Anomalous Payload-based Worm Detection and Signature Generation[1]

Ke Wang Gabriela Cretu Salvatore J. Stolfo

Computer Science Department, Columbia University
500 West 120th Street, New York, NY, 10027
{kewang, gcretu, sal}@cs.columbia.edu

Abstract. New features of the PAYL anomalous payload detection sensor are presented and demonstrated to accurately detect and generate signatures for zero-day worm exploits. Experimental evidence is presented to demonstrate that "site-specific models" trained and used for testing by PAYL are capable of detecting new worms with high accuracy in a collaborative security system. A new approach is proposed that correlates ingress/egress payload alerts to identify the worm's initial propagation. The method also enables automatic signature generation very early in the worm's propagation stage. These signatures can be deployed immediately to network firewalls and content filters to proactively protect other hosts. Finally, we also propose a collaborative security strategy whereby different hosts can themselves exchange PAYL signatures to increase accuracy and mitigate against false positives. The method used to represent these signatures is also privacy-preserving to enable cross-domain sharing. The important principle demonstrated is that the reduction of false positive alerts from an anomaly detector is not the central problem. Rather, correlating multiple alerts identifies true positives from the set of anomaly alerts and reduces incorrect decisions producing accurate mitigation.

1. Introduction

Zero-day worms are a serious wide-scale threat due to the monoculture problem. Large numbers of replicated vulnerable systems allow wide-spread infection. Furthermore, if any standard signature-based detector is blind to a zero-day attack, it is safe to say that all installations of that same detector are also blind to the same attack. The time from worm launch to wide-spread infestation is now very short, far shorter than the time to generate signatures for filtering, and certainly far shorter than the time to patch vulnerable systems. We consider the problem of detecting these "zero-day" attacks quickly and accurately upon their very first appearance, or very soon thereafter.

Some attacks exploit the vulnerabilities of a protocol; others seek to survey a site by scanning and probing. These attacks can often be detected by analyzing the

[1] This work has been partially supported by a contract with the Army Research Office/DHS, No. DA W911NF-04-1-0442 and an SBIR subcontract with the HS ARPA division of the Department of Homeland Security.

network packet headers, or monitoring the connection attempts and traffic volume. But some other attacks display normal protocol behavior except that they may carry malicious content in an otherwise normal connection. For example, *slow*-propagating worms and viruses targeting specific sites may not exhibit any unusual volumes of connection attempts, and hence may go unnoticed by worm detection sensors based upon scan or probe behavior.

We posit that analyzing the packet payload provides a reliable way to detect these attacks. State-of-the-art content-based detectors depend on "signatures" or "thumbprints" developed from known attacks, or a possibly error-prone specification of expected content, and hence may not be capable of detecting new attacks that were not covered by known examples or incomplete specifications. We focus this research on payload-based anomaly detection and seek to develop algorithms and systems for network intrusion detection that are light-weight and real-time and. The approach we take is to automatically learn the expected content flow in packets to hosts under normal operation of a system.

The PAYL anomaly detection sensor previously reported in [19] accurately models normal payload flowing to and from a site using unsupervised machine learning techniques. PAYL uses the models it computes to detect anomalous content that may indicate a new attack. The first principle behind PAYL is that a new zero-day attack will have content data never before seen by the victim host, and will likely appear quite different from normal data and be deemed anomalous. One of PAYL's innovations is the efficient means of modeling "normal data" effectively, as we describe shortly. Thus, PAYL is designed to detect the very first occurrences of an attack that exhibits anomalous content to stop the propagation of the new attack to many other potential victims.

Key features of worms include their self-propagation strategy and the means by which they seek new victims. A considerable amount of prior work depends upon the detection of worm-like scan/probe behavior to catch the worm propagation. We propose a new approach which is based on *ingress/egress anomalous payload correlation,* and uses *no* scan or probe information. The key idea is that a newly infected host will begin sending outbound traffic that is substantially similar (if not exactly the same) as the original content that attacked the victim (even if it is fragmented differently across multiple packets). Correlating ingress/egress anomalous payload can detect a worm propagation and stop the worm spread from the *very moment* it first attempts to propagate itself, instead of waiting until the volume of outgoing scans suggests full-blown propagation attempts. The important principle demonstrated is that the reduction of false positive alerts from an anomaly detector is not the central problem. Rather, correlating multiple alerts identifies true positives from the set of alerts and reduces incorrect decisions producing accurate mitigation. Furthermore, since this strategy is not dependent upon detecting scanning patterns, the approach may be applied to a broader class of worms. For example, worms like "Witty" target a specific set of IP addresses and exhibit no scanning behavior.

We do not propose to store and correlate all incoming packet content with outbound packets; that would be enormously expensive in space and time and may lead to many false alarms. Rather, we automatically identify a set of "suspect inbound packets", considered to contain anomalous content, and inspect them for anomalous outbound content directed to the same ports. The number of suspect packets is a function of the anomaly detector in PAYL and the particular traffic characteristics in which it is placed and the amount of training to compute stable models. In many of the environments in which PAYL has been tested, the number of anomalies is a very small percentage of the network traffic. Another important aspect of this strategy is that the correlated ingress/egress content anomalies are used to automatically generate *content-filtering signatures*. The overlapping content of the similar outgoing and incoming anomalous payloads are a natural set of candidate worm signatures. PAYL generates worm signatures from this shared content, which can be distributed over the network to other collaborating hosts to prevent any further worm infections.

In this paper, we will show that PAYL can successfully detect inbound worm packets with high accuracy and a low false positive rate. We will then show that if the worm has already infected a machine and starts to infect others, PAYL can quickly detect the propagation with an automatically generated signature that can be distributed to other machines in the local LAN or across domains. This signature is accurate, and won't block normal traffic (thus exhibiting a low false positive rate).

New and successful wide-scale infections occur on the internet with relative frequency. The *monoculture problem* applies not only to a high density of common vulnerable services and applications on the Internet, but it also applies to deployed security systems. If one standard commonly used open-source or COTS security system is blind to a new zero-day attack, then it is safe to say that all are blind to the same attack.

Some researchers have studied a solution to the monoculture problem by considering methods to diversify common application software, making each distinct site invulnerable to the same exact attack exploit [1]. We conjecture that systems that run the same services and software applications already exhibit diversity through their content flows. This provides the means of creating "site-specific" anomaly detectors capable of detecting new exploits, especially if many sites collaborate with each other and exchange alert information about suspicious packet content.

The core mindset of most security architectures dictates that each site or domain is an enclave, and any external site is regarded as the enemy. Worm writers and attackers, on the other hand, do collaborate and share information amongst themselves about vulnerabilities and tools to rapidly create new attack exploits, launch them, and form shared drone sites, often simultaneously worldwide. Defenders still depend on centralized management to update detection signatures and deploy patches on time scales that are no longer tenable.

We posit that a *collaborative security* system [16, 17], a distributed detection system that automatically shares information *in real-time* about anomalous behavior experienced at the moment of attack among collaborating sites, will substantially improve protection against wide-scale infections. Indeed, most collaborating systems can be protected against new exploits by limiting propagations to a small set of initial victims. By integrating the PAYL anomalous payload sensor into a collaborative security system, and exchanging information about suspect packet content, the resulting system not only can detect new zero-day exploits but can also automatically generate new zero-day attack signatures on-site for content filtering. In this paper, we demonstrate this strategy and show that a collaborative detection system using multiple PAYL sensors, each trained on a distinct site, can accurately detect an emerging worm outbreak very fast, and reduce the incidence of false positives to nearly zero.

PAYL has been under development for over a year and was first reported in the RAID 2004 conference [19], where many of the details about the underlying algorithms are fully described.

The rest of the paper is organized as follows. Section 2 discusses related work in worm detection and automatic signature generation. In Section 3, we give an overview of the PAYL detection sensor and demonstrate how well it can detect real-world worms. Section 4 presents an evaluation of the ingress/egress traffic correlation techniques, and the automatic worm signature generation. In Section 5 we introduce the idea of collaborative security among sites, and demonstrate its effectiveness using anomalous payload collaboration. Section 6 concludes the paper.

2 Related Work

Rule-based network intrusion detection systems such as Snort and Bro can do little to stop zero-day worms. They depend upon signatures only known after the worm has been launched successfully, essentially disclosing their new content and method of infection for later deployment. Shield [18] provides vulnerability signatures instead of string-oriented content signatures, and blocks attacks that exploit that vulnerability. The vulnerability signatures specify in general what an exploit would look like in the datagram of packets and a host-based "shield" agent would drop any connections that match this specification. A shield is manually specified for a vulnerability identified in some network available code, and is distributed to all desktops to provide protection against attacks. Once again, the time lag to specify, test and deploy shields from the moment the vulnerability is identified favors the worm writer, not the defenders.

Several researchers have considered the use of packet flows, and in some cases content, to attack the zero-day worm problem. Honeycomb [6] is a host-based intrusion detection system that automatically creates signatures. It uses a honeypot to capture malicious traffic targeting dark space, and then applies the longest common substring (LCS) algorithm on the N connections going to the same services. The computed substring is used as candidate worm signature.

Another system, Autograph [5] uses heuristics to classify the traffic into two categories: a flow pool with suspicious scanning activity and a non-suspicious flow pool. TCP flow reassembly is applied to the suspicious flow pool and they employ Rabin fingerprints to partition the payload into small blocks. These blocks are then counted to determine their prevalence, and the most frequent substrings from these blocks form the signatures. The signature generator uses blacklisting in order to decrease the number of false positives. They also describe collaboration between multiple sensors, but the sensors exchange only suspicious IPs and destination ports. This approach to sharing scan alerts is similar to other projects including the Worminator project [9] at Columbia University.

Earlybird [14] is another system that can automatically detect unknown worms and generate signatures for them. For each packet, the substrings computed by Rabin fingerprints are inserted into a frequency count table, incrementing a count field each time the substrings are encountered. The information about source and destination IPs is recorded. The table is stored in rank order by the frequency counts so that it produces the set of likely worm traffic. This system measures the prevalence of all common content in the network and then applies IP address dispersion, counting distinct source and destination IPs for each suspicious content, in order to keep the false positive rate small. This system is not used in collaboration between multiple sensors; it has been developed as a centralized system.

Each of the aforementioned projects are based on detecting frequently occurring payloads delivered by a source IP that is “suspicious”, either because the connection targeted dark IP space or the source IP address exhibited pre-scanning behavior. These approaches imply that the detection occurs *some time after* the propagation of the worm has executed. Unlike these approaches, PAYL does not depend on scanning behavior and payload prevalence. PAYL detects anomalous payloads immediately, and detects the first propagation attempt of the worms by correlating ingress/egress packet content alerts. PAYL has also been put to use in a system that automatically generates patches in a sandbox version of vulnerable software systems. See [13] for complete details. A more general discussion of related work in the area of anomaly detection can be found in [19].

3 Payload Based Anomaly Detection

3.1. Overview of the PAYL Sensor

The PAYL anomaly detection system is based on the principle that zero-day attacks are delivered in packets whose data is usual and distinct from all prior “normal content” flowing to or from the victim site. We assume that the packet content is available to the sensor for modeling. (Encrypted channels can be treated separately in various ways, such as the use of a host-sensor that captures content at the point of decryption, or by using a decryption/re-encryption proxy server. For the present paper, we simply assume the data is available for modeling.)

We compute a “normal profile” of a site’s unique content flow, and use this information to detect anomalous data. A “profile” is a model or a set of models that represent the set of data seen during training. Since we are profiling content data flows, the method must be general to work across all sites and all services, and it must be efficient and accurate. Our initial design of PAYL uses a “language independent” methodology, the statistical distribution of n-gram [2] values extracted from network packet datagrams. This methodology requires no parsing, no interpretation and no emulation of the content.

An n-gram is the sequence of n adjacent byte values in a packet payload. A sliding window with width n is passed over the whole payload one byte at a time and the frequency of each n-gram is computed. This frequency count distribution represents a “statistical centroid” or model of the content flow. The normalized average frequency and the variance of each gram are computed. The first implementation of PAYL uses the byte value distribution when n=1. The statistical means and variances of the 1-

grams are stored in two 256-element vectors. However, we condition a distinct model on the port (or service) and on packet length, producing a set of statistical centroids that in total provides a fine-grained, compact and effective model of a site's actual content flow. Full details of this method and its effectiveness are described in [19].

The first packet of CodeRed II illustrates the 1-gram data representation implemented in PAYL. Figure 1 shows a portion of the CRII packet, and its computed byte value distribution along with the rank ordered distribution is displayed in Figure 2, from which we extract the Z-string. The Z-string is a the string of distinct bytes whose frequency in the data is ordered from most frequent to least, serving as representative of the entire distribution, ignoring those byte values that do not appear in the data. The rank ordered distribution appears similar to the "Zipf distribution", and hence the name Z-string. The Z-string representation provides a privacy-preserving summary of payload that may be exchanged between domains without revealing the true content. Z-strings are not used for detection, but rather for message exchange and cross domain correlation of alerts. We describe this further in section 5.

```
   GET./default.ida?XXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXXXXXXXXXXXXXXXXXXXX
XXXXXX%u9090%u6858%ucbd3%u7
801%u9090%u6858%ucbd3%u7801%u
9090%u6858%ucbd3%u7801%u9090%
u9090%u8190%u00c3%u0003%u8b00
%u531b%u53ff%u0078%u0000%u0
```

Fig. 1. A portion of the first packet of CodeRed II

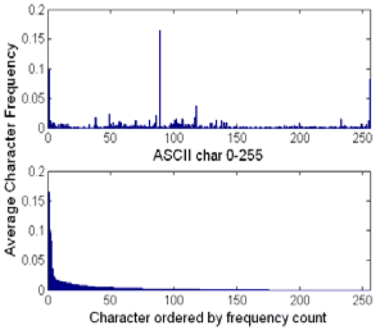

Fig. 2. The CRII payload distribution (top plot) and its ranked order distribution (bottom plot)

To compare the similarity between test data at detection time and the trained models computed during the training period, PAYL uses *Mahalanobis distance*. Mahalanobis distance weights each variable, the mean frequency of a 1-gram, by its standard deviation and covariance. The distance values produced by the models are then subjected to a threshold test. If the distance of a test datum is greater than the threshold, PAYL issues an alert for the packet. There is a distinct threshold setting for each centroid computed automatically by PAYL during a calibration step. To calibrate the sensor, a sample of test data is measured against the centroids and an initial threshold setting is chosen. A subsequent round of testing of new data updates the threshold settings to calibrate the sensor to the operating environment. Once this step converges, PAYL is ready to enter detection mode. Although the very initial results of testing PAYL looked quite promising, we devised several improvements to the modeling technique to reduce the percentage of false positives.

3. 2. New PAYL Features: Multiple Centroids

PAYL is a fully automatic, “hands-free” online anomaly detection sensor. It trains models and determines when they are stable; it is self-calibrating, automatically observes itself, and updates its models as warranted.

The most important new feature implemented in PAYL over our prior work is the use of multiple centroids, and ingress/egress correlation. In the first implementation, PAYL computes one centroid per length bin, followed by a stage of clustering similar centroids across neighboring bins. We previously computed a model M_{ij} for each specific observed packet payload length i of each port j. In this newer version, we compute a *set of models* M^k_{ij}, $k \geq 1$. Hence, within each length bin, multiple models are computed prior to a final clustering stage. The clustering is now executed across centroids within a length bin, and then again across neighboring length bins. This two stage clustering strategy also substantially reduces the memory requirements for models that represent normal content flow more accurately, revealing anomalous data with greater clarity (and higher distance values).

Since there might be different types of payload sent to the same service, e.g., pure text, .pdf, or .jpg, we used an incremental online clustering algorithm to create multiple centroids to model the traffic with finer granularity. This modeling idea can be extended to include centroids for different media that may be transmitted in packet flows. We note with interest that each of the different standard file and media types follow their own characteristic 1-gram distribution, so including models for standard file types can help reduce false positives. (The reader is encouraged to see [7] for a detailed analysis of this approach.)

The multi-centroid strategy requires a different test methodology. During testing, an alert will be generated by PAYL if a test packet matches none of the centroids within its length bin. The multi-centroid technique gives more accurate payload models and separates the anomalous payloads in a more precise manner.

For the present paper, a crucial issue we study is whether or not payload models are truly distinct across multiple sites. This is an important question in a collaborative security context. We have claimed that the monoculture problem applies not only to common services and applications, but also to security technologies. Hence, if a site is blind to a zero-day attack this implies that many other sites are blind to the same attack. Researchers are considering solutions to the monoculture problem by various techniques that “diversify” implementations. We conjecture that the content data flow among different sites is already diverse even when running the exact same services. Hence, each site’s profile will be substantially different from all others. A zero-day attack that may appear as normal data at one site, will likely not appear as normal data at other sites since the normal profiles are different. We test whether or not this conjecture is true by several experiments reported next.

3. 3 Data Diversity across Sites

In our previous work of PAYL [19], we have shown that byte distributions differ for each port and length. We also conjecture that it should be different for each host. For example, each web server contains different URLs, implements different functionality like web email or media uploads, and the population of service requests and responses sent to and from each site may differ, producing a diverse set of content profiles across all collaborating hosts and sites.

One of the most difficult aspects of doing research in this area is the lack of real-world datasets available to researchers that have full packet content for formal scientific study[2]. Privacy policies typically prevent sites from sharing their content data. However, we were able to use data from three sources, and show the distribution for each. The first one is an external commercial organization that wishes to remain anonymous, which we call **EX**. The others are the two web servers of the Computer Science Department of Columbia University, www.cs.columbia.edu and www1.cs.columbia.edu; we call these two datasets **W** and **W1,** respectively. The following plots show the profiles of the traffic content flow of each site.

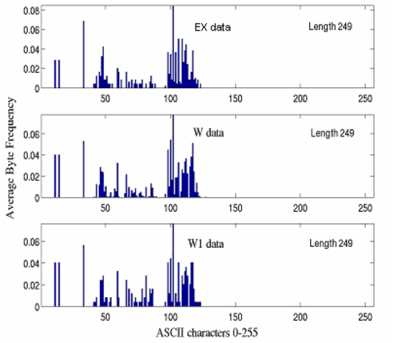

Fig. 3. Example byte distribution for payload length 249 of port 80 for the three sites EX, W, W1, in order from top to bottom

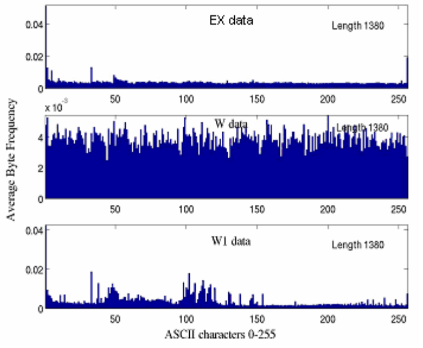

Fig. 4. Example byte distribution for payload length of 1380 of port 80 for the three sites EX, W, W1

The plots display the payload distributions for different packet payload lengths i.e. 249 bytes and 1380 bytes, spanning the whole range of possible payload lengths in order to give a general view of the diversity of the data coming from the three sites. Each byte distribution corresponds to the first centroid that is built for the respective payload lengths. We consider the analysis of the byte distribution of the three sites for different lengths because PAYL constructs models for each possible packet payload length.

We observe from the above plots that there is a visible difference in the byte distributions among the sites for the same length bin. This is confirmed by the values of Manhattan distances computed between the distributions, with results displayed in Table 1.

[2] Fortunately, HS ARPA is working to provide data to researchers through the PREDICT project; see www.predict.org.

Table 1. The Manhattan distance between the byte distributions of the profiles computed for the three sites, for each length bin.

	249 bytes	940 bytes	1380 bytes
MD(EX, W)	0.4841	0.6723	0.2533
MD(EX,W1)	0.3710	0.8120	0.4962
MD(W,W1)	0.3689	0.5972	0.6116

The content traffic among the sites is quite different. For example, the EX dataset is more complex containing file uploads of different media types (pdf, jpg, ppt, etc.) and webmail traffic; the W dataset contain less of this type of traffic while W1 is the simplest, containing almost no file uploads. Hence, each of the *site-specific* payload models is diverse, increasing the likelihood that a worm payload will be detected by at least one of these sites. To avoid detection, the worm exploit would have to be padded in such a way that its content description would appear to be normal concurrently for all of these sites.

Mimicry attacks are possible if the attacker has access to the same information as the victim. In the case of application payloads, attackers (including worms) would not know the distribution of the normal flow to their intended victim. The attacker would need to sniff each site for a long period of time and analyze the traffic in the same fashion as the detector described herein, and would also then need to figure out how to pad their poison payload to mimic the normal model. This is a daunting task. The attacker would have to be clever indeed to guess the exact distribution (the frequency and variances) as well as the threshold logic to deliver attack data that would go unnoticed. Additionally, any attempt to do this via probing, crawling or other means is very likely to be detected.

3. 4 Worm Detection Evaluation

In this section, we provide experimental evidence of the effectiveness of PAYL to detect incoming worms, using the three different datasets mentioned above. In our previous RAID paper [19], we showed PAYL's accuracy for the DARPA99 dataset, which contains a lot of artifacts that make the data too regular [8]. Here we report how PAYL performs over the three real-world datasets using known worms available for our research.

First, we describe the experimental setting. Since all three datasets were captured from real traffic, there is no ground truth, and measuring accuracy was not immediately possible. We thus needed to create test sets with ground truth, and we applied Snort for this purpose.

Each dataset was split into two distinct chronologically-ordered parts, one for training and the other for testing, following the 80%-20% rule. For each test dataset, we first created a clean set of packets free of any known worms still flowing on the Internet as background radiation. We then inserted the same set of worm traffic into the cleaned test set using tcpslice. Thus, we created ground truth in order to compute the accuracy and false positive rates.

The worm set includes CodeRed, CodeRed II, WebDAV, and a worm that exploits the IIS Windows media service, the nsiislog.dll buffer overflow vulnerability (MS03-

022). These worm samples were collected from real traffic as they appeared in the wild, from both our own dataset and from a third-party. Because PAYL only considers the packet payload, the worm set is inserted at random places in the test data. The ROC plots in Figure 5 show the result of the detection rate versus false positive rate over varying threshold settings of the PAYL sensor.

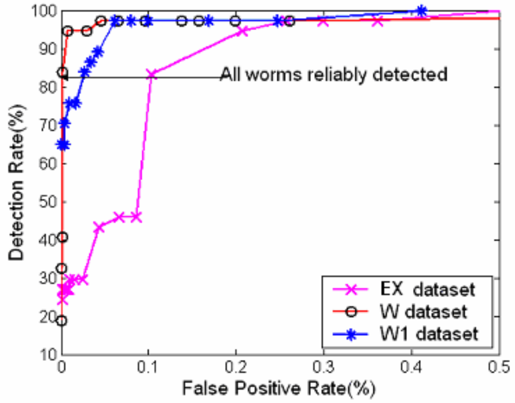

Fig. 5. ROC of PAYL detecting incoming worms, false positive rate restricted to less than 0.5%

The detection rate and false positive are both based on the number of packets. The test set contains 40 worm packets although there are only 4 actual worms in our zoo. The plots show the results for each data set, where each graphed line is the detection rate of the sensor where all 4 worms were detected. (This means more than half of each the worm's packets were detected as anomalous content.)

From the plot we can see that although the three sites are quite different in payload distribution, PAYL can successfully detect all the worms at a very low false positive rate. To provide a concrete example about how many alerts are generated at such low false positive rate, we measure the average false alerts per hour for these three sites. For 0.1% false positive rate, the EX dataset has 5.8 alerts per hour, W1 has 6 alerts per hour and W has 8 alerts per hour.

We manually checked the packets that were deemed false positives. Indeed, most of these are actually quite anomalous containing very odd abnormal payload. For example, in the EX dataset, there are weird file uploads, in one case a whole packet containing nothing but a repetition of a character with byte value **E7** as part of a word file. Other packets included unusual HTTP Get requests, with the referrer field padded with many "Y" characters (via a product providing anonymization).

We also tested the detection rate of the W32.Blaster worm (MS03-026) on TCP port 135 port using real RPC traffic inside Columbia's CS department. Despite being much more regular compared to HTTP traffic, the worm packets in each case were easily detected with zero false positives. Although at first blush, 5-8 alerts per hour may seem too high, a key contribution of this paper is a method to correlate multiple alerts to extract from the stream of alerts true worm events.

4 Worm Propagation Detection and Signature Generation by Correlation

In the previous section, we described the results using PAYL to detect anomalous packet content. To try and mitigate the scenario where a worm successfully evades detection and infects a protected host, we extended the detection strategy to model both inbound and outbound traffic from a protected host, computing models of content flows for ingress and egress packets. The strategy thus implies that within a protected LAN, some initial host victimized by a worm attack will begin a propagation sending outbound anomalous packets. When this occurs for any host in the LAN, we wish to inoculate all other hosts by generating and distributing worm packet signatures and their distribution to other hosts for content filtering.

We leverage the fact that self-propagating worms will start attacking other machines automatically by replicating itself, or at least the exploit portion of its content, shortly after a host is infected. (Polymorphic worms may randomly pad their content, but the exploit should remain intact.) For example, a machine gets infected by the Code Red II worm from some request received at port 80, and then this machine starts to send the same request to port 80 to other intended victims. This propagation pattern appears in every worm. So if we detect these anomalous *egress* packets to port *i* very similar to those anomalous *ingress* traffic to port *i*, there is a high probability that a worm that exploits the service at port *i* has started its propagation.

Note that these are the very first packets of the propagation, unlike the other approaches which have to wait until the host has already shown substantial amounts of unusual scanning and probing behavior. Thus, the worm may be stopped at its *very first propagation attempt* from the first victim even if the worm attempts to be slow and stealthy to avoid detection by probe detectors.

Careful treatment of port-forwarding protocols and services, such as P2P and NTP (Port 123) is required to apply this correlation strategy, otherwise normal port forwarding may be misinterpreted as worm propagations. Our work in this area involves two strategies, truncation of packets (focusing on control data) and modeling of the content of media [7]. This work is beyond the scope of this paper due to space limitations, and will be addressed in a future paper.

4.1. Ingress and Egress Traffic Correlation

When PAYL detects some incoming anomalous traffic to port *i*, it generates an alert and places the packet content on a buffer list of “suspects”. Any outbound traffic to port *i* that is deemed anomalous is compared to the buffer. The comparison is performed against the packet contents and a string similarity score is computed. If the score is higher than some threshold, we treat this as possible worm propagation and block or delay this outgoing traffic. This is different from the common quarantining or containment approaches which block all the traffic to or from some machine. PAYL will only block traffic whose content is deemed very suspicious, while all other traffic may proceed unabated maintaining critical services.

There are many possible metrics which can apply to decide the similarity between two strings. The several approaches we have considered, tested and evaluated include:

String equality (SE): this is the most intuitive approach. We decide that a propagation has started only if the egress payload is exactly the same as the ingress suspect packet. This metric is very strict and good at reducing false positives, but too sensitive to any tiny change in the packet payload. If the worm changes a single byte or just changes its packet fragmentation, the anomalous packet correlation will miss the propagation attempt.

Longest common substring (LCS): the next metric we considered is the LCS approach. LCS is less exact than SE, but avoids the fragmentation problem and other small payload manipulations. The longer the LCS that is computed between two packets, the greater the confidence that the suspect anomalous ingress/egress packets are more similar. The main shortcoming of this approach is its computation overhead compared to string equality, although it can also be implemented in linear time [3].

Longest common subsequence (LCSeq): this is similar to LCS, but unlike an LCS, the longest common subsequence need not be contiguous. LCSeq has the advantage of being able to detect polymorphic worms, but it may introduce more false positives.

For each pair of strings that are compared, we compute a *similarity score*, the higher the score, the more similar the strings are to each other. For SE, the score is 0 or 1, where 1 means equality. For both LCS and LCSeq, we use the percentage of the lcs or lcseq length out of the total length of the candidate strings. Let's say string s_1 has length L_1, and string s_2 has length L_2, and their lcs/lcseq has length C. We compute the similarity score as $2*C/(L_1+ L_2)$. This normalizes the score in the range of [0…1], where 1 means the strings are exactly equal. We show how well each of these measures work in Section 4.3.

Since we may have to check each outgoing packet (to port *i*) against possibly many suspect strings inbound to port *i*, we need to concern ourselves with the computational costs and storage required for such a strategy. On a real server machine, e.g., a web server, there are large numbers of incoming requests but very few, if any, outgoing requests to port 80 from the server (to other servers). So any outgoing request is already quite suspicious, and we should compare each of them against the suspects. If the host machine is used as both a server and a client simultaneously, then both incoming and outgoing requests may occur frequently. This is mitigated somewhat by the fact that we check only packets deemed anomalous, not every possible packet flowing to and from a machine. We apply the same modeling technique to the outgoing traffic and only compare the egress traffic we already labeled as anomalous.

4.2 Automatic Worm Signature Generation

There is another very important benefit that accrues from the ingress/egress packet content correlation and string similarity comparison: *automatic worm signature generation*. The computation of the similarity score produces the matching substring or subsequence which represents the common part of the ingress and egress malicious traffic. This common subsequence serves as a *signature content-filter*. Ideally, a worm signature should match worms and only worms. Since the traffic being compared is already judged as anomalous, and has exhibited propagation behavior – quite different from normal behavior – and the similar malicious payload is being sent to the same service at other hosts, these common parts are very possibly core exploit strings and hence can represent the worm signature. By using LCSeq, we may capture even polymorphic worms since the core exploit usually remains the same within each worm instance.

Thus, by correlating the ingress and egress malicious payload, we are able to detect the very initial worm propagation, and compute its signature immediately. Further, if we distribute these strings to collaborating sites, they too can leverage the added benefit of corroborating suspects they may have detected, and they may choose to employ content filters, preventing them from being exploited by a new, zero-day worm. We discuss this possibility further in Section 5.

4.3 Evaluation

In this section, we evaluate the performance of ingress/egress correlation and the quality of the automatically generated signatures.

Since none of the machines were attacked by worms during our data collection time at the three sites, we launched real worms to un-patched Windows 2000 machines in a controlled environment. For testing purposes, the packet traces of the worm propagation were merged into the three sites' packet flows as if the worm infection actually happened at each site. Since PAYL only uses payload, the source and target IP addresses of the merged content are irrelevant.

Without a complete collection of worms, and with limited capability to attack machines, we only tested Code Red and Code Red II out of the executable worms we collected. After launching these in our test environment and capturing the packet flow trace, we noticed interesting behavior: after infection, these two worms propagate with packets fragmented differently than the ones that initially infected the host. In particular, Code Red can separate "GET." and "/default.ida?" and "NNN…N" into different packets to avoid detection amongst many signature-based IDSes. The following table shows the length sequences of different packet fragmentation for Code Red and Code Red II.

Table 2. Different fragmentation for CR and CRII

Code Red (total 4039 bytes)	
Incoming	Outgoing
1448, 1448, 1143	4, 13, 362, 91, 1460, 1460, 649
	4, 375, 1460, 1460, 740
	4, 13, 453, 1460, 1460, 649
Code Red II (total 3818 bytes)	
Incoming	Outgoing
1448, 1448, 922	1460, 1460, 898

To evaluate the accuracy of worm propagation detection, we appended the propagation trace at the very end of one full day's network data from each of the three sites. When we collected the trace from our attack network, we not only captured the incoming port 80 requests, but also all the outgoing traffic directed to port 80. We checked each dataset manually, and found there is a small number of outgoing packets for the servers that produced the datasets W and W1, as we expected, and not a single one for the EX dataset. Hence, any egress packets to port 80 would be obviously anomalous without having to inspect their content. For this experiment, we captured all suspect incoming anomalous payloads in an unlimited sized buffer for comparison across all of the available data in our test sets.

We also purposely lowered PAYL's threshold setting (after calibration) in order to generate a very high number of suspects in order to test the accuracy of the string comparison and packet correlation strategies. In other words, we increased the noise (increasing the number of false positives) in order to determine how well the correlation can still separate out the important signal in the traffic (the actual worm content).

The result of this experiment is displayed in the following table for the different similarity metrics. The number in the parenthesis is the threshold used for the similarity score. For an outgoing packet, PAYL checks the suspect buffer and returns the highest similarity score. If it's higher than the threshold, we judge there is a worm propagation. False alerts suggest that an alert was mistakenly generated for a normal outgoing packet. The reason why SE does not work here is obvious: worm fragmentation blinds the method from seeing the worm's entire matching content. The other two metrics worked perfectly, detecting all the worm propagations with zero false alerts.

Table 3. Results of correlation for different metrics

	Detect propagate	False alerts
SE	No	No
LCS(0.5)	Yes	No
LCSeq(0.5)	Yes	No

To evaluate the false alerts more carefully, we decided to use some other traffic to simulate the outgoing traffic of the servers. For EX data, we used the outgoing port 80 traffic of other clients in that enterprise as if it originated from the EX server itself. For the W1 and W datasets, we used the outgoing port 80 traffic from the CS department. Then we repeated the previous experiments to detect the worm propagation with the injected outgoing traffic on each server. The result remains the same - using the same thresholds as before, we can successfully detect all the worm propagations without any false alerts.

In these experiments, we used an unlimited buffer for the incoming suspect payloads. The buffer size essentially stores packets for some period of time that is dependent upon the traffic rate, and the number of anomalous packet alerts that are generated from that traffic. That amount is indeterminate a priori, and is specific to both the environment being sniffed and the quality of the models computed by PAYL for that environment. Since CodeRed and CodeRed II launch their propagations immediately after infecting their victim hosts, a buffer holding only the most recent 5 or 10 suspects is enough to detect their propagation. But for slow-propagating or stealthy worms which might start propagating after an arbitrarily long hibernation period, the question is how many suspects should we save in the suspect buffer? If the ingress anomalous payloads has been removed from the suspect buffer before such a worm starts propagating, PAYL can no longer detect it by correlation. Theoretically, the larger the buffer the better, but there is tradeoff in memory usage and computation time. But for those worms that may hibernate for a long period of time, cross-site collaboration and exchange of suspect packet payloads might provide a solution. We discuss this in the next section.

As we mentioned earlier, the worm signature is a natural byproduct of the ingress/egress correlation. When we identified a possible worm propagation, the LCS or LCseq can be used as the worm signature. Figure 6 displays the actual content signatures computed for the CodeRed II propagations detected by PAYL in a style suitable for deployment in Snort. Note the signature contains some of the system calls used to infect a host, which is one of the reasons the false positive rate is so low for these detailed signatures.

```
|d0|$@|0 ff|5|d0|$@|0|h|d0| @|0|j|1|j|0|U|ff|
5|d8|$@|0 e8 19 0 0 0 c3 ff|%`0@|0 ff|%d0@|0
ff|%h0@|0 ff|%p0@|0 ff|%t0@|0 ff|%x0@|0 ff|%|
0@|fc fc fc fc fc fc fc fc fc fc fc fc fc fc
fc fc fc fc fc 0 0 0 0 0 0 0 0 0 0 0 0 0|\EXP
LORER.EXE|0 0 0|SOFTWARE\Microsoft\Windows NT
\CurrentVersion\Winlogon|0 0 0|SFCDisable|0 0
 9d ff ff ff|SYSTEM\CurrentControlSet\Service
s\W3SVC\Parameters\Virtual Roots|0 0 0 0|/Scr
ipts|0 0 0 0|/MSADC|0 0|/C|0 0|/D|0 0|c:\,,21
7|0 0 0 0|d:\,,217|fc fc fc fc fc fc fc fc fc
 fc fc fc fc fc fc fc fc fc fc fc fc fc fc fc
…
```

Fig. 6. The initial portion of the PAYL generated signature for CodeRed II.

We replicated the above experiments in order to test if any normal packet is blocked when we filter the real traffic against all the worm signatures generated. For our experiments we used the datasets from all the three sites, which have had the CRII attacks cleaned beforehand, and in all cases no normal packet was blocked.

5 Anomalous Payload Collaboration among Sites

Most current attack detection systems are constrained to a single ingress point within an enterprise without sharing any information with other sites. There are ongoing efforts that share suspicious source IP address [5, 9], but to our knowledge no such effort exists to share content information across sites in real time.

Here we focus on evaluating the detection accuracy of using collaboration among sites, assuming a scaleable, privacy-preserving secured communication infrastructure is available. (We have implemented a prototype in Worminator [9].)

Recall that, in Section 3.4, we described experiments measuring the diversity of the models computed at multiple sites. As we saw, the different sites tested have different normal payload models. This implies from a statistical perspective that they should also have different false positive alerts. Any “common or highly similar anomalous payloads” detected among two or more sites logically would be caused by a common worm exploit targeting many sites. Cross-site or cross-domain sharing may thus reduce the false positive problem at each site, and may more accurately identify worm outbreaks in the earliest stages of an infection.

To test this idea, we used the traffic from the three sites. There are two goals we seek to achieve in this experiment. One is to test whether different sites can help confirm with each other that a worm is spreading and attacking the Internet. The other is to test whether false alerts can be reduced, or even eliminated at each site when content alerts are correlated.

In this experiment, we used the following simple correlation rule: if two alerts from distinct sites are similar, the two alerts are considered true worm attacks; otherwise they are ignored. Each site's content alerts act as confirmatory evidence of a new worm outbreak, even after two such initial alerts are generated. This is very strict, aiming for the optimal solution to the worm problem.

This is a key observation. The optimal result we seek is that for any payload alerts generated from the same worm launched at two ore more sites, those payloads should be similar to each other, but not for normal data from either site that was a false positive. That is to say, if a site that generates a false positive alert about normal traffic it has seen, it will not produce suspect payloads that any other site will deem to be a worm propagation. Since we conjectured that each site's content models are diverse and highly distinct, even the false positives each site may generate will not match the false positives of other sites; only worms (i.e., true positives) will be commonly matched as anomalous data among multiple sites.

To make the experiment more convincing, we no longer test the same worm traffic against each site as in the previous section, since the sensor will obviously generate the exact same payload alert at all the sites. Instead, we use multiple variants of the same worm CodeRed and CodeRed II to test, which were extracted from real traffic and the launched worms. To make the evaluation strict, we tested different packet payloads for the same worm, and all the variant packet fragments it generates. We purposely lowered the PAYL threshold to generate many more false positives from each site than it otherwise would produce.

As in the case described above where we correlated ingress/egress packets to a host, the cross-site correlation uses the same metrics (SE, LCS and LCSeq) to judge whether two payload alerts are "similar". However, another problem that we need to consider when we exchange information between sites is *privacy*. It may be the case that a site is unwilling to allow packet content to be revealed to some external collaborating site. A false positive may reveal true content.

A packet payload could be presented by its 1-gram frequency distribution (see Figure 2). This representation already aggregates the actual content byte values in a form making it nearly impossible (but not totally impossible) to reconstruct the actual payload. (Since byte value distributions do not contain sequential information, the actual content is hard to recover. 2-gram distributions simplify the problem making it more likely to recover the content since adjacent byte values are represented. 3-grams nearly make the problem trivial to recover the actual content in many cases.)

However, we note that the 1-gram frequency distribution reordered into the *rank-ordered frequency distribution* produces a distribution that appears quite similar to an the exponential decreasing Zipf-like distribution. The rank-ordered distribution sorts the 1-gram distribution from the most frequent byte value to the least frequent. The rank ordering of the resultant distinct byte values is a string that we call the "Z-string" (as discussed in Section 3.1). One cannot recover the actual content from the Z-String. Rather, only an aggregated representation of the byte value frequencies is revealed, without the actual frequency information. This representation may convey sufficient information to correlate suspect payloads, without revealing the actual payload itself in almost all cases (except for the most trivial of payloads). Hence, false positive content alerts would not reveal true content, and privacy policies would be maintained among sites.

In this cross-domain correlation experiment we propose two more metrics which don't require exchanging raw payloads, but instead only the 1-gram distributions, and the privacy-preserving Z-string representation of the payload:

Manhattan distance (MD): Manhattan distance requires exchange of the byte distribution of the packet, which has 256 float numbers. Two distributions are similar, and hence two payloads are similar, if they have a small Manhattan distance. The maximum possible distance is 2. So we define the similarity score as (2-distance)/2, to normalize the score range to the same range of the other metrics described above.

LCS of Z-string (Zstr): While maintaining maximal privacy preservation, we perform the LCS on the Z-string of two alerts. The similarity score is the same as the one for LCS, but here the score evaluates the similarity of two Z-strings, not the raw payload strings.

Figure 7 presents the results achieved by sharing PAYL alerts among the three sites using CodeRed, CodeRed II and their variant packet fragments. The results are shown in terms of the similarity scores computed by each of the metrics. Each plot is composed of two different representations: one for false alerts (histogram) and the other for worm alerts (dots on the x-axis). The bars in the plots are histograms for the similarity scores computed for false PAYL alerts. The x-axis shows the similarity score, defined within the range [0...1], and the y-axis is the number of pairs of alerts within the same score range. The similarity scores for the worm alerts are shown separately as dots on the x-axis. The worm alerts include those for CodeRed, CodeRed II and their variant fragments. Note that all of the scores calculated between worm alerts are much higher than those of the "false" PAYL alerts and thus they would be correctly detected as true worms among collaborating sites. The alerts that scored too low would not have sufficient corroboration to deem them as true worms.

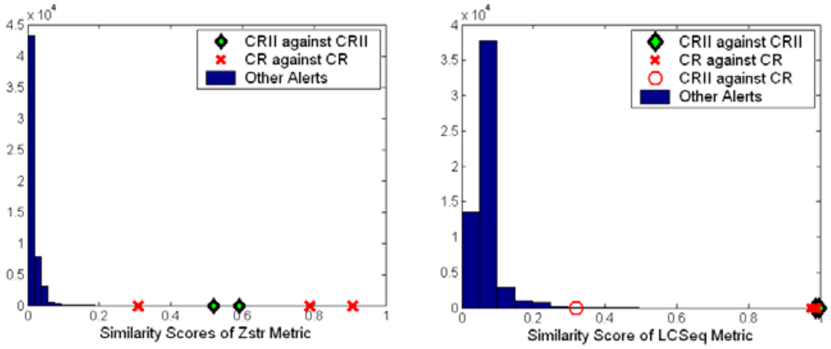

Fig. 7. Similarity scores of Zstr and LCSeq metrics for collaboration.

The above two plots show the similarity scores using Zstr and LCSeq metrics. LCS produced a similar result to LCSeq. String equality and Manhattan distance metrics did not perform well in distinguishing true alerts and false ones, so their plots are not shown here. The other two metrics presented in figure 7 give particularly good results. The worms and their variant packet fragments have much higher similarity scores than all the other alerts generated at each distinct site. This provides some evidence that this approach may work very well in practice and provide reliable information that a new zero-day attack is ongoing at different sites. Note too that each site can contribute to false positive reduction since the scores of the suspects are relatively low in comparison to the true worms. Furthermore, the Zstr metric shows the best separation here, and with the added advantage of preserving the privacy of the exchanged content.

There are two interesting observations we made from this data. The red circle in the LCSeq plot represents the similarity score when exchanging the alerts among the sites that PAYL generated for CodeRed and CodeRed II. LCSeq is the only metric that gave a relatively higher score that is worth noticing, while all the others provide less compelling scores. When we looked back at the tcpdump of CodeRed and CodeRed II, both of them contained the string:

"GET./default.ida?........u9090%u6858%ucbd3%u7801%u9090%u6858%ucbd3%......"

while CodeRed has a string of repeated "N", and CodeRed II has string of repeated "X" padding their content. Since subsequences do not need to be adjacent in the LCSsq metric, LCSeq ignored the repetitions of the unmatched "N" and X substrings and successfully picked out the other common substrings. LCS also performed higher-than-average score here, but not as good as LCSeq. This example suggests that polymorphic worms attempting to mask themselves by changing their padding may be detectable by cross-site collaboration under the LCSeq metric.

Another observation is that the LCSeq and LCS results display several packet content alerts with high similarity scores. These were false alerts generated by the correlation among the sites. The scores were measured at about 0.4 to 0.5. Although they are still much smaller than the worm scores, they are already outliers since they exceeded the score threshold used in this experiment. We inspected the content of these packets, and discovered that they included long padded strings attempting to hide the HTTP headers. Some proxies try to hide the query identity by replacing some headers with meaningless characters – in our case, consisting of a string of "Y"s. Such payloads were correlated as true alerts while using LCSeq/LCS as metrics, although they are not worms. However, these anomalies did not appear when we used the Zstr metric, since the long string of "Y's" used in padding the HTTP header only influences one position in the Z-string, but has no impact on the remainder of the Z-string.

These results suggest that cross-sites collaboration can greatly help identify the early appearance of new zero-day worms while reducing the false positive rates of the constituent PAYL anomaly detectors. The similarity score between worms and their variants are much higher than those between true false positives (normal data incorrectly deemed anomalies), and can be readily separated with high accuracy.

When several sites on the Internet detect similar anomalous payloads directed at them, they can confirm and validate with each other with high confidence that an attack is underway. As we mentioned earlier, this strategy can also solve the limited buffer size problem described in Section 4.3. If we only consider one single host, a stealthy worm can hibernate for a long period of time until a record of its appearance as an anomaly is no longer stored in the buffer of suspect packets. However, in the context of collaborating sites, the suspect anomaly can be corroborated by some other site that may also have a record of it in their buffer, as a remote site may have a larger buffer or may have received the worm at an earlier date. The distributed sites essentially serve as a remote long-term store of information, extending the local buffer memory available on the host. Furthermore, this strategy concurrently generates content filtering signatures. Any two sites that correlate and validate suspects as being true worms both have available the actual packet content from which to generate a signature, even if only Z-strings are exchanged between those sites.

6 Conclusion

In this paper, we provided experimental evidence that payload anomaly detection and content alert correlation, either on the host or across hosts and sites, holds promise for the early detection of zero-day worm outbreaks.

It is important to note that the range of worms tested and reported in the paper is limited in number and in scope. We hope that others with substantially larger zoos might make them available for testing, or to repeat the experiments reported herein to validate the results. Although we used real packet traces from three sources, a larger scale study of the methods described in this paper is necessary to understand whether the methods scale as we conjecture, and whether sites' content flows provide the necessary diversity to more readily detect common attack exploits that each may see during a worm outbreak.

PAYL can accurately detect new worms without signatures using machine learned models of normal network content traffic. Correlating content alerts generated by PAYL reduces false alarms, and generates detailed content signatures that may be used for filtering worm attacks at multiple sites. We believe that, over the next few years, worm writers will have substantially new and effective defenses to overcome, and we wish them nothing but failure and frustration in attempting to thwart these new generation of defensive systems. We further posit that the worm problem will ultimately be solved by defensive "coalitions", making the Internet and network systems in general safe from at least this class of cyber attacks for the foreseeable future.

Acknowledgments

We'd like to thank Janak J. Parekh, Wei-Jen Li for help on collecting data, the experimental set up, and for useful discussions and helpful comments on this paper.

References:

[1] S. Bhatkar. D. C. DuVarney, R. Sekar. Address Obfuscation: an Efficient Approach to Combat a Broad Range of Memory Error Exploits, *12th USENIX Security Symposium*, 2003.

[2] M. Damashek. Gauging similarity with n-grams: language independent categorization of text. *Science*, 267(5199):843--848, 1995

[3] D. Gusfield. Algorithms on Strings, Trees and Sequences, *Cambridge University Press*, 1997.

[4] J. O. Kephart and W. C. Arnold. Automatic extraction of computer virus signatures. In *Processing of the 4th International Virus Bulletin Conference*, Sept. 1994.

[5] K.-A Kim and B. Karp. Autograph: Toward Automated Distributed Worm Distribution, *In Proceedings of the USENIX Security Symposium,* August 2004.

[6] C. Kreibich and J. Crowcroft. Honeycomb-Creating Intrusion Detection Signatures Using Honeypots, *In Proceedings of the 2nd Workshop on Hot Topics in Networks (HotNets-II)*, November 2003.

[7] W. Li and S. Stolfo. Fileprints: Identifying File Types by N-gram Analysis, *CU technical report (in preparation, available at www.cs.columbia.edu/ids)*, Feb 2005.

[8] R. Lippmann, et al. The 1999 DARPA Off-Line Intrusion Detection Evaluation, *Computer Networks* 34(4) 579-595, 2000.

[9] M. Locasto, J. Parekh, S. Stolfo, A. Keromytis, T. Malkin and V. Misra. Collaborative Distributed Intrusion Detection, *Columbia University Tech Report CUCS-012-04,* 2004.

[10] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford and N. Weaver. The Spread of the Sapphire/Slammer Worm, http://www.cs.berkeley.edu/~nweaver/sapphire/

[11] D. Moore and C. Shannon. Code-Red: A Case Study on the Spread and Victims of an Internet Worm, *In Proceeding of the 2002 ACM SIGCOMM Internet Measurement Workshop (IMW 2002)*, November 2002.

[12] D. Moore, C. Shannon, G. Voelker, and S. Savage. Internet Quarantine: Requirements for Containing Self-Propagating Code. In *IEEE Proceedings of the INFOCOM*, Apr. 2003.

[13] S. Sidiroglou and A. D. Keromytis. Countering Network Worms through Automatic Patch Generation. To appear in *IEEE Security and Privacy* 2005.

[14] S. Singh, C. Estan, G. Varghese and S. Savage. Automated Worm Fingerprinting, *Sixth Symposium on Operating Systems Design and Implementation (OSDI),* 2004.

[15] S. Staniford, V. Paxson, and N. Weaver. How to Own the Internet in Your Spare Time. In *Proceedings of the USENIX Security Symposium*, Aug. 2002.

[16] S. Stolfo. Collaborative Security, The Black Book on Corporate Security, Ch 9. *Larstan publishing*, 2005.

[17] V. Yegneswaran, P. Barford, and S. Jha. Global Intrusion Detection in the DOMINO Overlay System. In *Proceedings of Network and Distributed System Security Symposium (NDSS),* Feb, 2004.

[18] H. J. Wang, C. Guo, D. R. Simon, and A. Zugenmaier. Shield: Vulnerability-Driven Network Filter for Preventing Known Vulnerability Exploits. In *Proceedings of the ACM SIGCOMM Conference*, Aug. 2004.

[19] K. Wang and S. Stolfo. Anomalous payload-based network intrusion detection, in *Proceedings of Recent Advance in Intrusion Detection (RAID),* Sept. 2004.