An Automated Signature-Based Approach against Polymorphic Internet Worms

Yong Tang and Shigang Chen

Abstract—Capable of infecting hundreds of thousands of hosts, worms represent a major threat to the Internet. However, the defense against them is still an open problem. This paper attempts to answer an important question: How can we distinguish polymorphic worms from normal background traffic? We propose a new worm signature, called the *position-aware distribution signature* (PADS), which fills the gap between traditional signatures and anomaly-based intrusion detection systems. The new signature is a collection of position-aware byte frequency distributions. It is more flexible than the traditional signatures of fixed strings while it is more precise than the position-unaware statistical signatures. We propose two algorithms based on Expectation-Maximization (EM) and Gibbs Sampling to efficiently compute PADS from a set of polymorphic worm samples. We also discuss how to separate a mixture of different polymorphic worms such that their respective PADS signatures can be calculated. We perform extensive experiments to demonstrate the effectiveness of PADS in separating new worm variants from normal background traffic.

Index Terms—Internet security, polymorphic worms, worm detection.

1 Introduction

An Internet worm is a self-propagating program that spreads across the Internet by replicating itself to vulnerable systems. It represents a huge threat to the network community [1], [2], [3], [4], [5], [6], [7]. Last year alone, the Internet witnessed the Santy worm, W32/Zafi.D, variants of W32/Sober, variants of W32/MyDoom, variants of W32/Bagle, W32/Sasser, etc. Much of the recent research concentrates on worm propagation models [8], [9], [10], [11], [12]. Defense against worm attacks remains largely an open problem. In most cases, because worms are able to spread substantially faster than humans can respond, people realize there is a worm attack only after the damage has already been done.

Intrusion detection has been intensively studied in the past decade. *Anomaly-based* systems [4], [13], [14] profile the statistical features of normal traffic. Any deviation from the profile will be treated as suspicious. Although these systems can detect previously unknown attacks, they have high false-positive ratios when the normal activities are diverse and unpredictable. On the other hand, *misuse detection* systems look for explicit attack indications, such as the pattern of a malicious traffic payload. They can detect known worms but will fail on new ones. Most widely deployed worm detection systems are *signature-based*, which puts them in the misuse-detection category. They look for specific byte sequences (called *attack signatures*) that are known to appear in the attack traffic. The signatures are traditionally identified by human experts through careful analysis of the byte sequence from captured attack traffic. A good signature should consistently show up in the attack traffic but rarely appear in the normal traffic.

The signature-based systems [15], [16] have advantages over the anomaly-based systems due to their simplicity and the ability to operate online in real time. The problem is that they can only detect known attacks with identified signatures. Generating signatures for new polymorphic worms is extremely difficult. Worms may be programmed to deliberately modify themselves each time they replicate and, thus, fool the defense system. This problem has so far been handled in an ad hoc way based on human judgment, which cannot keep up with the quick pace of worm mutation. To deal with polymorphic worms, the signature generation must be general enough to capture all attack traffic of a certain type while being specific enough to avoid overlapping with the content of normal traffic in order to reduce false positives.

In this paper, we summarize the polymorphism techniques that a worm may use to evade detection by the current defense systems. We then propose the *position-aware distribution signature* (PADS), which is capable of detecting polymorphic worms of certain types. The new signature is a collection of position-aware byte frequency distributions. It is more flexible than the traditional signatures of fixed strings while being more precise than the position-unaware statistical signatures. We propose two algorithms based on Expectation-Maximization [17] and Gibbs Sampling [18] to efficiently compute PADS from polymorphic worm samples. We describe how to match an incoming byte sequence against this "nonconventional" PADS, and we perform experiments to validate the effectiveness of this new signature. Four worms, MSBlaster, Sasser, Sapphire, and PUD, are used in the experiments. The results show that, by using the PADS signature derived from the past samples, we can accurately separate new worm variants from normal background traffic.

The rest of the paper is organized as follows: Section 2 covers related work. Section 3 studies worm polymorphism. Section 4 proposes a position-aware distribution signature. The algorithms for calculating such a signature are presented in Section 5. Section 6 generalizes the signature. Section 7 discusses how to handle multiple concurrent

- *Y. Tang is with Sunbelt Software, 33 N. Garden Ave., Suite 1200, Clearwater, FL 33755. E-mail: yongt@sunbelt-software.com.*
- *S. Chen is with the Department of Computer and Information Science and Engineering, University of Florida, Gainesville, FL 32611. E-mail: sgchen@cise.ufl.edu.*

Manuscript received 22 June 2006; revised 19 Oct. 2006; accepted 26 Oct. 2006; published online 16 Jan. 2007.
Recommended for acceptance by J. Hou.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDS-0165-0606.
Digital Object Identifier no. 10.1109/TPDS.2007.1050.

polymorphic worms. Section 8 presents the experiment results. Section 9 draws the conclusion.

2 Related Work

Much recent research on Internet worms concentrates on propagation modeling. A classic epidemiological model of computer viruses was proposed by Kephart and White [12]. This model was later used to analyze the propagation behavior of Code-Red-like worms by Staniford et al. [1] and Moore et al. [11]. Refinements were made to the model by Zou et al. [8] and Weaver et al. [19] in order to fit with the observed propagation data.

Chen et al. proposed a sophisticated worm propagation model (called AAWP [10]) based on discrete times. In the same work, the model is applied to monitor, detect, and defend against the spread of worms under a rather simplified setup, where a range of unused addresses monitored and a connection made to those addresses triggers a worm alert. The distributed early warning system by Zou et al. [9] also monitors unused addresses for the "trend" of illegitimate scan traffic on the Internet. There are two problems with these approaches: First, the attackers can easily overwhelm such a system with false positives by sending packets to those addresses or some normal programs may scan the Internet for research or other purposes and hit the monitored addresses. Second, to achieve good response time, the number of unused addresses to be monitored has to be large, but addresses are a scarce resource in the IPv4 world and only few have the privilege of establishing such a system. A monitor/detection system based on used addresses will be much more attractive. It allows more institutions or commercial companies to participate in the quest to defeat Internet worms.

For worms that propagate among certain types of servers, a solution is to block the servers' outbound connections so that the worms cannot spread among them. This approach works only when it is implemented for all or a vast majority of the servers on the Internet. Such an Internet-wide effort has not been and may never be achieved, considering that there are so many countries in the world and home users are setting up their servers without knowing this "good practice." In addition, the approach does not apply when a machine is used both as a server and as a client.

Moore et al. studied the effectiveness of worm containment technologies (*address blacklisting* and *content filtering*) and concluded that such systems must react in a matter of minutes and interdict nearly all Internet paths in order to be successful [2]. Williamson proposed modifying the network stack so that the rate of connection requests to distinct destinations is bounded [20], [21]. Schechter et al. [22] used the sequential hypothesis test to detect scan sources and proposed a credit-based algorithm for limiting the scan rate of a host. Weaver et al. [23] developed containment algorithms suitable for deployment with high-speed, low-cost network hardware. The main problem of the above approaches is that their effectiveness against worm propagation requires Internet-wide deployment. Gu et al. [24] proposed a simple two-phase local worm victim detection algorithm based on both infection pattern and scanning pattern. Apparently, it cannot issue a warning before some local hosts are compromised.

Honeypots [25] have gained a lot of attention recently. Their goal is to attract and trap the attack traffic on the Internet. Provos [26] designed a virtual honeypot framework to exhibit the TCP/IP stack behavior of different operating systems. Kreibich and Crowcroft [27] proposed the Honeycomb to identify the worm signatures by using longest common substrings. Dagon et al. developed HoneyStat [28] to detect worm behaviors in small networks. The above systems either assume that all incoming connections to the honeypot are from worms or rely on experts for the manual worm analysis. These restrictions greatly undermine the effectiveness of the systems.

Kruegel and Vigna [4] discussed various ways of applying anomaly detection in Web-based attacks. Several methods, such as the χ^2-test and Markov models, were presented. Wang and Stolfo [14] used the byte-frequency distribution of the traffic payload to identify anomalous behavior and, possibly, worm attacks. Autograph by Kim and Karp [29] and EarlyBird by Singh et al. [30] used pattern-based analysis to extract a single, contiguous, invariant string of a worm's payload for a signature. These methods are not effective against sophisticated polymorphic worms. The research in defending against polymorphic worms is in its infancy. Christodorescu and Jha [31] discussed a variety of different polymorphism techniques that could be used to obfuscate malicious code. They also proposed a static analysis method to identify malicious patterns in executables. Kolesnikov and Lee [32] described some advanced polymorphic worms that mutate based on normal traffic. TaintCheck by Newsome and Song [33] generated attack signatures using information about the specific software vulnerability being exploited. Newsome et al. proposed Polygraph [34], which identifies the disjoint, invariant elements in a worm's payload and uses these substrings as a signature. The effectiveness of the signature depends on how much invariant a worm will carry and how often the invariant will appear in normal traffic. Given that the invariant can be very limited, we observe that there is much more in a polymorphic worm that we can exploit. Developed in parallel, the statistical signature in this paper complements Polygraph by capturing not only the invariant elements, but also the variant elements that follow certain distributions.

3 Polymorphism of Internet Worms

The attackers will try every possible way to extend the lifetime of their worms. In order to evade the signature-based system, a polymorphic worm appears to be different each time it replicates itself. This section discusses worm polymorphism, while the next section provides a solution against some common polymorphism techniques.

There are many ways to make polymorphic worms. One technique relies on self-encryption with a variable key. It encrypts the body of a worm, which erases both the signatures and the statistical characteristics of the worm byte string. A copy of the worm, the decryption routine, and the key are sent to a victim machine, where the encrypted text is turned into a regular worm program by the decryption routine. The program is then executed to infect other victims. While different copies of a worm look different if different keys are used, the encrypted text tends to follow a uniform byte frequency distribution [35], which itself is a statistical feature that can be captured by anomaly detection based on its deviation from normal-traffic distributions [4], [14]. Moreover, if the same decryption routine is always used, the byte sequence in the decryption routine can serve as the worm signature.

A more sophisticated method of polymorphism is to change the decryption routine each time a copy of the worm is sent to another victim host. This can be achieved by keeping several decryption routines in a worm. When the worm tries

to make a copy, one routine is randomly selected and other routines are encrypted together with the worm body. The number of different decryption routines is limited by the total length of the worm. For example, consider a buffer-overflow attack that attempts to copy malicious data to an unprotected buffer. Oversized malicious data may cause severe memory corruption outside of the buffer, leading to system crash and spoiling the compromise. Given a limited number of decryption routines, it is possible to identify all of them as attack signatures after enough samples of the worm have been obtained.

Another polymorphism technique is called garbage-code insertion. It inserts garbage instructions into the copies of a worm. For example, a number of nop (i.e., no operation) instructions can be inserted into different places of the worm body, making it more difficult to compare the byte sequences of two instances of the same worm.

However, from the statistics point of view, the frequencies of the garbage instructions in a worm can differ greatly from those in normal traffic. If that is the case, anomaly-detection systems [4], [14] can be used to detect the worm. Furthermore, some garbage instructions such as nop can be easily identified and removed. Techniques of executable analysis can be used to identify certain other obfuscated garbage [31]. In the obfuscation-deobfuscation game between attackers and defenders, however, the existing techniques of executable analysis are unlikely to identify all obfuscated code introduced in sophisticated worms.

The instruction-substitution technique replaces one instruction sequence with a different but equivalent sequence. Unless the substitution is done over the entire code without compromising the code integrity (which is a great challenge by itself), it is likely that shorter signatures can be identified from the stationary portion of the worm. The code-transposition technique changes the order of the instructions with the help of jumps. The excess jump instructions provide a statistical clue, and executable-analysis techniques can help to remove the unnecessary jump instructions. Finally, the register-reassignment technique swaps the usage of the registers, which causes extensive "minor" changes in the code sequence.

The space of polymorphism techniques is huge and still growing. With the combinations of different techniques, a cure-all solution is unlikely. The pragmatic strategy is to enrich the pool of defense tools, each being effective against certain attacks. The current defense techniques fall in two main categories, 1) misuse detection/signature matching and 2) anomaly detection. The former matches against the known patterns in attack traffic. The latter matches against the statistical distributions of normal traffic. We propose a hybrid approach based on a new type of signature, consisting of position-aware byte frequency distributions. The new signatures can tolerate extensive, "local" changes as long as certain "global" characteristics remain. Good examples are polymorphism caused by register reassignment and modest instruction substitution. The new signatures may work with executable-analysis techniques to characterize statistical patterns that appear after certain garbage instructions and unnecessary jumps are removed. However, we do not claim that such signatures are suitable for all attacks, particularly extremely sophisticated worms that do not carry any position-dependent statistical features after obfuscation transformation that cannot be deobfuscated.

4 Position-Aware Distribution Signature (PADS)

4.1 Background and Motivation

Most deployed antivirus/worm systems are signature-based. They rely on an exact match of the packet payload with a database of fixed signatures. Though effective in dealing with the known attacks, they fail to detect new or variants of the old worms, especially polymorphic worms whose instances can be carefully crafted to circumvent the signatures [31]. Moreover, manually identifying the signatures may take days if not longer.

To address these problems, several anomaly-based systems [4], [14] use the *byte frequency distribution* (BFD) to identify the existence of a worm. Their basic approach is to profile the byte frequency distribution of normal network traffic. When a new incoming connection is established, the payload of the packets is examined. The byte frequency distribution of the connection is computed and compared with the distribution of normal traffic. A large deviation is considered suspicious. The problem is that an intelligent attacker can easily cheat the system by attaching the worm body to a lengthy normal session. Since the majority of the payload is from legitimate operations, its byte frequency distribution will not vary much from the normal traffic.

Both signature-based and anomaly-based systems have their pros and cons. The signature-based systems work well against the technique of attaching a worm to normal traffic, but they are weak against polymorphism. On the other hand, the anomaly-based systems are able to handle polymorphism only when the worm is largely separated from the background and does not carry too much garbage that distorts the distribution.

Our system inherits the positive aspects of both the signature-based and anomaly-based systems. It is based on a new defense technique that is complementary to the existing ones. We define a relaxed, inexact form of signatures that have flexibility against certain polymorphism. The new signature is called the *position-aware distribution signature* (PADS for short). It has a byte frequency distribution (instead of a fixed value) for each position in the signature "string." The idea is to focus on the generic pattern of the signature while allowing some local variation.

Consider a polymorphic worm with register reassignment (Section 3). Because registers are used extensively in executables, swapping registers is effective against traditional signatures. However, when a signature is expressed in position-aware distributions, not only are the static elements in the executable captured, but the set of likely values for the variable elements are also captured. Hence, PADS allows a more precise measurement of "matching." A similar example is instruction substitution, where the mutually replaceable instructions (or sequences) can be represented by the position-aware distributions.

4.2 PADS

We first describe what a PADS signature is, then explain how to match a byte sequence against a signature, and, finally, motivate how to compute such a signature from captured worm samples. How to capture worm samples is beyond the scope of this paper. Readers are referred to [36] for a honeypot approach and [33] for additional discussions.

At each byte position p of a PADS signature, the byte-frequency distribution is a function $f_p(b)$, which gives the probability for b to appear at position p, where $b \in [0..255]$,

TABLE 1
An Example of a PADS Signature with Width $W = 10$

b	0	1	2	...	9	10
0x00	0.001	0.001	0.001	...	0.500	0.100
0x01	0.001	0.001	0.001	...	0.200	0.500
0x02	0.005	0.001	0.001	...	0.001	0.100
...	...	...	...	...	...	...
0xfe	0.100	0.001	0.001	...	0.001	0.001
0xff	0.001	0.700	0.700	...	0.001	0.001

the set of possible values for a byte. $\sum_{b \in [0...255]} f_p(b) = 1$. We use $(f_1, f_2, \ldots f_W)$ to characterize the byte-frequency distribution of the worm, where W is the width of the signature in terms of the number of bytes. Let $f_0(b)$ be the byte frequency distribution of normal traffic. The PADS signature is defined as $\Theta = (f_0, f_1, f_2, \ldots f_W)$, which consists of a *normal signature* f_0 and an *anomalous signature* $(f_1, f_2, \ldots f_W)$. Table 1 gives an example of a PADS signature with width $W = 10$.

Consider a set of byte sequences $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$, where $S_i, 1 \leq i \leq n$, is the content of an incoming connection. We decide whether S_i is a variant of the worm by matching it against the signature Θ. Let l_i be the length of S_i. Let $S_{i,1}, S_{i,2}, \ldots, S_{i,l_i}$ be the bytes of S_i at position $1, 2, \ldots, l_i$, respectively. Let $seg(S_i, a_i)$ be the W-byte segment of S_i starting from position a_i. The matching score of $seg(S_i, a_i)$ with the anomalous signature is defined as

$$M(\Theta, S_i, a_i) = \prod_{p=0}^{W-1} f_p(S_{i,a_i+p}),$$

which is the probability for $seg(S_i, a_i)$ to occur, given the distribution $(f_1, f_2, \ldots f_W)$ of the worm. Similarly, the matching score of $seg(S_i, a_i)$ with the normal signature is defined as

$$\overline{M}(\Theta, S_i, a_i) = \prod_{p=0}^{W-1} f_0(S_{i,a_i+p}).$$

We want to find a position a_i that maximizes $M(\Theta, S_i, a_i)$ and minimizes $\overline{M}(\Theta, S_i, a_i)$. To quantify this goal, we combine the above two scores in order to capture both the "similarity" between $seg(S_i, a_i)$ and the anomalous signature and the "dissimilarity" between $seg(S_i, a_i)$ and the normal signature. For this purpose, we define $\Lambda(\Theta, S_i, a_i)$ as the matching score of $seg(S_i, a_i)$ with the PADS signature:

$$\Lambda(\Theta, S_i, a_i) = \frac{M(\Theta, S_i, a_i)}{\overline{M}(\Theta, S_i, a_i)} = \prod_{p=0}^{W-1} \frac{f_p(S_{i,a_i+p})}{f_0(S_{i,a_i+p})}. \quad (1)$$

The matching score of the byte sequence S_i with the signature is defined as the maximum $\Lambda(\Theta, S_i, a_i)$ among all possible positions a_i, that is,

$$\max_{a_i=1}^{l_x-W+1} \Lambda(\Theta, S_i, a_i).$$

Alternatively, we can use the logarithm of Λ as the score, which makes it easier to plot our experiment results. Our final matching score of S_i with the PADS signature Θ is defined as

$$\begin{aligned} \Omega(\Theta, S_i) &= \max_{a_i=1}^{l_i-W+1} \frac{1}{W} \log(\Lambda(\Theta, S_i, a_i)) \\ &= \max_{a_i=1}^{l_i-W+1} \sum_{p=0}^{W-1} \frac{1}{W} \log \frac{f_p(S_{i,a_i+p})}{f_0(S_{i,a_i+p})}, \end{aligned} \quad (2)$$

where $\frac{1}{W}$ serves as a normalization factor.

The *significant region* of S_i is defined as the W-byte segment whose matching score is equal to $\Omega(\Theta, S_i)$. The significant region is the segment of S_i that matches best with the PADS signature.

For any incoming byte sequence S_i, if $\Omega(\Theta, S_i)$ is greater than a threshold value, a warning about a (possibly variant) worm attack is issued. Additional defense actions may be carried out, e.g., rejecting the connection that carries S_i. The threshold is typically set at 0. From the definition of Ω, above zero means that S_i is closer to the anomalous signature $(f_1, f_2, \ldots f_W)$; below zero means that S_i is closer to the normal signature f_0.

Next, we discuss how to calculate Θ based on a set of previously collected worm samples. Each variant is a byte sequence with a variable length. It contains one copy of the worm, possibly embedded within a normal byte sequence. Now, let $\mathcal{S} = \{S_1, S_2, \ldots, S_n\}$ be the set of collected worm variants. Our goal is to find a signature with which the matching scores of the worm variants are maximized. We attempt to model it as the classical "missing data problem" in statistics and then apply the expectation-maximization algorithm (EM) to solve it.

To begin with, we know neither the signature, which is the unknown parameter, nor the significant regions of the variants, which are the missing data. Knowing one would allow us to compute the other. We have just shown how to compute the significant region of a byte sequence if the signature Θ is known. Next, we describe how to compute the signature if the significant regions of the variants are known.

First, we compute the byte frequency distribution for each byte position of the significant regions. At position $p \in [0 \ldots W-1]$, the maximum likelihood estimation of the frequency $f_p(b)$, $b \in [0 \ldots 255]$, is the number $c(p, b)$ of times that b appears at position p of the significant regions, divided by n:

$$f_p(b) = \frac{c(p, b)}{n}.$$

One problem is that $f_p(b)$ will be zero for those byte values b that never appear at position p of any significant region. Mathematically, we do not want that to happen because it will cause the logarithm in (2) to be undefined. We add a "pseudocount" to the observed byte count $c(p, b)$, and the byte frequency $f_p(b)$ is estimated as

$$f_p(b) = \frac{c(p, b) + d}{n + 256 \cdot d}, \quad (3)$$

where d is a small predefined pseudocount number.

We have established that the PADS signature and the significant regions can lead to each other. We do not know either of them, but we know that the significant regions are those segments that can maximize the matching score with the signature. This "missing data problem" can be solved by

an iterative algorithm, which first makes a guess on the starting positions of the significant regions, computing the signature, using the signature to compute the new starting positions of the significant regions, and repeating the process until convergence.

5 Algorithms for Signature Generation

In the following, we show how to use the expectation-maximization algorithm and the optimized Gibbs Sampling algorithm to compute the PADS signature from a collection of worm variants.

5.1 Expectation-Maximization Algorithm

Expectation-Maximization (EM) [17] is an iterative procedure that obtains the maximum-likelihood parameter estimation. Given a set $\mathcal{S}$ of worm variants, we lack the starting positions $a_1, a_2, \ldots, a_n$ of their significant regions, which are the missing data. The PADS signature Θ is the unknown parameter. To obtain both significant regions and the signature, the EM algorithm iterates between the expectation step and the maximization step after the initialization. The description of the algorithm is given below.

Initialization. The starting positions $a_1, a_2, \ldots, a_n$ of the significant regions for worm variants $S_1, S_2, \ldots, S_n$ are assigned randomly. The maximum likelihood estimate of the signature Θ is calculated based on this initial guess of the significant regions.

Expectation. The new guess on the locations of the significant regions is calculated based on the estimated signature Θ. Specifically, the new starting position a_i of the significant region for S_i is the position that maximizes the matching score with the signature Θ. Namely, we seek

$$a_i = \arg\max_{a_i} \Lambda(\Theta, S_i, a_i) \quad \forall i \in [1..n].$$

Maximization. By (3), the new maximum likelihood estimate of the signature Θ is calculated based on the current guess on the locations of the significant regions.

The algorithm terminates if the average matching score Ω between the worm variants and the signature is within $(1 \pm \varepsilon)$ of the previous iteration, where ε is a small predefined percentage.

Starting with a large signature width W, we run the above algorithm to decide the signature as well as the significant regions. If the minimum matching score of all significant regions deviates greatly from the average score, we repeat the algorithm with a smaller W. This process continues until we reach a signature that matches well with the significant regions of all collected worm variants.

5.2 Gibbs Sampling Algorithm

The main drawback of the EM algorithm is that it may get struck in a local maxima. There is no guarantee that the global maxima can be reached. In order to solve this problem, many strategies have been proposed. One approach is to try out multiple random initial configurations and look for the best result. Another approach is to preprocess the data and choose a "good" initial configuration. In recent years, the simulated annealing [37] approach has attracted great attention. Simply speaking, it allows certain random selection of the parameter (with a small probability moving toward a worse direction), which provides a chance to jump out of a local maxima. One example of simulated annealing is the Gibbs Sampling Algorithm [18], which we will adopt for computing PADS.

The algorithm is initialized by assigning random starting positions for the significant regions of the worm variants. During each iteration, one variant is selected randomly. This selected variant is temporarily excluded from $\mathcal{S}$. The signature is calculated based on the remaining variants. After that, the starting position for the significant region of the selected variant is updated according to a probability distribution based on the matching scores at different positions. The above process repeats until a convergence criterion is met. The description of the algorithm is given below.

Initialization. The starting positions $a_1, a_2, \ldots, a_n$ of the significant regions for worm variants $S_1, S_2, \ldots, S_n$ are assigned randomly.

Predictive Update. One of the n worm variants, S_x, is randomly chosen. The signature Θ is calculated based on the other variants, $\mathcal{S} - S_x$.

Sampling. Every possible position $a_x \in [1 \ldots l_x - W + 1]$ is considered as a candidate for the next starting position of the significant region of S_x. The matching score for each candidate position is $\Lambda(\Theta, S_x, a_x)$ as defined in (1). The next starting position for the significant region of S_x is randomly selected. The probability that a position a_x is chosen is proportional to $\Lambda(\Theta, S_x, a_x)$. That is,

$$\Pr(a_x) = \frac{\Lambda(\Theta, S_x, a_x)}{\sum_{a_x=1}^{l_x - W + 1} \Lambda(\Theta, S_x, a_x)}.$$

Go back to the predictive update step.

The algorithm terminates if the average matching score between the worm variants and the signature is within $(1 \pm \varepsilon)$ of the maximum averge in the previous t iterations, where ε is a small predefined percentage.

6 MPAD with Multiple Signatures

Thus far, the PADS signature is defined as a continuous "string" of byte frequency distributions. It identifies a single significant region in an incoming byte sequence. This strategy has a couple of limitations. First, a worm may include a common segment that appears often in normal traffic. This common segment defeats any attempt by the worm to be polymorphic because the worm is easily identifiable by the segment. However, it can lure our system into choosing the common segment as the PADS signature and, consequently, producing false positives on the normal traffic that happens to carry that segment. Second, a polymorphic worm may have multiple characteristic segments that all carry useful information. PADS captures the most significant one but discards the rest, which renders it less powerful against highly sophisticated polymorphic worms.

To address the above limitations, we propose a natural generalization, called the *multisegment position-aware distribution signature* (MPAD for short), which is a set of PADS signatures for identifying the same worm. It is denoted as $\mathcal{M} = (\Theta_1, \ldots, \Theta_k)$, where Θ_i, $1 \le i \le k$, is a PADS signature. Each PADS signature may have a different width.

To calculate $\mathcal{M}$, we first use the algorithms in Section 5 to compute a PADS signature, Θ_1, and the significant regions for Θ_1. We then remove these significant regions from the worm samples and compute the next PADS signature, Θ_2, and the significant regions for Θ_2. We further remove these significant regions and compute Θ_3 ... until there are no more signatures that can produce good matching scores for all worm samples. When an incoming byte sequence is matched against $\mathcal{M}$, it is classified as a potential worm variant only when its matching scores with all PADS signatures are above

zero. To reduce the matching overhead, the PADS signature with the most diverse distribution can be used first, which attempts to separate worm variants (with some false positives) from the background traffic. The rest of PADS signatures are then applied one after another to progressively filter out the false positives.

7 Mixture of Polymorphic Worms and Clustering Algorithm

Until now, we have only discussed how to calculate a PADS/MPAD signature from a collection of worm variants that belong to the same polymorphic worm. In reality, multiple different polymorphic worms may rage on the Internet at the same time and the captured worm samples may belong to different worms. We have to first partition this mixed set into clusters, each sharing similar traffic patterns and thus likely to come from the same worm. This is called the *cluster partitioning problem*. After partitioning, a PADS/MPAD signature is calculated for each cluster. The signatures can then be used to identify new variants of the worms. We describe two algorithms for the cluster partitioning problem.

7.1 Normalized Cuts

We define a *similarity* metric between any two variants. A naive definition is to first compute the byte-frequency distributions of the two variants and then measure the difference (e.g., KL-divergence) between them. Another naive definition is to count the length of the longest common substring or the combined length of the k longest common substrings. A better definition is to compute a PADS/MPAD signature from the two variants and then take the combined matching score between the variants and the signature. Our experiments will use this definition of similarity. Consider two worm variants, S_i and S_j. Suppose PADS is used. Based on (2), the similarity between S_i and S_j can be expressed as

$$\begin{aligned}\Omega_{ij} =\ & \Omega(\Theta, S_i) + \Omega(\Theta, S_j),\\ =\ & \max_{a_i=1}^{l_i-W+1} \sum_{p=0}^{W-1} \frac{1}{W} \log \frac{f_p(S_i, a_i+p)}{f_0(S_i, a_i+p)}\\ & + \max_{a_j=1}^{l_j-W+1} \sum_{p=0}^{W-1} \frac{1}{W} \log \frac{f_p(S_j, a_j+p)}{f_0(S_j, a_j+p)},\end{aligned} \tag{4}$$

where Θ is the PADS signature calculated from S_i and S_j and W is the length of the signature.

The cluster partitioning problem can be formulated in a graph-theoretic way. We construct a complete graph with n nodes representing the variants $\mathcal{S} = \{S_1, \ldots, S_n\}$. The edge between S_i and S_j is associated with a similarity value of Ω_{ij} as defined in (4). $\Omega_{ii} = 0$. Given the $n \times n$ similarity matrix $\Pi = (\Omega_{ij})$, $i, j \in [1..n]$, we want to find such clusters (e.g., cliques in the graph) that have large similarity values for intracluster edges but small similarity values for intercluster edges. Fig. 1 illustrates a simple example, where a shorter edge means a larger similarity value. This is a well-studied problem and a spectral clustering algorithm called *normalized cuts* can be used to extract the clusters [38], [39]. For purposes of completeness, we briefly describe the algorithm in our context.

The normalized cuts algorithm first decomposes the graph G into two clusters, A and B, that minimize the following criterion:

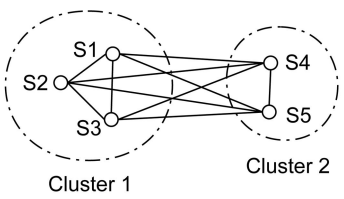

Fig. 1. Clusters.

$$\frac{cut(A,B)}{assoc(A,G)} + \frac{cut(A,B)}{assoc(B,G)},$$

where $cut(A,B)$ is the sum of the similarity values of all edges that have one end in A and the other end in B, $assoc(A,G)$ is the sum of the similarity values of all edges that have one end in A and the other end unrestricted, and $assoc(B,G)$ is similarly defined.

A vector y is used to define the two clusters. If the ith value of y is 1, then S_i belongs to the first cluster. If it is -1, then S_i belongs to the second cluster. In addition to the similarity matrix Π, we define a *degree matrix* D as follows:

$$D_{ii} = \sum_j \Omega_{ij}$$

for the diagonal elements and zero for all off-diagonal elements.

The criterion can then be rewritten as

$$\frac{y^T(D-\Pi)y}{y^T D y}.$$

Minimizing the above criterion is an integer programming problem if y only takes discrete elements. An approximation is to treat y as a real vector [38] with positive elements for the first cluster and negative elements for the second cluster. It can be shown that any y satisfying the following equation for some λ value will minimize the criterion:

$$(D-\Pi)y = \lambda D y.$$

Following certain transformations that we omit here, the generalized eigenvector y correponding to the second smallest eigenvalue is used [38]. Readers are referred to [38] for details.

After the algorithm partitions the graph into two clusters, we can recursively apply the algorithm to further partition each cluster until there is no significant difference between average intracluster similarity and average intercluster similarity.

7.2 Expectation-Maximization (EM)

The Expectation-Maximization algorithm [17] can also be adapted to solve the cluster partitioning problem. Starting from $k = 2$, we divide the set of worm variants randomly into k clusters, compute a signature for each cluster, examine each variant and move it to the cluster whose signature matches it the best, recompute the signatures, examine each variant and move it to the cluster whose signature matches it the best, ..., until no variant is moved between clusters. We repeat the above computation with an increased number (k) of clusters until at least one cluster is empty or the signatures of two clusters are very close to each other.

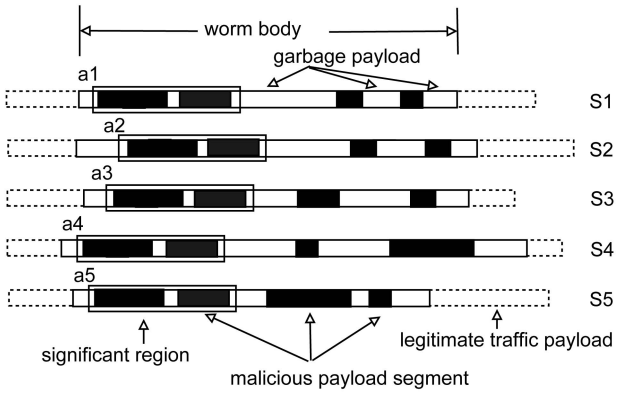

Fig. 2. Variants of a polymorphic worm.

8 Experiments

We perform experiments to demonstrate the effectiveness of the proposed signatures in identifying polymorphic worms. The malicious payloads of the MS Blaster worm, W32/Sasser worm, Sapphire worm, and Peer-to-peer UDP Distributed Denial of Service (PUD) worm are used in the experiments. The MS Blaster worm exploits a vulnerability in Microsoft's DCOM RPC interface. Upon successful execution, the MS Blaster worm retrieves a copy of the file msblast.exe from a previously infected host [40]. The W32/Sasser worm exploits a buffer overflow vulnerability in the Windows Local Security Authority Service Server (LSASS) on TCP port 445. This vulnerability allows a remote attacker to execute arbitrary code with system privileges [41]. The Sapphire (also called Slammer) worm caused considerable harm simply by overloading networks and taking database servers out of operation. Many individual sites lost connectivity as their access bandwidth was saturated by the activities of the worm [42]. The PUD worm tries to exploit the SSL vulnerability on i386 Linux machines [43].

In the experiments, we artificially generate the variants of these worms based on some polymorphism techniques discussed in Section 3, including instruction substitution, garbage-code insertion, and normal-traffic embedding. For normal-traffic streams, we use traces taken from the UF CISE network.

Fig. 2 illustrates the polymorphic worm design with five variants, S1, S2, ..., and S5, whose significant regions start at a1, a2, ..., and a5, respectively. Each variant consists of three different types of regions. The black regions are segments of the malicious payload in the worm. Substitution is performed on 20 percent of the malicious payload. Garbage code, which is represented as the white regions with solid lines, is inserted at random locations in the malicious payload. The default ratio of the malicious payload to the garbage code is 1:2. In addition to garbage code, each variant is embedded in the legitimate traffic of a normal session, represented by the white regions with dotted lines. The default length of the normal traffic carried by a worm variant is between 2 KB to 20 KB. The malicious payload is embedded at a random location in the normal traffic carried by the worm variant. The worm variants in our experiments have different lengths because each variant carries a different amount of normal traffic.

The experiments are performed on a low-end DELL PC with 2.39 GHz Pentium 4 CPU and 512 MB RAM. For all experiments except that, in Section 8.7, one PADS signature is generated for each worm. For the experiment in Section 8.7, multiple PADS signatures (together called MPAD) are generated for each worm. The default length of a PADS signature is 10 bytes.

8.1 Convergence of Signature Generation Algorithms

In the first experiment, 100 variants of MS Blaster worm are generated and they are used as worm samples for signature generation. The EM algorithm and the Gibbs Sampling algorithm each run three times with different initial configurations. Specifically, the initial starting points of significant regions are randomly selected for each run. Fig. 3a shows that the quality of the PADS signature obtained by EM or Gibbs is improved over time. Recall that the execution of either algorithm consists of iteration cycles (Expectation/Maximization steps for EM and Update/Sampling steps for Gibbs). The y axis is the average matching score between the 100 samples and the signature obtained so far. From the figure, the best matching score is around 8.7, which is likely to be the global maxima. EM stabilizes the signature quicker; the three runs stabilize after 7.6, 5.0, and 5.0 seconds of execution, respectively. However, EM tends to settle down at a local maxima, depending on

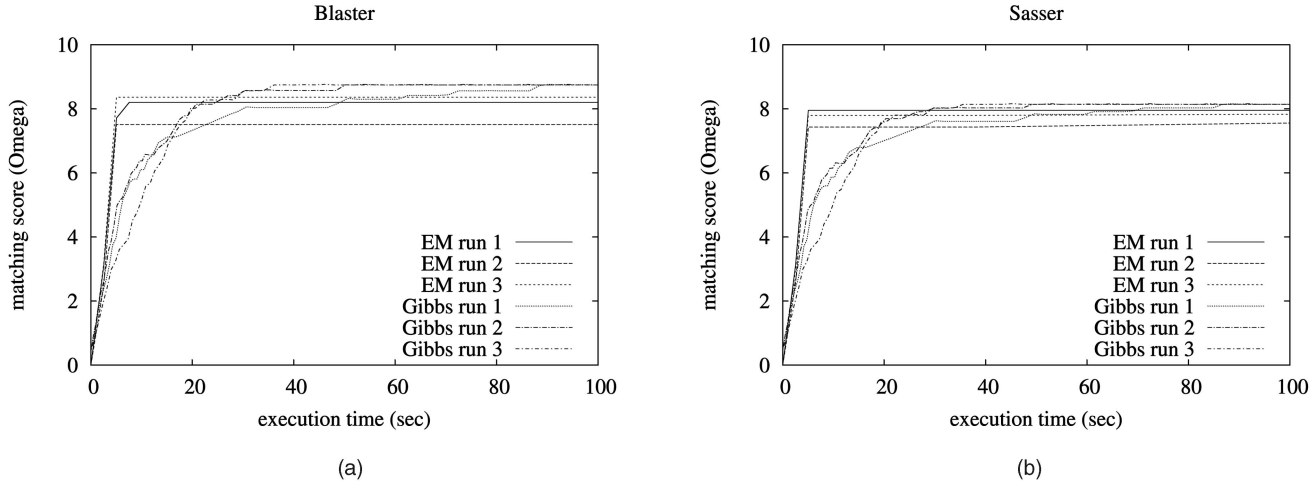

Fig. 3. Convergence of EM and Gibbs.

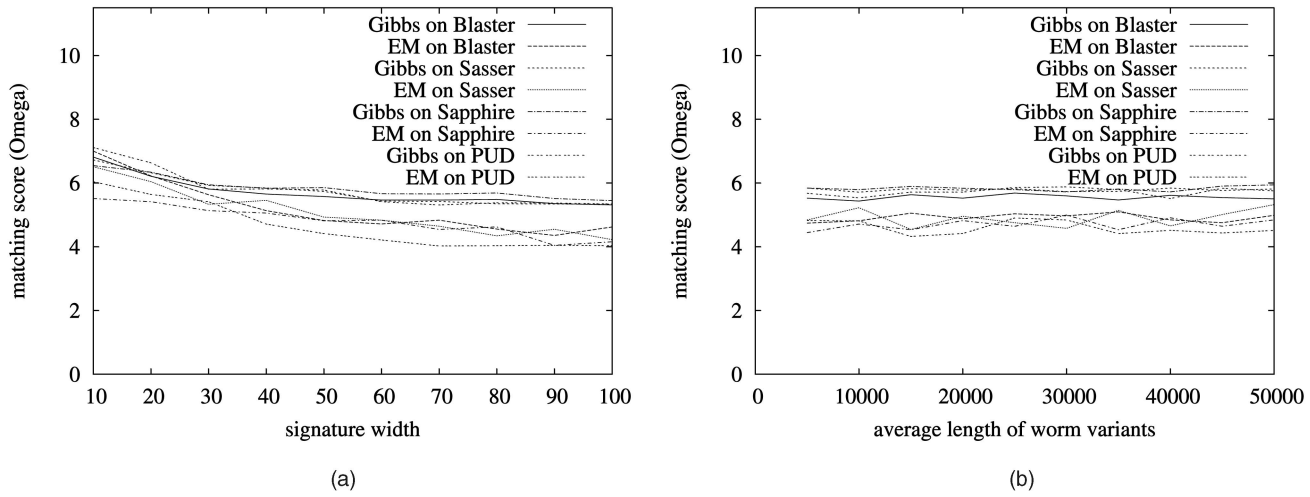

Fig. 4. Average matching score between 100 sample variants and the signature produced by EM or Gibbs based on these samples. (a) The score with respect to signature width. (b) The score with respect to worm length.

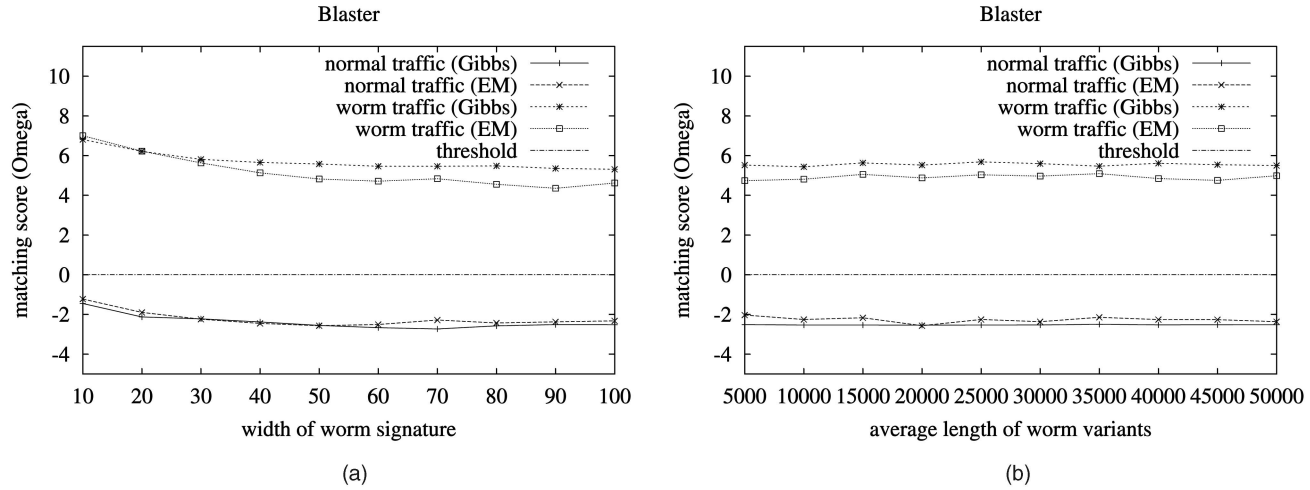

Fig. 5. The average matching score between test variants of *Blaster* and the signature produced by EM or Gibbs is above the threshold (zero), while the average matching score between normal traffic and the signature produced by EM or Gibbs is below the threshold. (a) The score with respect to signature width. (b) The score with respect to worm length.

the initial configuration. Gibbs takes more time to stabilize the signature; the three runs stabilize after 88.4, 50.2, and 36.1 seconds, respectively. But the quality of the signature is already comparable to that of EM after 20 seconds of execution. Gibbs tends to find the global maxima due to its ability of jumping out of a local maxima.

We repeat the experiment for the Sasser worm and obtain similar results, which are shown in Fig. 3b.

8.2 Impact of Signature Width and Worm Length

In the next set of experiments, we use 100 sample variants from each of the four worms for signature generation. We then use 10,000 test variants from each of the four worms, mixed with 10,000 normal-traffic streams, to test the quality of the signature.

Fig. 4 shows the average matching score between the 100 sample variants and the signature produced by EM or Gibbs based on these samples. Fig. 4a shows the average score when signatures with different widths are used. Fig. 4b shows the average score with respect to the average length of the worm variants. Because the worm code has a fixed length, we change the length of a variant by letting it carry a variable amount of normal traffic.

Figs. 5, 6, 7, and 8 show the average matching scores between the test worm variants (as well as normal-traffic streams) and the signatures of the four worms, respectively. The average scores for the test worm variants are always above zero and the average scores for normal-traffic streams are always below zero. Therefore, with a threshold of zero, worm variants can be separated from normal traffic. In our experiments, false positives and false negatives are extremely rare, which will be discussed in the next section.

We have two additional observations from the above figures. 1) Increasing the signature width will decrease the average matching score for worm variants and normal traffic. This is reasonable because a longer signature is harder to match precisely by the byte sequences of worm variants or normal traffic. 2) Increasing the length of normal traffic carried by a worm variant, which has been widely used by some polymorphic worms to elude the anomaly-based systems, provides no help in avoiding detection by our PADS approach. The reason is that our approach identifies a significant region and only uses the significant region for

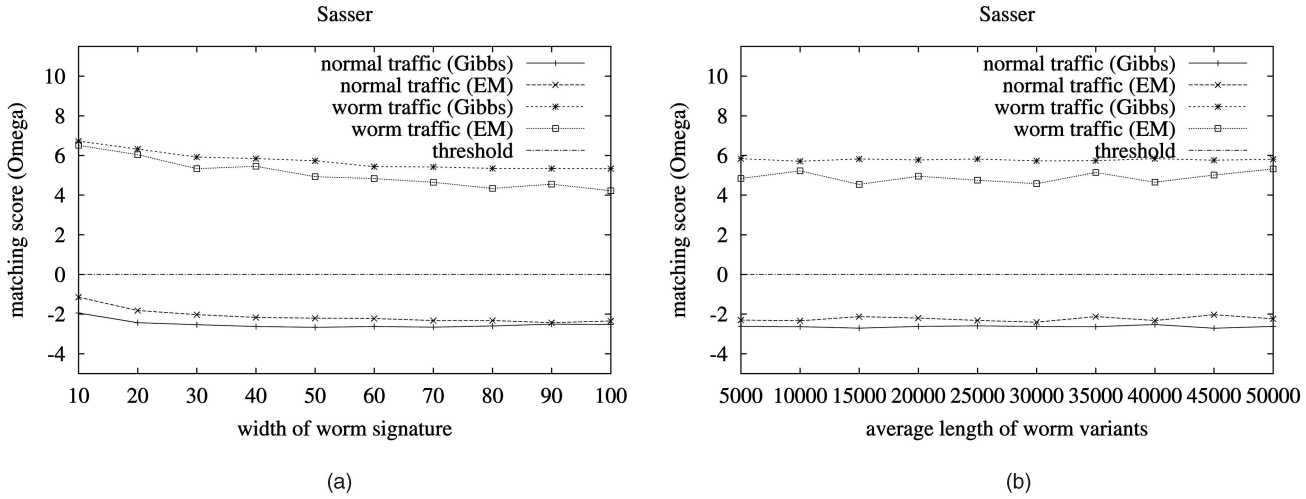

Fig. 6. The average matching score between test variants of *Sasser* and the signature produced by EM or Gibbs is above the threshold (zero), while the average matching score between normal traffic and the signature produced by EM or Gibbs is below the threshold. (a) The score with respect to signature width. (b) The score with respect to worm length.

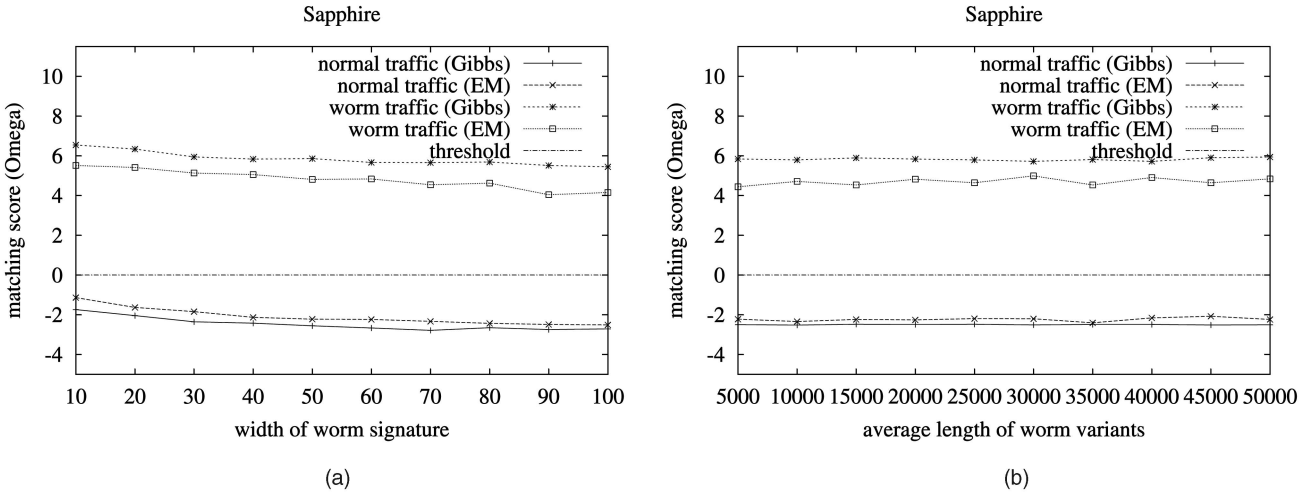

Fig. 7. The average matching score between test variants of *Sapphire* and the signature produced by EM or Gibbs is above the threshold (zero), while the average matching score between normal traffic and the signature produced by EM or Gibbs is below the threshold. (a) The score with respect to signature width. (b) The score with respect to worm length.

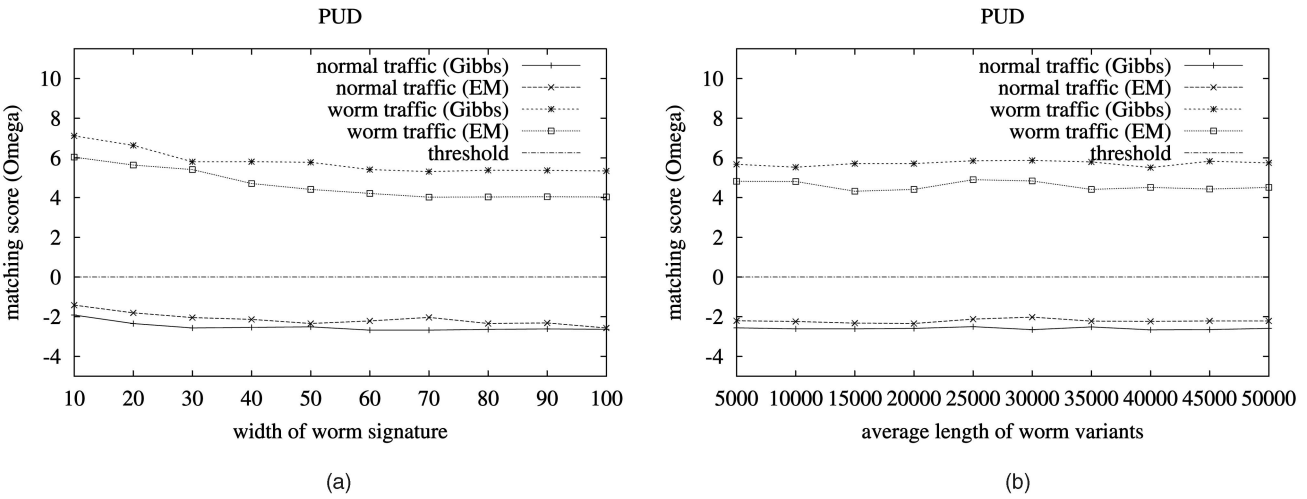

Fig. 8. The average matching score between test variants of *PUD* and the signature produced by EM or Gibbs is above the threshold (zero), while the average matching score between normal traffic and the signature produced by EM or Gibbs is below the threshold. (a) The score with respect to signature width. (b) The score with respect to worm length.

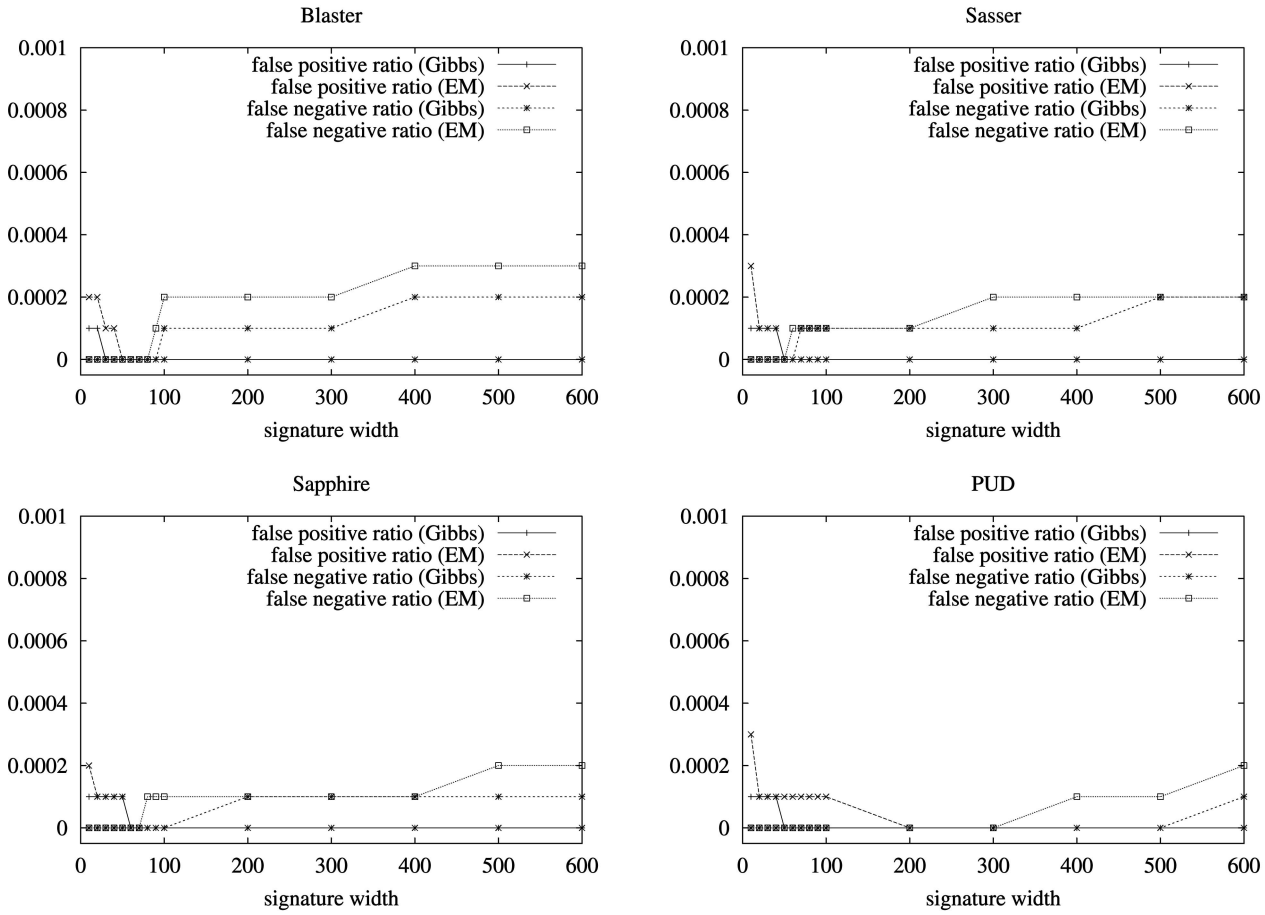

Fig. 9. Both false positive ratios and false negative ratios are very small for all four worms.

signature generation or matching. The normal traffic carried by a worm variant, no matter how much it is, will not be used for signature generation or matching.

8.3 False Positives and False Negatives

Fig. 9 shows the false positive ratio and false negative ratio of using the PADS signature to detect variants of the four worms. The *false positive ratio* is defined as the number of normal-traffic streams misclassified as worm variants divided by the total number of normal-traffic streams used in the experiment. The *false negative ratio* is defined as the number of worm variants misclassified as normal traffic divided by the total number of worm variants used in the experiment. For all four worms, neither the false positive ratio nor the false negative ratio exceeds 0.0003. As we can see from the figure, the signature produced by Gibbs works better than the signature by EM for all four worms. When the signature width increases, the false positive ratio decreases gradually while the false negative ratio increases gradually. From Figs. 5, 6, 7, and 8, we know that a longer signature will reduce the average matching score for both worm traffic and normal traffic. It moves the average score for worm traffic closer to the threshold but moves the average score for normal traffic further away from the threshold. Consequently, false negatives are more likely to happen, but false positives are less likely to happen.

8.4 Matching Time

Our next experiment measures the amount of time it takes to scan a certain amount of data for a signature match, called *matching time*. A signature match is announced if the matching score is above zero. How to calculate the matching score between a byte sequence and a PADS signature is described in Section 4.2. Fig. 10 shows the matching time with respect to the amount of data that is scanned. For example, when the signature width is 10 bytes, the time it takes to scan 10 MB data is 253.1 ms on our low-end Dell computer. It is equivalent to 316.1 Mbps, much higher than the speed of the fast Ethernet. The performance can certainly be improved by implementing PADS at high-end firewalls with hardware support. Today's high-end firewalls (PIX and Netscreen) have reached 30 Gbps with far more expensive functions such as encryption. If a firewall cannot keep up with the matching load because there are too many PADS signatures, it can choose to work only with the most recent signatures. At the least, PADS provides a defense option for dealing with new worm variants that have not been fully analyzed and cannot yet be mitigated with faster means. For older worms, faster defense methods are likely to exist.

8.5 Comparing PADS with Existing Methods

For the purpose of comparison, we also perform experiments with some existing methods. Fig. 11 shows the experimental results based on the *longest common substring* method [27], which first identifies the longest common substring among the sample worm variants and then uses the substring as a signature to match against the test variants. Based on Fig. 11a, as the number of sample variants increases, the length of the longest common substring decreases. A shorter signature

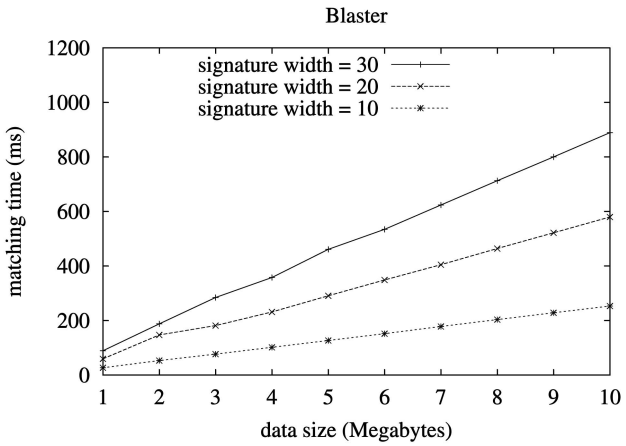

Fig. 10. The time it takes to scan a certain amount of data for signature match. For example, when the signature width is 10 bytes, the time it takes to scan 10 MB of data is 253.1 ms on our low-end Dell computer. It is equivalent to 316.1 Mbps, much higher than the speed of the fast Ethernet.

increases the chance for it to appear in normal traffic. Consequently, as the false negative ratio decreases, the false positive ratio increases dramatically (Fig. 11b). On the contrary, without the requirement of exact matching, a PADS signature is able to retain many more (statistical) characteristics of a polymorphic worm.

Now, consider the *position-unaware* byte frequency distributions that are used in some current systems. Fig. 12a shows the position-unaware byte frequency distribution of 100 normal-traffic streams and Fig. 12b shows the byte frequency distribution of the malicious payload of the MS blaster worm. These two distributions are very different, which seems to provide a way to detect the worm. However, if we create a worm variant by embedding the malicious payload in normal traffic, the combined byte frequency distribution can be made very similar to that of normal traffic. Fig. 13 shows the byte frequency distributions of two worm variants whose normal traffic payloads are 1 and 9 times the malicious payload, respectively. Fig. 13b is very similar to Fig. 12a. Therefore, using byte frequency distributions alone cannot handle worm variants. The proposed *position-aware* distribution signature works better against polymorphic worms.

Not surprisingly, the matching times of the above two simple approaches are much smaller than that of PADS. In our experiments, the time it takes to scan 10 MB data is 253.1 ms for PADS of 10 bytes, 26.4 ms for the longest common substring signature (generated from 100 sample variants), and 25.2 ms for the position-unaware byte-frequency distribution signature.

8.6 Effectiveness of Normalized Cuts Algorithm

The next experiment is to evaluate the effectiveness of the normalized cuts algorithm in solving the cluster partitioning problem. In this experiment, 50 variants of the MS Blaster worm with ids $[1 \ldots 50]$ and 50 variants of the W32/Sasser worm with ids $[51 \ldots 100]$ are generated. The normalized cuts algorithm is used to separate the mixed 100 variants into clusters. The similarity matrix, as defined in Section 7.1 and particularly by (4), is calculated by using the Gibbs Sampling algorithm. The result is shown in Fig. 14a, where the horizontal axes are variant ids, representing the rows i and the columns j of the matrix, and the vertical axis is the similarity value between variants i and j. The surface of the plot can be roughly partitioned into three regions. The first region ($i, j \in [1 \ldots 50]$) shows the similarity values among the set of MS Blaster worm variants. The second region ($i, j \in [51 \ldots 100]$) shows the similarity values among the set of W32/Sasser worm variants. The remaining region shows the similarity values between MS Blaster variants and W32/Sasser variants. By using the normalized cuts algorithm, the 100 worm variants are separated into two clusters, one for MS Blaster and one for W32/Sasser. The resulting y vector is shown in Fig. 14b, where each point represents one element in y. The variants whose values in y are below zero belong to one cluster. The variants whose values in y are above zero belong to the other cluster.

8.7 Effectiveness of MPAD

Our final experiment verifies the effectiveness of MPAD. We add a common string of 50 bytes to all test worm variants and all normal-traffic streams. As shown in Fig. 15, if a single PADS signature is used, the false positive ratio will be 100 percent in this scenario because the signature will simply be the common string. However, as shown in

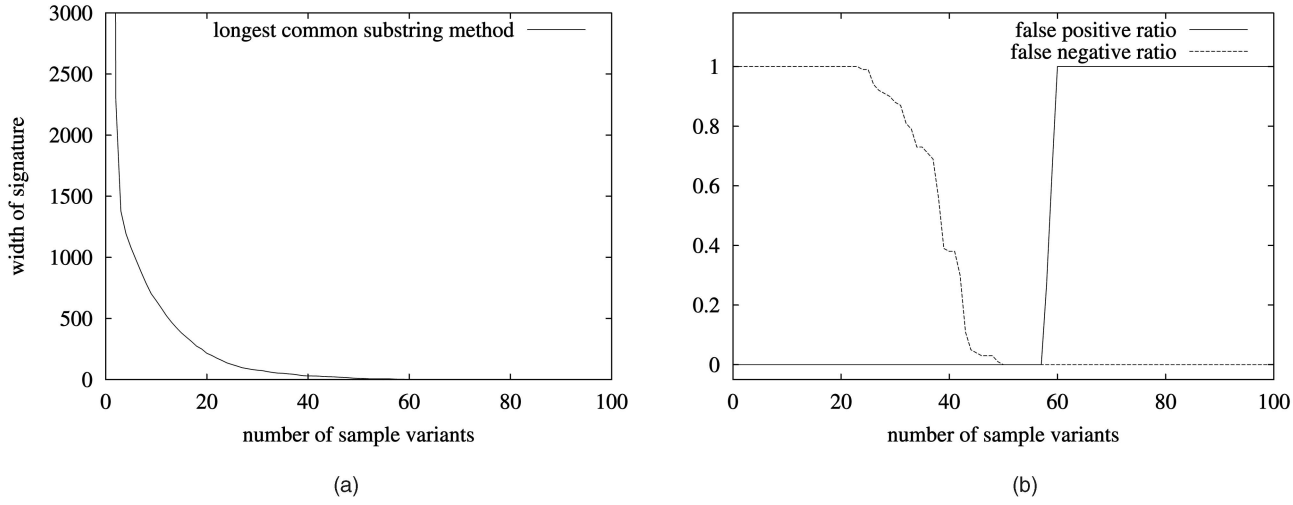

Fig. 11. Performance of a signature-based system using longest common substrings.

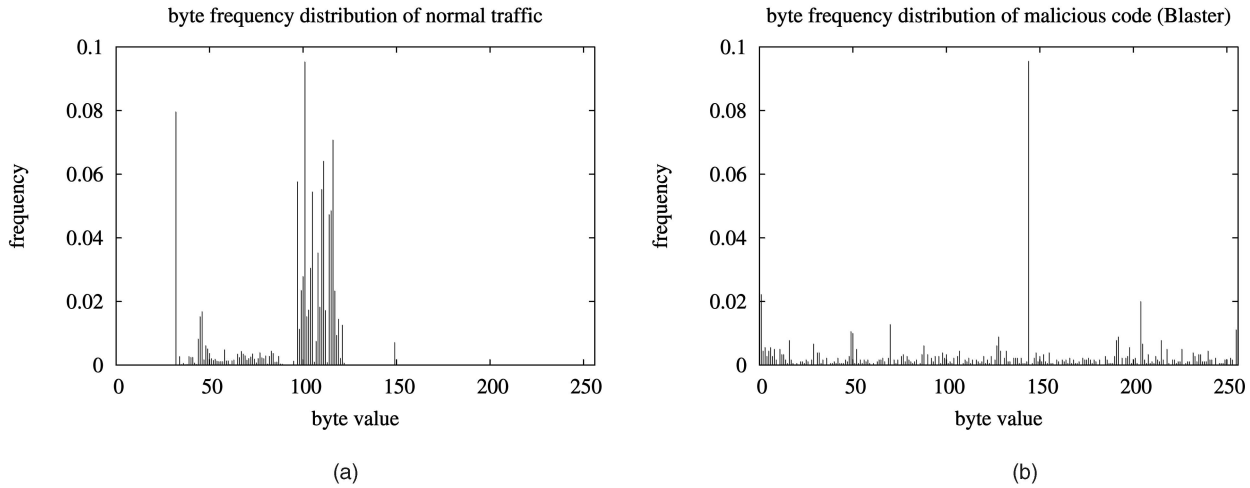

Fig. 12. Byte frequency distributions of (a) normal traffic and (b) worm traffic.

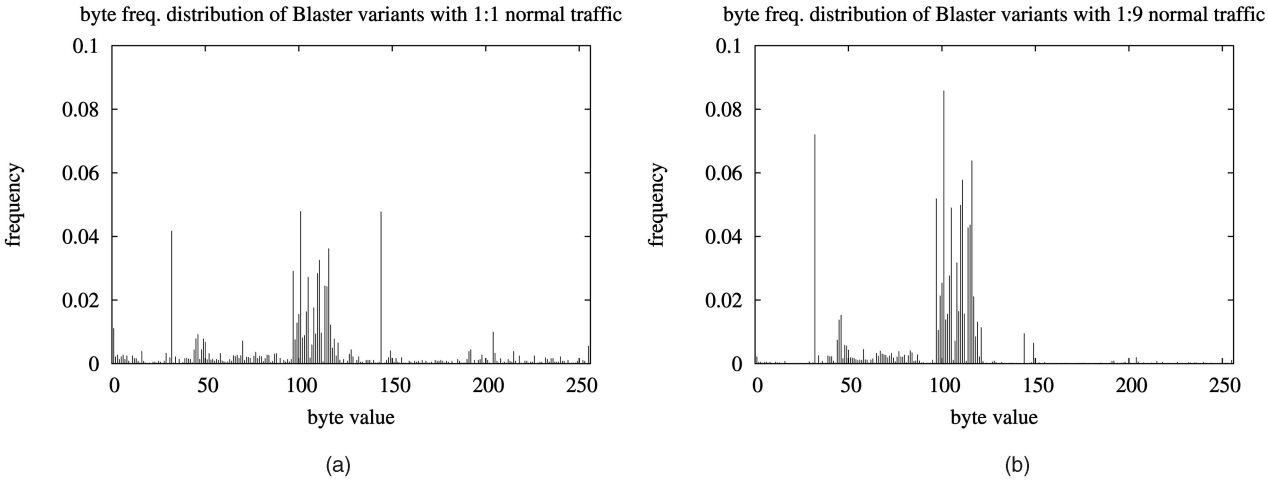

Fig. 13. Byte frequency distributions of worm variants. (a) The malicious and normal payloads carried by a worm variant have equal length. (b) The normal payload carried by a worm variant is 9 times the malicious payload.

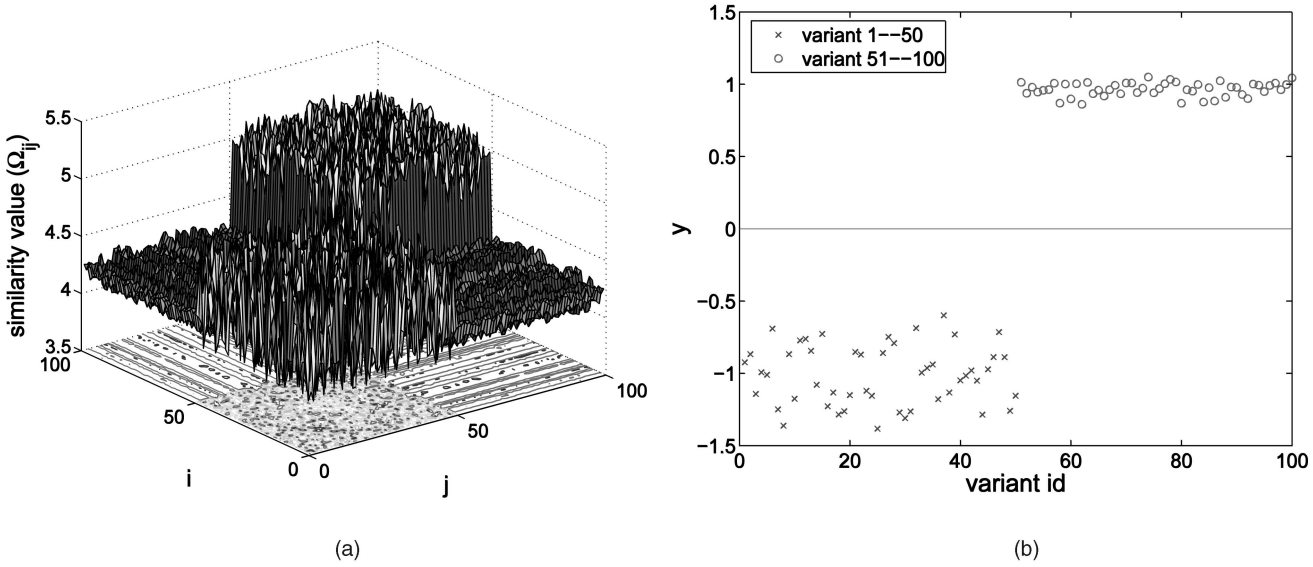

Fig. 14. Clustering variants using normalized cuts.

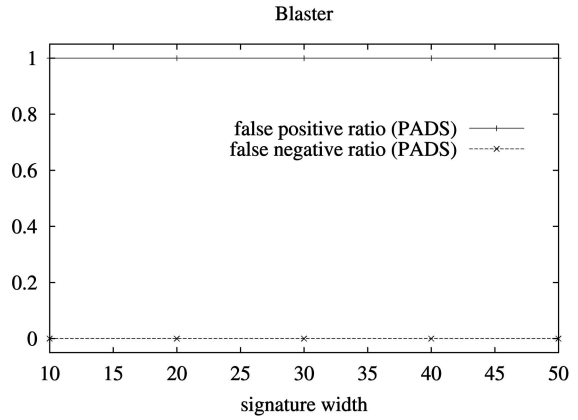

Fig. 15. If the normal-traffic streams used in the experiment all carry a common string and the worm variants also carry the same string, the false positive ratio will be 100 percent.

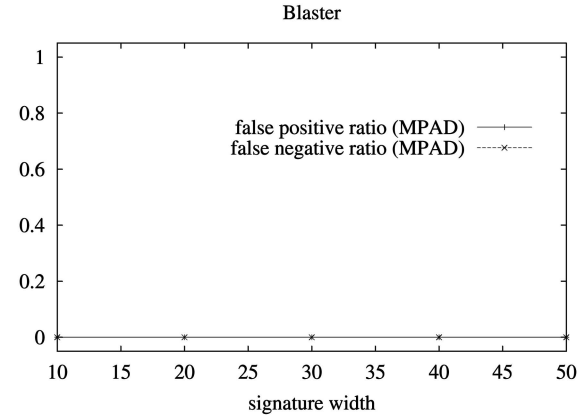

Fig. 16. By using multiple signatures, MPAD is able to drive the false positive ratio to zero.

Fig. 16, if the MPAD signature (Section 6) is used, the false positive ratio is driven to zero due to the use of multiple PADS signatures. In practice, the PADS signatures that frequently match normal traffic should be removed from the MPAD signature. When this is done in the above experiment, the MPAD signature will contain only one PADS signature; those PADS signatures resulted from the common string are removed.

9 Conclusion

In this paper, we make the following contributions: First, we analyze worm polymorphism and introduce a new position-aware distribution signature against polymorphic worms. We propose iterative algorithms to calculate the signature from captured worm samples. Second, the signature is generalized to overcome some limitations. Third, algorithms are given to solve the cluster partitioning problem when samples from different worms are mixed together. Fourth, extensively experiments are performed on four worms to validate the proposed signature and its algorithms.

References

[1] S. Staniford, V. Paxson, and N. Weaver, "How to Own the Internet in Your Spare Time," *Proc. 11th USENIX Security Symp. (Security '02),* Aug. 2002.

[2] D. Moore, C. Shannon, G.M. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," *Proc. 22nd Ann. Joint Conf. IEEE Computer and Comm. Socs. (INFOCOM '03),* pp. 1901-1910, Apr. 2003.

[3] S. Chen and Y. Tang, "Slowing Down Internet Worms," *Proc. 24th IEEE Int'l Conf. Distributed Computing and Systems (ICDCS '04),* Mar. 2004.

[4] C. Kruegel and G. Vigna, "Anomaly Detection of Web-Based Attacks," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '03),* pp. 251-261, Oct. 2003.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Security and Privacy,* pp. 33-39, July 2003.

[6] C. Cowan, C. Pu, D. Maier, J. Walpole, P. Bakke, S. Beattie, A. Grier, P. Wagle, Q. Zhang, and H. Hinton, "StackGuard: Automatic Adaptive Detection and Prevention of Buffer-Overflow Attacks," *Proc. Seventh USENIX Security Conf. (Security '98),* pp. 63-78, Jan. 1998.

[7] M. Eichin and J. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," *Proc. IEEE Symp. Security and Privacy,* pp. 326-344, May 1989.

[8] C.C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," *Proc. Ninth ACM Conf. Computer and Comm. Security (CCS '02),* pp. 138-147, Nov. 2002.

[9] C.C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and Early Warning for Internet Worms," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '2003),* pp. 190-199, Oct. 2003.

[10] Z. Chen, L. Gao, and K. Kwiat, "Modeling the Spread of Active Worms," *Proc. 22nd Ann. Joint Conf. IEEE Computer and Comm. Socs. (INFOCOM '03),* pp. 1890-1900, Mar. 2003.

[11] D. Moore, C. Shannon, and K. Claffy, "Code-Red: A Case Study on the Spread and Victims of an Internet Worm," *Proc. Second Internet Measurement Workshop (IMW '02),* pp. 273-284, Nov. 2002.

[12] J.O. Kephart and S.R. White, "Directed-Graph Epidemiological Models of Computer Viruses," *Proc. 1991 IEEE Symp. Security and Privacy,* pp. 343-361, May 1991.

[13] H. Javitz and A. Valdes, "The NIDES Statistical Component Description and Justification," technical report, Computer Science Laboratory, SRI Int'l, 1994.

[14] K. Wang and S.J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," *Proc. Seventh Int'l Symp. Recent Advances in Intrusion Detection (RAID '04),* Sept. 2004.

[15] K. Ilgun, R. Kemmerer, and P. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach," *IEEE Trans. Software Eng.,* vol. 21, pp. 181-199, 1995.

[16] U. Lindqvist and P. Porras, "Detecting Computer and Network Misuse through the Production-Based Expert System Toolset (P-BEST)," *Proc. IEEE Symp. Security and Privacy,* May 1999.

[17] C.E. Lawrence and A.A. Reilly, "An Expectation Maximization (EM) Algorithm for the Identification and Characterization of Common Sites in Unaligned Biopolymer Sequences," *Proteins: Structure, Function and Genetics,* vol. 7, pp. 41-51, 1990.

[18] C.E. Lawrence, S.F. Altschul, M.S. Boguski, J.S. Liu, A.F. Neuwald, and J.C. Wootton, "Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment," *Science,* vol. 262, pp. 208-214, Oct. 1993.

[19] N. Weaver, I. Hamadeh, G. Kesidis, and V. Paxson, "Preliminary Results Using Scale-Down to Explore Worm Dynamics," *Proc. ACM Workshop Rapid Malcode (WORM '04),* pp. 65-72, 2004.

[20] M.M. Williamson, "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code," *Proc. Ann. Computer Security Applications Conf. (ACSAC '02),* pp. 61-68, Oct. 2003.

[21] J. Twycross and M.M. Williamson, "Implementing and Testing a Virus Throttle," *Proc. 12th USENIX Security Symp. (Security '03),* pp. 285-294, Aug. 2003.

[22] S. Schechter, J. Jung, and A.W. Berger, "Fast Detection of Scanning Worm Infections," *Proc. Seventh Int'l Symp. Recent Advances in Intrusion Detection (RAID '04),* Sept. 2004.

[23] N. Weaver, S. Staniford, and V. Paxson, "Very Fast Containment of Scanning Worms," *Proc. 13th USENIX Security Symp. (Security '04),* pp. 29-44, Aug. 2004.

[24] G. Gu, D. Dagon, X. Qin, M.I. Sharif, W. Lee, and G.F. Riley, "Worm Detection, Early Warning, and Response Based on Local Victim Information," *Proc. 20th Ann. Computer Security Applications Conf. (ACSAC '2004),* pp. 136-145, Dec. 2004.

[25] L. Spitzner, *Honeypots: Tracking Hackers.* Addison-Wesley, 2002.

[26] N. Provos, "A Virtual Honeypot Framework," *Proc. 13th USENIX Security Symp. (Security '04),* pp. 1-14, Aug. 2004.

[27] C. Kreibich and J. Crowcroft, "Honeycomb: Creating Intrusion Detection Signatures Using Honeypots," *Proc. Second Workshop Hot Topics in Networks (HotNets-II),* Nov. 2003.

[28] D. Dagon, X. Qin, G. Gu, W. Lee, J. Grizzard, J. Levin, and H. Owen, "HoneyStat: Local Worm Detection Using Honeypots," *Proc. Seventh Int'l Symp. Recent Advances in Intrusion Detection (RAID '04),* Sept. 2004.

[29] H.-A. Kim and B. Karp, "Autograph: Toward Automated, Distributed Worm Signature Detection," *Proc. 13th USENIX Security Symp. (Security '04),* pp. 271-286, Aug. 2004.

[30] S. Singh, C. Estan, G. Varghese, and S. Savage, "The EarlyBird System for Real-Time Detection of Unknown Worms," *Proc. Sixth Symp. Operating System Design and Implementation (OSDI '04),* Dec. 2004.

[31] M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," *Proc. 12th USENIX Security Symp. (Security '2003),* pp. 169-186, Aug. 2003.

[32] O. Kolesnikov and W. Lee, "Advanced Polymorphic Worms: Evading IDS by Blending in with Normal Traffic," technical report, College of Computing, Georgia Inst. Technology, 2004.

[33] J. Newsome and D. Song, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software," *Proc. 12th Ann. Network and Distributed System Security Symp. (NDSS '05),* Feb. 2005.

[34] J. Newsome, B. Karp, and D. Song, "Polygraph: Automatically Generating Signatures for Polymorphic Worms," *Proc. 2005 IEEE Symp. Security and Privacy,* May 2005.

[35] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World.* Prentice Hall, 2002.

[36] Y. Tang and S. Chen, "Defending against Internet Worms: A Signature-Based Approach," *Proc. IEEE INFOCOM,* May 2005.

[37] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 6, pp. 721-741, 1984.

[38] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.

[39] D.A. Forsyth and J. Ponce, *Computer Vision A Modern Approach,* Prentice Hall, 2003.

[40] "CERT Advisory CA-2003-20: W32/Blaster Worm," Computer Emergency Response Team, http://www.cert.org/advisories/CA-2003-20.html, 2003.

[41] "US-CERT Cyber Security Bulletin SB04-133," US Computer Emergency Readiness Team, http://www.us-cert.gov/cas/body/bulletins/SB04-133.pdf, 2004.

[42] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "The Spread of the Sapphire/Slammer Worm," http://www.caida.org/outreach/papers/2003/sapphire/, 2003.

[43] C. Hoepers and K. Steding-Jessen, "The Scan of the Month 25," http://www.honeynet.org/scans/scan25/writeup.html, 2003.

[44] J.A. Rochlis and M.W. Eichin, "With Microscope and Tweezers: The Worm from MIT's Perspective," *Comm. ACM,* vol. 32, no. 6, pp. 689-698, 1989.

[45] C.C. Zou, W. Gong, and D. Towsley, "Worm Propagation Modeling and Analysis under Dynamic Quarantine Defence," *Proc. ACM Workshop Rapid Malcode (WORM '03),* Aug. 2003.

[46] N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, "A Taxonomy of Computer Worms," *Proc. ACM Workshop Rapid Malcode (WORM '2003),* Aug. 2003.

[47] R. Sommer and V. Paxson, "Enhancing Byte-Level Network Intrusion Detection Signatures with Context," *Proc. 10th ACM Conf. Computer and Comm. Security (CCS '2003),* pp. 262-271, Oct. 2003.

[48] "CERT Advisory CA-2001-23: 'Code Red' Worm Exploiting Buffer Overflow in IIS Indexing Service DLL," Computer Emergency Response Team, http://www.cert.org/advisories/CA-2001-23.html, 2001.

[49] "CERT Advisory CA-2001-26: Nimda Worm," Computer Emergency Response Team, http://www.cert.org/advisories/CA-2001-26.html, 2001.

[50] "CERT Advisory CA-2001-26: MS-SQL Server Worm," Computer Emergency Response Team, http://www.cert.org/advisories/CA-2003-04.html. 2003.

[51] M. Liljenstam, Y. Yuan, B. Premore, and D. Nicol, "A Mixed Abstraction Level Simulation Model of Large-Scale Internet Worm Infestations," *Proc. 10th IEEE Int' Symp. Modeling, Analysis, and Simulation of Computer and Telecomm. Systems (MASCOTS '02),* pp. 109-116, Oct. 2002.

[52] H.W. Hethcote, "The Mathematics of Infectious Diseases," *SIAM Rev.,* vol. 42, no. 4, pp. 599-653, 2000.

[53] F. Buchholz, T.E. Daniels, J.P. Early, R. Gopalakrishna, R.P. Gorman, B.A. Kuperman, S. Nystrom, A. Schroll, and A. Smith, "Digging for Worms, Fishing for Answers," *Proc. 18th Ann. Computer Security Applications Conf. (ACSAC '02),* pp. 219-226, Dec. 2002.

[54] J. Wu, S. Vangala, L. Gao, and K. Kwiat, "An Effective Architecture and Algorithm for Detecting Worms with Various Scan Techniques," *Proc. 11th Ann. Network and Distributed System Security Symp. (NDSS '04),* Feb. 2004.

[55] F. Castaneda, E.C. Sezer, and J. Xu, "WORM vs. WORM: Preliminary Study of an Active Counter-Attack Mechanism," *Proc. ACM Workshop Rapid Malcode (WORM '04),* pp. 83-93, 2004.

[56] D.R. Ellis, J.G. Aiken, K.S. Attwood, and S.D. Tenaglia, "A Behavioral Approach to Worm Detection," *Proc. '04 ACM Workshop on Rapid Malcode (WORM '04),* pp. 43-53, 2004.

[57] S. Staniford, D. Moore, V. Paxson, and N. Weaver, "The Top Speed of Flash Worms," *Proc. ACM Workshop Rapid Malcode (WORM '04),* pp. 33-42, 2004.

[58] H.J. Wang, C. Guo, D.R. Simon, and A. Zugenmaier, "Shield: Vulnerability-Driven Network Filters for Preventing Known Vulnerability Exploits," *Proc. Conf. Applications, Technologies, Architectures, and Protocols for Computer Comm. (SIGCOMM '04),* pp. 193-204, 2004.

[59] H. Feng, A. Kamra, V. Misra, and A. Keromytis, "The Effect of DNS Delays on Worm Propagation in an IPv6 Internet," *Proc. 24th Ann. Joint Conf. IEEE Computer and Comm. Socs. (INFOCOM '05),* Mar. 2005.

Yong Tang received the BS degree from Peking University, Beijing, China, in 1999. He received the PhD degree in computer science from the University of Florida in 2006. He is currently with Sunbelt Software. His primary research field is network security, especially on mitigating distributed denial-of-service attacks and Internet worm attacks.

Shigang Chen received the BS degree in computer science from the University of Science and Technology of China in 1993. He received the MS and PhD degrees in computer science from the University of Illinois at Urbana-Champaign in 1996 and 1999, respectively. After graduation, he worked at Cisco Systems for three years before joining the University of Florida as an assistant professor in the Department of Computer and Information Science and Engineering. His research interests include wireless networks, Internet security, and overlay networks. He received the IEEE Communications Society Best Tutorial Paper Award in 1999. He was a guest editor for the *ACM/Baltzer Journal of Wireless Networks (WINET)* and the *IEEE Transactions on Vehicle Technologies*. He served as a TPC cochair for the Computer and Network Security Symposium of the 2006 IEEE International Workshop on Cluster Computing, a vice TPC chair for the 2005 IEEE International Conference on Mobile Ad Hoc and Sensor Systems, a vice general chair for QShine 2005, a TPC cochair for the 2004 International Conference on Quality of Service in Heterogeneous Wired/Wireless Networks (QShine), and a TPC member for many conferences.