our statistical tests conclude that the KNN algorithm is slightly more accurate than the NB when using our particular data set, but the NB algorithm is much faster in terms of prediction time. To overcome our work limitations, the future work will focus on enhancing the feature pattern and increasing the entries of the training and testing data sets to detect P2P worms and Instant Messaging worms.

References

[1] Analysis: Sasser Worm, available at: http://www.eeye.com/Resources/SecurityCenter/Research/SecurityAdvisories/AD20040501, last visited 2011.

[2] Binsalleeh H. and Youssef A., "An Implementation for a Worm Detection and Mitigation System," *in Proceedings of the 24th Biennial Symposium on Communications*, Kingston, Jamaica, pp. 54-57, 2008.

[3] Claise B., "Cisco Systems NetFlow Services Export Version 9," available at: http://www.ietf.org/rfc/rfc3954.txt, last visited 2008.

[4] El-Halees A., "Filtering Spam E-Mail from Mixed Arabic and English Messages: A Comparison of Machine Learning Techniques," *the International Arab Journal of Information Technology*, vol. 6, no. 2, pp. 52-59, 2009.

[5] Global Hackers, available at: http://globalhackers.blogspot.com/2008/06/virus-collections.html, last visited 2011.

[6] Huang C-T., Thareja S., and Shin Y-J., "Wavelet-Based Real Time Detection of Network Traffic Anomalies," *in Proceedings of Securecomm and Workshops*, Baltimore, USA, pp. 1-7, 2006.

[7] Ishibashi K., Toyono T., Toyama K., Ishino M., Ohshima H., and Mizukoshi I., "Detecting Mass-Mailing Worm Infected Hosts by Mining DNS Traffic Data," *in Proceedings of the ACM SIGCOMM Workshop on Mining Network Data*, USA, pp. 159-164, 2005.

[8] Ismail I., Marsono M., and Nor S., "Detecting Worms using Data Mining Techniques: Learning in the Presence of Class Noise," *in Proceedings of the 6th International Conference on Signal-Image Technology and Internet-Based Systems*, Malaysia, pp. 187-194, 2010.

[9] Khayam S., Radha H., and Loguinov D., "Worm Detection at Network Endpoints using Information-Theoretic Traffic Perturbations," *in Proceedings of IEEE International Conference on Communications*, China, pp. 1561-1565, 2008.

[10] Kolter J. and Maloof M., "Learning to Detect Malicious Executables in the Wild," *in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge*

Discovery and Data Mining, USA, pp. 470-478, 2004.

[11] Lee W., Wang C., and Dagon D., *Botnet Detection: Countering the Largest Security Threat*, USA, Springer, 2010.

[12] Li P., Salour M., and Su X., "A Survey of Internet Worm Detection and Containment," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 1, USA, pp. 20-35, 2008.

[13] Masud M., Khan L., and Thuraisingham B., "Email Worm Detection using Data Mining," *in Proceedings of Techniques and Applications for Advanced Information Privacy and Security: Emerging Organizational, Ethical, and Human Issues*, USA, pp. 20-34, 2009.

[14] Matrosov A., Rodionov E., Harley D., and Malcho J., *Stuxnet Under the Microscope*, USA, ESET, 2011.

[15] Moore D., Paxson V., Savage S., Shannon C., Staniford S., and Weaver N., "Inside the Slammer Worm," *IEEE Security & Privacy*, vol. 1, no. 4, pp. 33-39, 2003.

[16] Musashi Y. and Rannenberg K., "Detection of Mass Mailing Worm-Infected PC Terminals by Observing DNS Query Access," *Science Links Japan*, *Gateway to Japan's Scientific and Technical Information*, vol. 2004, no. 129, pp. 39-44, 2004.

[17] Musashi Y., Matsuba R., and Sugitani K., "Indirect Detection of Mass Mailing Worms-Infected PC Terminals for Learners," *in Proceedings of the 3rd International Conference on Emerging Telecommunications Technologies and Applications*, Slovakia, pp. 233-237, 2004.

[18] Naiman D., "Statistical Anomaly Detection via HTTPD Data Analysis," *Computational Statistics & Data Analysis*, vol. 45, no. 1, pp. 51-67, 2004.

[19] Nguyen T. and Armitage G., "A Survey of Techniques for Internet Traffic Classification using Machine Learning," *IEEE Communications Surveys & Tutorials*, vol. 10, no. 4, pp. 56-76, 2008.

[20] Phaal P., Panchen S., and McKee N., "InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," available at: http://tools.ietf.org/html/rfc3176, last visited 2001.

[21] Porras P., Saidi H., and Yegneswaran V., "An Analysis of Conficker's Logic and Rendezvous Points," available at: http://mtc.sri.com/Conficker/, last visited 2009.

[22] Rasheed M., Norwawi N., Ghazali O., and Kadhum M., "Intelligent Failure Connection Algorithm for Detecting Internet Worms," *the International Journal of Computer Science and Network Security*, vol. 9, no. 5, pp. 280-285, 2009.

[23] Roesch M., Snort, Intrusion Detection System, available at: http://www.snort.org, last visited 2008.

[24] Schultz M., Eskin E., and Stolfo S., "Data Mining Methods for Detection of New Malicious Executables," *in Proceedings of the IEEE Symposium on Security and Privacy*, Oakland, New Zealand, pp. 38-49, 2001.

[25] Siddiqui M., Wang M., and Lee J., "Detecting Internet Worms using Data Mining Techniques," *Journal of Systemics*, *Cybernetics and Informatics*, vol. 6, no. 6, pp. 48-53, 2009.

[26] Sperotto A., Schaffrath G., Sadre R., Morariu C., Pras A., and Stiller B., "An Overview of IP Flow-Based Intrusion Detection," *IEEE Communications Surveys & Tutorials*, vol.12, no. 3, pp. 343-356, 2010.

[27] Tang Y., Luo J., Xiao B., and Wei G., "Concept, Characteristics and Defending Mechanism of Worms," *The Institute of Electronics, Information and Communication Engineers Transactions on Information and Systems*, vol. 92, no. 5, pp. 799-809, 2009.

[28] VX Heavens Virus Collection, available at: http://vx.netlux.org, last visited 2011.

[29] Wang W. and Luo D., "A New Attempt to Detect Polymorphic Worms Based on Semantic Signature and Data-Mining," *in Proceedings of the 1st International Conference of IEEE Communications and Networking*, China, pp. 1-3, 2006.

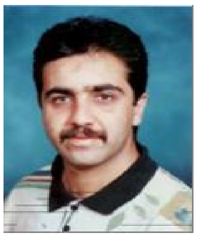
[30] Wei C., Sprague A., and Warner G., "Detection of Networks Blocks used by The Storm Worm Botnet," *in Proceedings of the 46th Annual Southeast Regional Conference*, Alabama, USA, pp. 356-360, 2008.

[31] Whyte D., Kranakis E., and Oorschot P., "DNS-Based Detection of Scanning Worms in an Enterprise Network," *in Proceedings of the 12th Symposium on Network and Distributed Systems Security*, pp. 181-195, 2005.

[32] Wu X., Kumar V., Quinlan J., Ghosh J., Yang Q., Motoda H., McLachlan G., Ng A., Liu B., and Yu P., "Top 10 Algorithms in Data Mining," *Knowledge and Information Systems*, vol. 14, no. 1, pp. 1-37, 2008.

[33] Zeng Y., Hu X., Wang H., Shin K., and Bose A., "Containment of Network Worms via Per-Process Rate-Limiting," *in Proceedings of the 4th International Conference on Security and Privacy in Communication Networks*, Istanbul, Turkey, pp. 14, 2008.

[34] Zou C., Gong W., and Towsley D., "Code Red Worm Propagation Modeling and Analysis," *in Proceedings of the 9th ACM Conference on Computer and Communications Security*, USA, pp. 138-147, 2002.



Shubair Abdullah is a PhD student at National Advanced IPv6 Centre of Excellence, University Sains Malaysia, Malaysia. He received his BSc degree in computer science from University of Basrah, Iraq in 1994 and MSc degree in computer sciences form University Sains Malaysia in 2007. His research interests include pattern recognition, network security, machine learning, and fuzzy inference systems.

Sureswaran Ramadass obtained his BsEE/CE (Magna Cum Laude) and Ms in electrical and computer engineering from the University of Miami in 1987 and 1990 respectively. He obtained his PhD from Universiti Sains Malaysia in 2000 while serving as a full time faculty in the School of Computer Sciences. Currently, he is a professor and the director of the National Advanced IPv6 Centre of Excellence at Universiti Sains Malaysia. His research interest includes multimedia streaming, network security and IPv6 deployment.

Altyeb Altaher received his BSc degree in computer science from International University of Africa in 2000. He received his Ms and PhD degrees in computer science from University of Khartoum in 2002 and 2007 respectively. Currently, he is a post doctoral research fellow at the National Advanced Ipv6 Center Universiti Sains Malaysia, before that he had been a senior lecturer at the computer science department, College of Computer Science and Information Technology, University of Sciences and Technology. His research interest include computer network monitoring, gird computing and computer network security.

Amer Al-Nassiri obtained his BSc degree in electrical engineering and MSc in computer science from University of Basrah, Iraq in 1977 and 1979, respectively. In 1996, he obtained his PhD in computer sciences. He has been working as faculty staff for almost 30 years. Currently, he is an associate professor in the Faculty of Computer Engineering and Computer Science in Ajman University of Science and Technology. His research interests include pattern recognition, arabic character recognition, data compression, and computer security.