Setting a Worm Attack Warning by using Machine Learning to Classify NetFlow Data

Shubair A. Abdulla
NAV6 Center of Excellence, Universiti Sains Malaysia, 11800 Penang, Malaysia

Sureswara Ramadass
NAV6 Center of Excellence, Universiti Sains Malaysia, 11800 Penang, Malaysia

Altyeb Altaher
NAV6 Center of Excellence, Universiti Sains Malaysia, 11800 Penang, Malaysia

Amer Al Nassiri
Ajman University of Science & Technology, Fujairah Campus, Fujairah, UAE

ABSTRACT
We present a worm warning system that leverages the reliability of IP-Flow and the effectiveness of machine learning techniques. Our system aims at setting an alarm in case a node is behaving maliciously. 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 features that uniquely identify worm's flow. The features are encapsulated into a set of feature patterns to train the support vector machines (SVM). A feature pattern includes: no of DNS requests, no of DNS responses, no of DNS normals, and no of DNS anomalies, for each PC on the network within a certain period of time. The SVM training is performed by using five of the most dangerous scanning worms: CodeRed, Slammer, Sasser, Witty, and Doomjuice as well as five email worms: Sobig, NetSky, MyDoom, Storm and Conficker. Eleven worms have been used during the test: Welchia, Dabber, BlueCode, Myfip, Nimda, Sober, Bagle, Francette, Sasser, MyDoom, and Conficker. The results of experiments manifest the soundness of the worm warning system.

General Terms
Machine Learning, IP Flow, Worm Detection.

Keywords
Intrusion detection systems, NetFlow, support vector machines, scanning worms, email worms.

1. INTRODUCTION
Nowadays, worms' inventors are continuously inventing malicious codes. They cause billions of dollars in damage to businesses around the world every year. They strive to discover the software defects in order to compromise systems, steal sensitive information, send spam emails, and generate distributed denial-of-service (DDoS) attacks. The security community has adopted Network Intrusion Detection (NID) systems to defend worms. These systems can be classified into two categories: behavior-based and content-based systems. In behavior-based systems, detection is based on watching anomalous behavior of the network. The poor accuracy is the major criticism against most of the existing behavior-based systems. They generate an alarm in the absence of worm attack (false-positive) and miss worm attack (false-negative). On the other hand, content-based systems look for signatures of worms within malicious traffic payload. These systems use a pre-compiled database of signatures, so they detect known worms but miss the unknown worms.

Generally, the NID systems inspect the payloads of every network packet to find known or unknown attacks [1], [2]. 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. 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 networked IP addresses within a certain period of time.

The flow can be exported by using an export mechanism such as CISCO NetFlow [3] and sFlow [4]. Although the information carried by flows is limited to the network nodes interactions, the volume of Flow Records is extremely huge. According to our experiments, the number of NetFlow Records exceeds 2000 within two hours in small network consisting of four PCs connected to one domain controller. Systems that employ IP Flow-based technique aggregate data-exchange information for every pair of IP addresses, and then encapsulate this information into a Flow Record. Figure 1 shows an example of IP Flow based system architecture. Two segments are connected through flow enable router that is responsible for capturing the IP flows and exporting them to a listening port on the flow record analyzer. The flow record analyzer performs a process of analyzing flow records to explore the status of the network and detect any intrusion.

The Flow Enable Router exports, for every node in segment1 exchange information with node in segment2, data-exchange information, such as: packet timestamp, source IP, destination IP, source port, destination port, and other useful protocol information.

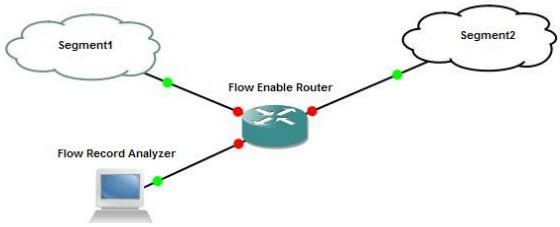

Fig 1: IP Flow-based intrusion detection system–basic architecture

Compared with other kinds of security threats, the worms are the most dangerous threat as they spread so quickly in the Internet. Typically, to perform spreading, updating, or any other mission, the worms do not rely on DNS to resolve a host name to a registered IP address as the vast majority of publicly available applications behave. However, based on our observations, there are legitimate applications and users that generally do not rely on DNS. Such exceptional scenarios are observed usually in carrying out connection maintenance procedures. In view of an enterprise network, these scenarios may lead to a decline in the detection accuracy for DNS-based systems. In this paper, we propose a system to set an initial alarm if a node initiates malicious traffic. It is an IP flow-based approach relying on the fact that the attacking worm uses IP addresses as the target for its infection attempt. We utilize the classification effectiveness of Support Vector Machines (SVM) to classify the worms' flows and the legitimate flows initiated by resources that do not rely on DNS. 18 worms were used to implement and evaluate the proposed system. The results of experiments reflect the effectiveness of the proposed approach.

The remainder of the paper is organized as follows: Section II describes related work by examining existing IP-flow, machine learning, and DNS anomaly based techniques with focus on the drawbacks. Section III provides background information on scanning and email worms. Section IV introduces a system that applies IP-flow and SVM to detect worms' flows. Section V explains our using of SVM. System implementation and evaluation are presented in Sections VI and VII. Finally, Section VIII concludes the paper and gives directions for future work.

2. RELATED WORK

In this section, we review some related work in the fields of IP Flow-based methods, machine learning and DNS anomaly-based methods with focus on the drawbacks that we are trying to overcome in our research.

2.1 IP Flow-based

Since the scans' threat is similar to worms' victim finding phase [1], some IP flow-based methods that are dedicated to defend against scan threats can be extended to worms detection [5,6,7]. The methods in [8] [9] employ IP flow-based technique to detect worms'. In [8], the authors focused on hit list worms' detection. A hit list worm probes a predefined list of hosts sequentially to find next victim. The algorithm slices the network according to a monitored protocol such as FTP, HTTP, or SMTP. Liu Bin et al [9] introduced a flow analysis and monitoring system based on NetFlow. Their system consists of a real-time anomalous traffic monitoring module equipped with two traffic static-based algorithms: variance similarity and Euclidean distance-based. It should be noted that, unlike our approach where we analyze the features of DNS queries, this approach uses ICMP and TCP_FLAGS information in the pattern matching. The main point that needs to be considered regarding these IP Flow-based methods is the extra load created on the monitoring and analysis systems as a consequence of worm attack. Searching within huge volume of flows for characteristics that identify worms uniquely is a very difficult and prone to error process. To interact efficiently with the huge volume of NetFlow records, we design a special sampling module which is responsible of categorizing the NetFlow records into: DNS requests, DNS responses, DNS normals, and DNS anomalies.

2.2 Machine Learning-based

Although machine learning has already been used for detecting malicious attacks, the authors in [10] mentioned that there have been few attempts to use data mining and machine learning techniques for the purpose of identifying unknown worms. 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 [11], [12], [13], [14]. 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: naive Bayes, SVM, Instance-Based k (IBk), and Term Frequency-Inverted Document Freq-uency (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. Our approach aims at analyzing the communication flows rather than payloads thus it runs with low computational processing. The most relevant to our work is [29] where the authors found that the worm actions grouped into 3 categories: Registry, file system, and network. They used SVM in 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.

2.3 DNS Anomaly-based

Using IP addresses by email worms obviates the need for a DNS query [15]. Based on this idea, some research is devoted to detect scanning worms by studying the DNS traffic. The method proposed by David Whyte et al [15] relies on the correlation of DNS queries with outgoing connections from an enterprise network to detect scanning worms. In 2008, H. Binsalleeh B. et al [16] proposed a system that followed the same architecture in [15] but the new system performs online processing of TCP dumps. The email worms also attracted the attention of the researchers. The methods in [17,18] are straightforward with focus on the volume of DNS queries for mail exchange (MX) to detect email worm infections. Ishibashi et al. [19] 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 [15] 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. We have employed the learning machine in our approach to overcome this shortcoming. We performed the training of SVM by using five of the most dangerous scanning worms: CodeRed, Slammer, Sasser, Witty, and Doomjuice as well as five email worms: Sobig, NetSky, MyDoom, Storm and Conficker. The results showed a good solution for the problem of isolating the legitimate traffic that does not rely on the DNS and could be initiated in certain exceptional circumstances.

3. SCANNING WORMS & EMAIL WORMS

A worm is a malicious program that self-propagates across the networks by exploiting the software vulner-abilities. According to the way that is used 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 computer worms and their categories can refer to [20, 21].

3.1 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 [21].

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.

3.2 Email Worms

This kind of worms spread by using email messages, they spread through HTML links or an email attachment. If the user opens the HTML link or email attachment, the worm will infect the computer and propagates by emailing itself using the user's address book. Email worms, such as Sobig, NetSky, and My-Doom, are programmed to drop backdoors, launch DoS, or 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 [22]. 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 is 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 a more updated list of "peer nodes" [23]. 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 is 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 [24]. 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 a number of requests for same port in a short period indicates that the host behaves abnormally.

4. SYSTEM STRUCTURE

Fig. 2 shows the overall structure of the system proposed. 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.

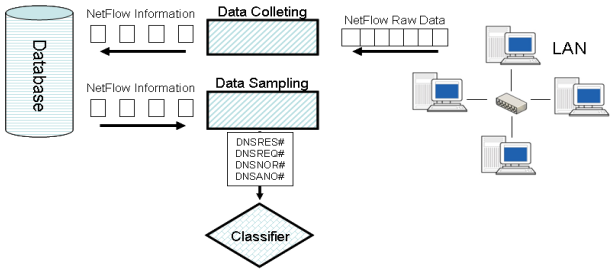

Fig 2: System Structure

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

The system has the following characteristics:

1. The system modules can be installed on one PC or on separated PCs.
2. The data collecting module is customizable to the network configuration in terms of routers' types, number of listening ports, and data collection's time period.
3. The system is intended to organize, store, and retrieve the NetFlow records efficiently by using database.
4. The data collecting module supports NetFlow 5 which is the most common version of NetFlow.

5. The SVM classifier that has been embedded into the system is BSVM [25] which is error free and commonly used for research purposes.

5. USE OF SUPPORT VECTOR MACHINES

In this section, we first present background information on Support Vector Machines. Next, we present how we applied the support vector machines along with the IP-flow features extraction process.

5.1 Support Vector Machines

Support Vector Machines (SVM) [26] is a supervised learning method for automatic pattern recognition. It appeared in the mid 1990s as an advanced computation learning theory combined with kernel functions. The SVM have been successfully applied to solve a large number of pattern recognition problems including image classification and hand-written character recognition. The SVM classification process involves dividing a given training data set by a separating hyperplane. The main difference that distinguishes between SVM and other machine learning is that the SVM is able to find the optimal separating hyperplane by minimizing the margin between the hyperplane and the training data points. If the data is linearly separable, the hyperplane will be a line, as shown in figure 3. And in the case that the data is not linearly separable, a kernel function should be used to remap non-linear data points into different dimensions where they can be separated linearly by a line, see figure 4. The kernel is the key parameter in SVM since it performs a crucial job of finding a way to separate the data points and thus to classify unknown data.

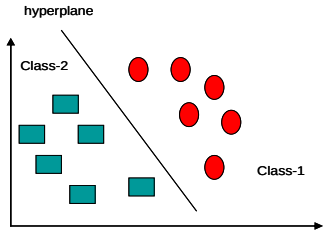

Fig 3: Dividing a data set into two classes by a hyperplane

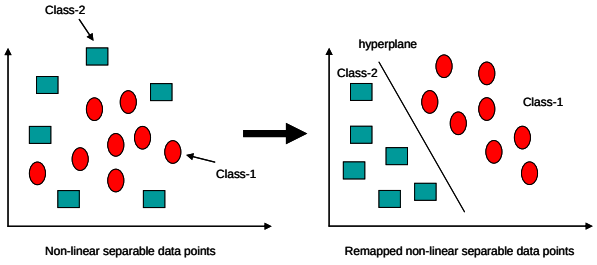

Fig 4: Kernel function remaps non-linear data points to divide them linearly by a hyperplane

The SVM cycle consists of learning and classifying phases. During the learning phase, the SVM are supplied with labeled training data to build a trained model that will be used in the classifying phase. In most cases, this model is derived with help of kernel function. Classifying unknown classes depends on the trained model, the more comprehensively the model is trained; the more successful the classification process. The process of building a trained model includes converting the input data into a format that comprises a set of features to be read by the SVM. The effectiveness of a feature set is measured by how accurately the SVM classifies unknown data into benign or malicious, and this effectiveness highly depends on features' comprehensiveness as well as uniqueness.

5.2 Feature Extraction

Our set of features used to build the trained model is based on our observations of scanning and email worms' behavior when they become active over the networks. As discussed in sections II and III, 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 feature extraction includes two steps: classifying and calculating. During the classifying step, for all the NetFlow records captured, the 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 ms)
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, including an explanation of feature extraction process

Type	Timestamp	SrcIP	DstIP	SrcPort	DstPort	.....
DNSANO	4/1/2011 12:17.320	192.168.1.11	192.168.1.56	1105	5358	.....
DNSREQ	4/1/2011 12:56.140	192.168.1.11	192.168.0.2	2115	53	.....
DNSRES	4/1/2011 12:56.165	192.168.0.2	192.168.1.11	53	2115	.....
DNSNOR	4/1/2011 12:56.210	192.168.1.11	192.168.1.13	138	230	....
DNSNOR	4/1/2011 12:56.610	192.168.1.11	192.168.1.13	138	230	....

Justification
DNSANO because of no previous DNSREQ
DNSREQ because it has been sent from port# greater than 1023 to destination port of 53 on the server
DNSRES because there was a DNSREQ within last 500 milliseconds from the same IP and port#
DNSNOR because there was a DNSRES within last 500 milliseconds from the same IP
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# fp(2) = DNSRES#
fp(3) = DNSNOR# fp(4) = DNSANO#

Since the SVM is supervised learning method, we labeled the fp created either 0 (benign) or 1 (malicious). Figure 5 shows the architecture of our SVM enabled worm detector. In the training phase, we captured the NetFlow records, produced the feature patterns and labeled them either malicious or benign depending on whether they are worm flows or legitimate flows, and finally, supplied the SVM with them. Consequently, the trained model is derived and used by the classifying phase to distinguish malicious and benign flows.

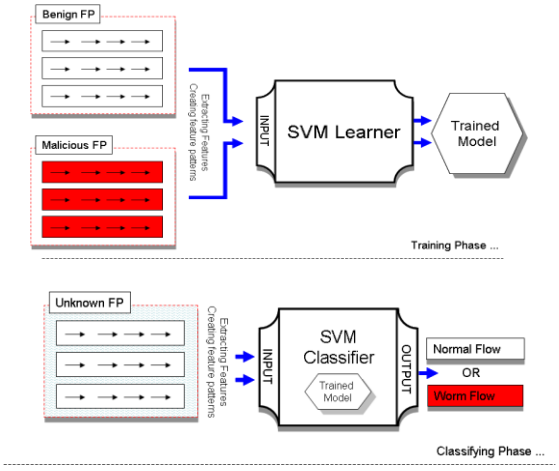

Fig 5: Our SVM algorithm operation cycle, the training and classifying phases

The experimental test has proved the effectiveness of the features and their impact on the classification of worms, as detailed in the following sections.

6. IMPLEMENTATION

6.1 Experiment Setup

Before performing the experiments, we built a virtual network environment by using GNS 3 0.7.1 and VMWare workstation 7.0.0 software to simulate the functions of 3600 CISCO router and two switched segments of network. This was done to test the software system. After the completion of programming the system, we put the same network design on the ground to conduct the experiments. The real network design is used in training and testing phases. Figure 6 depicts the network design used in our experiments. To make our setup closer to reality, we set up 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 increased the system 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.

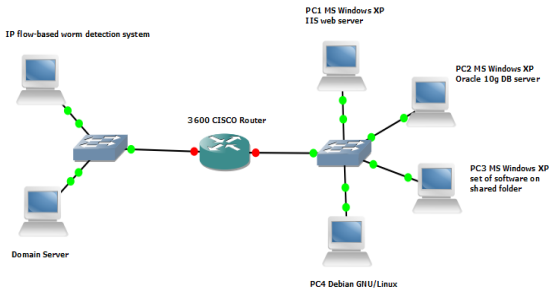

Fig 6: The experiment setup, two switched segments connected by a NetFlow enabled CISCO router

6.2 The Worms' Variants Used

Two different types of worm variants were used throughout the research. First, we created semi-real variants of Slammer and Storm worms to use them in developing and testing the software system. We created the semi-real variants of these two worms by injecting the Assembly codes that are responsible for activating the worm into a customized program. The result was two worms without harmful parts. Figure 7 gives an example of creating semi-slammer worm. The second set was 18 real-life variants obtained from a publicly available malware databases [27,30]. We used these variants in training and testing our algorithm.

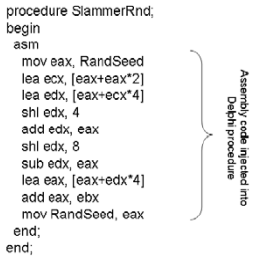

Fig 7: Assembly code that represents Slammer's Pseudo-Random Number Generator injected into Delphi procedure

6.3 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 it is free 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. See table 6 for more details about the data set used in the training phase.

For testing, we used a data set of 1000 feature patterns created by capturing three days NetFlow records. First, we cleaned the network and captured 500 benign feature patterns. Then, we injected 11 real-life worms to create 500 feature patterns: Welchia, Dabber, BlueCode, Myfip, Nimda, Sober, Bagle, Francette, Sasser, MyDoom, and Conficker.

6.4 Choosing the SVM kernel

The kernel and its optimal parameter values are the key to success in building an accurate trained model. There are three kernels that are commonly used: Linear, Polynomial, and Radial Basis Function (RBF). All of these kernels have been used and they have demonstrated similar accuracies in the experiments. However, we have reached the same conclusion reported in [28], which is that the linear kernel is an ideal choice since it outperformed the other two kernels in terms of training and prediction times.

Table 6 Monitoring and capturing of NetFlow records

Day	Beginning Time	Ending Time	Capturing Period	NetFlow Records	Sampling Period	Feature Pattern
25/05/11	10:27:18	17:20:00	6:52:42	5940	0:02:00	206
26/05/11	10:01:00	17:31:00	7:30:00	8350	0:02:00	225
28/05/11	9:55:00	18:16:54	8:21:54	5891	0:01:00	502
29/05/11	10:15:12	17:35:35	7:20:23	6660	0:01:00	440
30/05/11	9:59:14	18:25:10	8:25:56	7105	0:01:00	506
...	...	...	...	...	...	...
...	...	...	...	...	...	...
07/06/11	10:05:45	18:00:00	7:54:15	9198	0:01:30	316
08/06/11	9:36:30	18:02:00	8:25:30	9461	0:01:00	506
09/06/11	9:28:00	18:10:00	8:42:00	9482	0:01:30	348
11/06/11	9:51:43	18:15:00	8:23:17	9332	0:01:00	503
			Total	125661		6017

7. EVALUATION

To evaluate the SVM classifier, we repeated the test phase two times with same data set of 1000 feature patterns 50% of which were benign and 50% were malicious. Unlike the first test, the second one was preceded by a refinement process for the NetFlow records captured. The refinement process aimed at excluding the following unneeded traffic which was observed during the first test:

1. Traffic that was initiated by the router interfaces.
2. Traffic that was initiated by outsider IP addresses and destined to the DNS server, and
3. Traffic that was initiated by DNS in response to these outsiders IP addresses.

After the first test we found that involving the unneeded traffic in producing the feature patterns will greatly affect the accuracy of the SVM classifier. We excluded them by 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.

Refining NetFlow records to create feature patterns is the most influential factor on the classifier accuracy. To demonstrate level of influence of refinement process, we draw for each PC the false-positive (FPV) and the false-negative (FNV) rates resulted by test phase 1 and phase 2 in figures 8 and 9 respectively. The upper curves show the FPV and FNV rates before the process of refinement, while the lower curves show the new values of FPV and FNV rates after the refinement process.

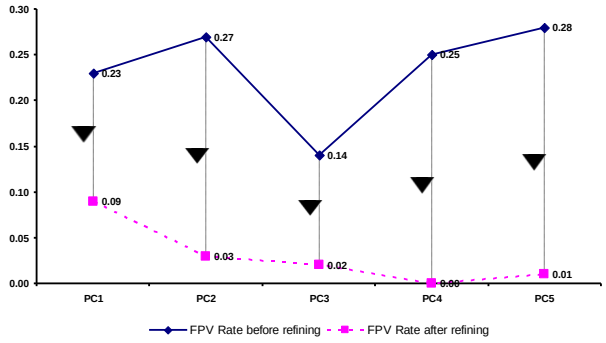

Fig 8: False-positive rates resulted before and after the refinement process for each PC

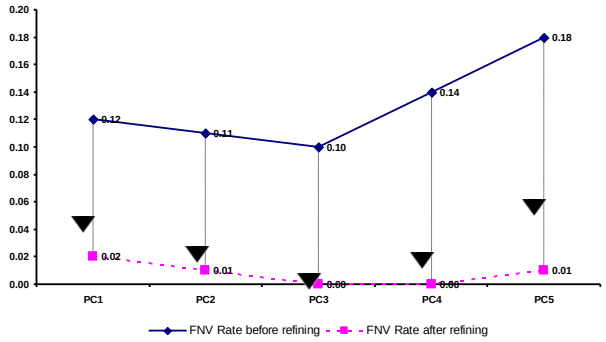

Fig 9: False-negative rates resulted before and after the refinement process for each PC.

Table 7 shows the results of computing the percentage change positively in the values of FPV and FNV rates. It is noticeable that the refinement process has improved the results of the tests where a 100% percentage change has been achieved in some PCs.

Table 7 The improvement in FPV and FNV rates after performing the refinement process

PCs	**1**	**2**	**3**	**4**	**5**
FPV Improvement	61%	89%	86%	100%	96%
FNV Improvement	83%	92%	100%	100%	94%

We can justify the presence of small rates of FPV and FNV to unusual situation caused by one user. An Oracle user has replaced the host parameter name of the tnsnames.ora file with an IP address and launched multiple nested SQL queries. As a result, the client PC generated a number of DNSANO close to the number of DNSANO created by the Sasser and MyDoom worms. However, this scenario is totally unusual as the humans tend to remember the names rather than the numeric IP addresses. Apart from this scenario, all the activities of other worms have been detected successfully by our system as the FPV and FNV rates are close to nil, see figure 10.

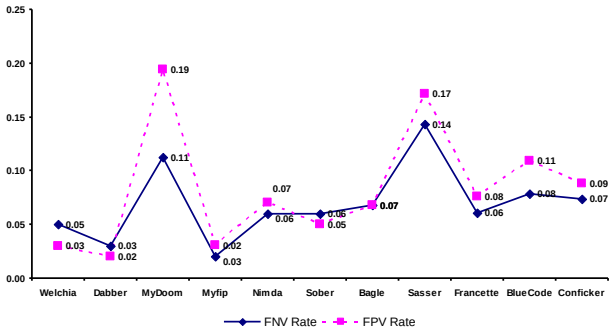

Fig 10: False-negative and false-positive rates for each worm, the high rates of FNV & FPV that resulted from unusual situation are in detection of Sasser and MyDoom

8. CONCLUSION AND FUTURE WORKS

In the worm detection research field, the machine learning and IP flow techniques show encouraging outcomes. We have investigated the optimal leveraging of the effectiveness of SVM and the reliability of NetFlow data captured. We have demonstrated that the captured NetFlow data, 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. In particular, we have addressed the problem of dealing with huge amount of NetFlow data to create feature patterns that identify the worms' flows uniquely. We have introduced a method of automatically generating worm variants by merging the instructions of Assembly and Delphi programming languages. This method has been applied to generate undamaging variants of Slammer and Storm worms used in testing the software system developed. 18 real-life worms were involved in the training and testing phases. The evidence of our approach's reliability was shown in the successful way of dealing with legitimate traffic that does not rely on DNS queries and could be originated by either normal users or network applications. Our future work will focus on enhancing the feature patterns and increasing the instances of the training and testing data sets to detect P2P worms and Instant Messaging worms.

9. REFERENCES

[1] Anna Sperotto, Gregor Schaffrath, Ramin Sadre, Cristian Morariu, Aiko Pras, and Burkhard Stiller, "An Overview of IP Flow-Based Intrusion Detection," Communications Surveys & Tutorials, IEEE, Vol. 12, No. 3. (2010).

[2] M. Roesch, "Snort, intrusion detection system," Jul. 2008. [Online]. Available: http://www.snort.org

[3] B. Claise, "Cisco Systems NetFlow Services Export Version 9," RFC 3954 (Informational), Jul. 2008. [Online]. Available: http: //www.ietf.org/rfc/rfc3954.txt

[4] P. Phaal, S. Panchen, and N. McKee, " InMon Corporation's sFlow: A Method for Monitoring Traffic in Switched and Routed Networks," RFC 3176 (Informational), Sep. 2001. [Online]. Available: http://tools.ietf.org/html/rfc3176

[5] A. Wagner and B. Plattner, "Entropy based worm and anomaly detection in fast IP networks," in Proc. of 14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE '05), June 2005, pp. 172–177

[6] Q. Zhao, J. Xu, and A. Kumar, "Detection of super sources and destinations in high-speed networks: Algorithms, analysis and evaluation," IEEE Journal on Selected Areas in Communications, vol. 24, no. 10, pp. 1840–1852, Oct. 2006

[7] Y. Gao, Z. Li, and Y. Chen, "A dos resilient flow-level intrusion detection approach for high-speed networks," in Proc. of the 26th IEEE International Conference on Distributed Computing Systems (ICDCS '06), 2006, p. 39

[8] M. Collins and M. Reiter, "Hit-list worm detection and bot identification in large networks using protocol graphs," in Proc. of 10th International Symposium on Recent Advances in Intrusion Detection (RAID'07), 2007, pp. 276–295

[9] L Bin, L Chuang, Q Jian, H Jianping, P Ungsunan, "A NetFlow based flow analysis and monitoring system in enterprise networks," Computer Networks (2008), Volume: 52, Issue: 5, Pages: 1074-1092

[10] Ismahani Ismail, Muhammad Nadzir Marsono, Sulaiman Mohd Nor, "Detecting Worms Using Data Mining Techniques: Learning in the Presence of Class Noise," sitis, pp.187-194, 2010 Sixth International Conference on Signal-Image Technology and Internet Based Systems, 2010

[11] E. Z. M. Schultz, E. Eskin and S. Stolfo, "Data Mining Methods for Detection of New Malicious Executables," in Proceedings of the IEEE Symposium on Security and Privacy, Los Alamitos,CA, 2001, pp. 38–49.

[12] J. Kolter and M. Maloof, "Learning to Detect Malicious Executables in the Wild," in Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, WA, USA, August 2004, pp. 470–478.

[13] W. Wang and D.-S. Luo, "A New Attempt to Detect Polymorphic Worms Based on Semantic Signature and Data-Mining," in First International Conference of IEEE Communications and Networking (ChinaCom '06), Beijing, China, October 2006, pp. 1–3.

[14] M. Siddiqui, M. C. WANG, and J. Lee, "Detecting Internet Worms Using Data Mining Techniques," Journal of Systemics, Cybernetics and Informatics, vol. 6, no. 6, pp. 48–53, 2009

[15] David Whyte, Evangelos Kranakis, Paul C. van Oorschot, " DNS-based Detection of Scanning Worms in an Enterprise Network," In Proceedings of The 12th Annual Network and Distributed System Security Symposium (February 2005)

[16] H. Binsalleeh and A. Youssef. "An implementation for a worm detection and mitigation system," Proc. of the 24th Biennial Symposium on Communications, pages 54–57, 2008.

[17] Y. Musashi, R. Matsuba, and K. Sugitani. "Indirect detection of mass mailing worms-infected pc terminals for learners," In 3rd International Conference on Emerging Telecommunications Technologies and Applications, pages 233{237, 2004

[18] Y. Musashi and K. Rannenberg. "Detection of mass mailing worm-infected pc terminals by observing dns query access," IPSJ SIG Notes, pages 39-44, 2004

[19] K. Ishibashi, T. Toyono, K. Toyama, M. Ishino, H. Ohshima, and I. Mizukoshi, "Detecting mass-mailing worm infected hosts by mining dns traffic data," in MineNet '05: Proc. of the 2005 ACM SIGCOMM Workshop on Mining Network Data. New York, NY, USA: ACM, 2005, pp. 159–164.

[20] P. Li, M. Salour, X. Su, " A survey of internet worm detection and containment, " Communications Surveys & Tutorials, IEEE, Vol. 10, No. 1. (2008)

[21] Tang, Yong; Luo, Jiaqing; Xiao, Bin; Wei, Guiyi, " Concept, Characteristics and Defending Mechanism of Worms, " IEICE Transactions on Information and Systems, Volume E92.D, Issue 5, pp. 799-809 (2009).

[22] Botnet Detection. Countering the Largest Security Threat. Spinger, 2008, vol. 36

[23] Wei, C., Sprague, A. and Warner, G. "Detection of Network Blocks Used by the Storm Worm Botnet,". In Proc. of 46th ACM Southeast Conference (2008)

[24] Phillip Porras, Hassen Saidi, and Vinod Yegneswaran, "An Analysis of Conficker's Logic and Rendezvous Points," March 2009.
[Online]. Available: http://mtc.sri.com/Conficker/

[25] Chih-Wei Hsu and Chih-Jen Lin, "BSVM," August, 2006. [Online]. Available: http://www.csie.ntu.edu.tw/~cjlin/bsvm/index.html

[26] C. Burges, "A Tutorial on Support Vector Machines for Pattern Recognition," Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121–167, 1998.

[27] VX Heavens Virus Collection, VX Heavens website. April 2011. [Online]. Available: http://vx.netlux.org.

[28] O. Sharma, M. Girolami, and J. Sventek, "Detecting worm variants using machine learning," in CoNEXT '07: Proceedings of the 2007 ACM CoNEXT conference, (New York, NY, USA), pp. 1–12, ACM, 2007.

[29] Yuanyuan Zeng, Xin Hu, Haixiong Wang, Kang G. Shin, and Abhijit Bose. 2008. "Containment of network worms via per-process rate-limiting". In Proceedings of the 4th international conference on Security and privacy in communication netowrks (SecureComm '08). ACM, New York, NY, USA

[30] Global hackers website. May 2011. [Online]. Available: http://globalhackers.blogspot.com/2008/06/virus-collections.html