# Defending Against Internet Worms: A Signature-Based Approach

Yong Tang Shigang Chen
Department of Computer & Information Science & Engineering
University of Florida, Gainesville, FL 32611-6120, USA
{yt1, sgchen}@cise.ufl.edu

*Abstract*— **With the capability of infecting hundreds of thousands of hosts, worms represent a major threat to the Internet. The defense against Internet worms is largely an open problem. This paper investigates two important problems. Can a localized defense system detect new worms that were not seen before and, moreover, capture the attack packets? How to identify polymorphic worms from the normal background traffic? We have two major contributions here. The first contribution is the design of a novel double-honeypot system, which is able to automatically detect new worms and isolate the attack traffic. The second contribution is the proposal of a new type of position-aware distribution signatures (PADS), which fit in the gap between the traditional signatures and the anomaly-based systems. We propose two algorithms based on Expectation-Maximization (EM) and Gibbs Sampling for efficient computation of PADS from polymorphic worm samples. The new signature is capable of handling certain polymorphic worms. Our experiments show that the algorithms accurately separate new variants of the MSBlaster worm from the normal-traffic background.**

*Key Words:* System Design

## I. Introduction

An Internet worm is a self-propagation program that automatically replicates itself to vulnerable systems and spread across the Internet. It represents a huge threat to the network community [1], [2], [3], [4], [5], [6], [7]. While much recent research concentrates on worm propagation models [8], [9], [10], [11], [12], the defense against worm attacks is largely an open problem due to the fact that worms are able to spread substantially faster than humans can respond. As a result, though worm attacks fall into the category of network intrusions, most intrusion detection techniques are not suitable with worm attacks. In most cases, the defense against the worm attack can only be done reactively after the damage has already happened.

Some worm defense mechanisms, which are mainly rate-limit based [3], [13], were proposed recently trying to defend against aggressive worms based on their aggressive behaviors. 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 to modify the network stack so that the rate of connection requests to distinct destinations is bounded [13]. The main problem here is that this approach becomes effective only after the majority of all Internet hosts is upgraded with the new network stack. Although the deployment on any host may benefit the Internet community, it does not provide immediate anti-worm protection to the host itself. This gives little incentive for an individual organization to upgrade their hosts without an Internet-wide coordinated effort, as their security depends on the same action taking by the rest of the Internet. In our previous work, a distributed anti-worm architecture (DAW) has been proposed [3]. By tightly restricting the connection-failure rates from worm-infected hosts while allowing the normal hosts to make successful connections at any rate, DAW is able to slows down the worm's propagation rate significantly within a single ISP.

Most existing worms that cause huge impacts so far have very aggressive behaviors. They try to spread the Internet at the highest speed. However, it is actually a less challenging work to deal with them becuase their aggressiveness can be easily identified by comparing them with normal traffic behaviors. A good example is the Slammer worm whose propagation has been largely restricted at the later stage because of the network failures (some networks shut down under the extreme load). While Slammer spread greatly faster than Code Red, it infected fewer machines [5]. For future worm attacks, it is conceivable that worms can be modified in a stealthy way in order to avoid the detection of rate-limit based systems mentioned above. The worm can slow down the propagation rate purposely and surreptitiously compromise a vast number of systems [2].

We briefly introduce the general network intrusion detection systems related to the worm detection here as they might give us some insight to design our systems on anti-worm defense, especially for stealthy worms. There exist several approaches for intrusion detection. *Anomaly-based* systems [4], [14], [15] derive the statistical features of normal traffic. Any deviation from the profile will be treated as suspicious. Though such systems can detect previously unknown attacks, they also lead to high false positives as the behavior of legitimate activities is largely unpredictable. On the other hand, *misuse detection* systems look for particular, explicit indications of attacks such as the pattern of malicious traffic payload. They can detect the existence of known worms but fail on those that are new.

Most deployed worm-detection systems are *signature-*

0-7803-8968-9/05/$20.00 (C)2005 IEEE

*based*, which belongs to the misuse-detection category. They look for specific byte sequences (called *attack signatures*) that are known to appear in the traffic generated by certain attacks. Normally, attack signatures are manually identified by human experts through careful analysis of the byte sequence from captured attack traffic. A good signature should be the one that consistently shows up in the attack traffic but rarely appears in normal traffic.

The signature-based approaches [16], [17] have the advantage over the anomaly-based systems in that they are simple and able to operate online in real time. The problem is that they can only detect known attacks with identified signatures that are produced by experts. Automated signature generation for new attacks is extremely difficult due to three reasons. First of all, in order to create an attack signature, we must identify and isolate the attack traffic from the legitimate traffic. Automatic identification of new worms is of the most importance, which is the foundation of other defense measures. Secondly, the signature generation must be general enough to capture all attack traffic of certain types while at the same time specific enough to avoid the overlap with the contents of legitimate traffic in order to reduce false-positives. There lacks a systematic solution for this problem, which has so far been handled in an ad-hoc way based on human judgement. Finally, the system must be flexible enough to deal with the polymorphism in the attack traffic. Otherwise, worms may be programmed to slightly modify the instances of themselves deliberately each time they replicate, thus easily fool the defense system.

This paper attempts to address the above problems. We design a novel double-honeypot system which is deployed in a local network for automatic detection of worm attacks from the Internet. The system is able to isolate the attack traffic from the potentially huge amount of normal traffic on the background. It not only allows us to trigger warnings but also record the attack instances of an on-going worm epidemic. We summarize the polymorphism techniques that a worm may use to evade the detection by the current defense techniques. A new type of *position-aware distribution signature* (PADS) that is capable of detecting polymorphic worms of certain types is then discussed. The signature is a collection of position-aware byte frequency distributions, which is more flexible than the traditional signatures of fixed strings and more precise than the position-unaware statistical signatures. We describe how to match a byte sequence against the "non-conventional" PADS. Two algorithms based on Expectation-Maximization [18] and Gibbs sampling [19] has been discussed for efficient computation of PADS from polymorphic worm samples. The experiments based on variants of the MSBlaster worm are performed. The results show that our signature-based defense system can accurately separate new variants of the worm from the normal background by using the PADS signature derived from the past samples.

The rest of the paper is organized as follows. Section II proposes a double-honeypot system that can detect worm activities. Section III studies the worm polymorphism. Section IV proposes a position-aware distribution signature, and presents the algorithms for calculating such a signature and using the signature to identify worm in a byte sequence. Section V presents the experiment results. Section VI draws the conclusion and discusses the future work.

## II. Double-Honeypot System

### A. Motivation

The spread of a malicious worm is often an Internet-wide event. The fundamental difficulty in detecting a previously unknown worm is due to two reasons. First, the Internet consists of a large number of autonomous systems that are managed independently, which means a coordinated defense system covering the whole Internet is extremely difficult to realize. Second, it is hard to distinguish the worm activities from the normal activities, especially during the initial spreading phase. Although the worm activities become apparent after a significant number of hosts are infected, it will be too late at that time due to the exponential growth rate of a typical worm [8], [9], [10], [11], [12]. In contrast to some existing defense systems that require large-scale coordinated efforts, we describe a double-honeypot system that allows an individual autonomous system to detect the ongoing worm threat to its servers without external assistance. Most importantly, the system is able to detect new worms that are not seen before.

Before the architecture of our double-honeypot system is presented, we give a brief introduction of the honeypot here. Developed in recent years, honeypot is a monitored system on the Internet serving the purpose of attracting and trapping attackers attempting to penetrate the protected servers on a network [20], [21]. Honeypots fall into two categories. A *high-interaction* honeypot operates a real operating system and applications. A *low-interaction* honeypot simulates only part of the system serving as a traffic sink. In general, any network activities observed on honeypots can be considered as suspicious and it is possible to capture the latest intrusions based on the analysis of these activities. However, the information provided by the honeypots is often mixed with normal network activities as legitimate users might access the honeypots by mistake. Hours or even days are necessary for experts to manually identify the existance of new worm types, which is insufficient as a worm may have infected the whole Internet in such a period of time.

The double-honeypot system proposed in this paper is designed to detect new worms automatically. The novelty of this system is its ability to distinguish worm activities from normal network activities without the involvement of experts. Furthermore, it is purely a local system. Its effectiveness does not require a wide deployment, which is a great advantage over many existing defense systems [2], [13].

The design of our system is based on the worm's self-replication characteristics. The nature of the worms is that an infected host will try to find and infect more victims, which is

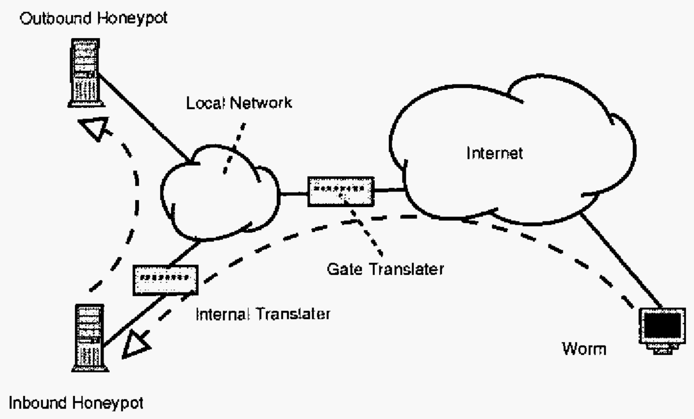


Fig. 1. Using double-honeypot detecting Internet worms

how a worm spreads itself. Therefore, outbound connections initiated from the compromised hosts are a common characteristic shared by all worms. Suppose we deliberately configure a honeypot to never initiate any outbound connections. Now if the honeypot suddenly starts to make outbound connections, it only means that the honeypot must be under foreign control. If the honeypot can be compromised, it might try to compromise the same systems on the Internet in the way it was compromised. Therefore, the situation is either a real worm attack or can be turned into a worm attack if the attacker behind the scene chooses to do so. We shall treat the two equally as a worm threat.

## B. System Architecture

Figure 1 illustrates the double-honeypot system. It is composed of two independent honeypot arrays, the *inbound array* and the *outbound array*, together with two address translators, the *gate translator* and the *internal translator*. A honeypot array consists of one or multiple honeypots, which may run on separate physical machines or on virtual machines simulated by the same computer [21]. Each honeypot in the array runs a server identical to a local server to be protected. A honeypot in the inbound (outbound) array is called an *inbound (outbound) honeypot*. Our goal is to attract a worm to hit an inbound honeypot before it compromises the local system. When the inbound honeypot attempts to compromise other machines by making outbound connections, its traffic is redirected to an outbound honeypot, which confirms the compromise of the inbound honeypot and captures its attack traffic.

An inbound honeypot should be implemented as a high-interaction honeypot with the purpose of accepting connections from outside world in order to be compromised by the worms that may pose a threat to the local server. An outbound honeypot should be implemented as a low-interaction honeypot so that it can remain uninfected when it records the worm traffic.

The gate translator is implemented at the edge router betwen the local network and the Internet. The gate translator samples the unwanted inbound connections, and redirects the sampled connections to inbound honeypots that run the server software the connections attempt to access (e.g., destination ports 80/8080 for a web server). There are several ways to determine which connections are "unwanted". The gate translator may be configured with a list of unused addresses. Connections to those addresses are deemed to be unwanted. It is very common nowadays for an organization to expose only the addresses of its public servers. The gate translator may be configured with those publicly-accessible addresses. When a service connection (e.g., to port 80) is not made to one of the servers, it is unwanted and redirected to an inbound honeypot. Suppose the size of the local address space is $N$ and there are $h$ publicly-accessible servers on a particular destination port. Typically, $N >> h$. For a worm that randomly scans that port, the chance for it to hit an inbound honeypot first is $\frac{N-h}{N}$, and the chance for it to hit a server first is $\frac{h}{N}$. With a ratio of $\frac{N-h}{h}$, it is almost certain that the worm will compromise the inbound honeypot before it does any damage to a real server within the network.

Once an inbound honeypot is compromised, it will attempt to make outbound connections. The internal translator is implemented at a router that separates the inbound array from the rest of the world. It intercepts all outbound connections from an inbound honeypot and redirects them to an outbound honeypot of the same type, which will record and analyze the traffic.

We give the following example to illustrate how the system works. Suppose that the IP address space of our network is 128.10.10.0/128, with one public web server $Y$ to be protected. The server's IP address is 128.10.10.1. Suppose an attacker outside the network initiates a worm attack against systems of type $Y$. The worm scans the IP address space for victims. It is highly probable that an unused IP address, e.g. 128.10.10.20, will be attempted before 128.10.10.1. The gate controller redirects the packets to an inbound honeypot of type $Y$, which is subsequently infected. As the compromised honeypot participates in spreading the worm, it will reveal itself by making outbound connections and provide the attack traffic that will be redirected to an outbound honeypot of the system.

We should emphasis that, the double-honeypot system is greatly different from a typical honeypot system. A typical system receives traffic from all kinds of sources, including those from the normal users. It is a difficult and tedious task to separate attack traffic from normal traffic, especially for attacks that are not seen before. It is more than often that, only after the damage of the new attacks is surfaced, the experts rush to search the recorded data for the trace of attack traffic. In our system, when an outbound honeypot receives packets from an inbound honeypot, it knows for sure that the packets are from a malicious source. The outbound honeypot does not have to face the potentially huge amount of normal background traffic that a traditional honeypot may receive.

## III. Polymorphism of Internet Worms

The double-honeypot system presented in the last section provides a means to capture the byte sequences of previous unknown Internet worms without manual analysis from the experts. The byte sequences can then be used to generate worm signatures, and future connections carrying them will be automatically blocked. This is a great advantage over the current systems because it can be done before new worms make a significant damage on the network.

The attackers will try every possible way to extend the life time of Internet worms. In order to evade the signature-based system, a polymorphic worm may appear differently each time it replicates itself. This section discusses the polymorphism of Internet worms, while the next section attempts to provide a solution against certain common polymorphism techniques.

There exist many techniques to make polymorphic worms. One technique relies on self-encryption with a variable key. This can be achieved by encrypting the body of the worm, which erases both signatures and statistical characteristics of the worm byte string. When a copy of the worm and the decryption routine is sent to another machine, the encrypted text is first turned into a regular worm program by the decryption routine. The program will then be executed to infect other victims and (possibly) damage the local system. However, the encrypted text tends to follow a uniform byte frequency distribution [22], which itself is a statistical feature that can be captured by anomaly detection based on deviation from normal-traffic distributions [4], [15]. Moreover, if the same decryption routine is always used, the byte sequence in the decryption routine can be served as the worm signature.

A more complex method of polymorphism is to change the decryption routine each time a copy of the worm is sent to another vulnerable host. This can be achieved by maintaining several decryption routines in one worm. When the worm tries to make a copy, only one routine is randomly selected and the rest routines are incorporated into the encryption text. 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. Over-sized 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, thus making it more difficult to compare the byte sequences of two instances of the same worm. However, from the statistical point of view, the frequencies of the garbage instructions can differ greatly from those of normal traffic. In that case, anomaly-detection systems [4], [15] can be used to identify the existence of the worm. Furthermore, some garbage instructions such as nop can be easily identified and removed. For better obfuscated garbage, techniques of executable analysis [23] can be used to identify and remove those instructions that will never be executed.

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. 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, with each being effective against certain attacks. The current defense techniques fall in two main categories, misuse/signature matching and anomaly detection. The former matches against known patterns in the attack traffic. The latter matches against the statistical distributions of the normal traffic. We propose a new hybrid approach based on a new type of signatures, consisting of position-aware byte frequency distributions. Such signatures can tolerate extensive, "local" changes as long as the "global" characteristics of the signature remain. Good examples are polymorphism cased by register reassignment and modest instruction substitution. We do not claim that such signatures are suitable for all attacks. On the other hand, it may work with executable-analysis techniques to characterize certain statistical patterns that appear after garbage instructions and excess jumps are removed.

## IV. Algorithms for Signature Detection

### A. Background

Most deployed defense systems against Internet worms are signature-based. They rely on the exact matching 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 the polymorphic worms whose instances can be carefully crafted to circumvent the signatures [23]. In addition, the manual identification of signatures may take days if not longer.

To address these problems, several anomaly-based systems [4], [15] use the *byte frequency distribution* (BFD) to identify the existence of a worm. Their basic approach is to derive a byte frequency distribution from the normal network traffic. When a new incoming connection is established, the payload of the packet is examined. The byte frequency distribution of the current connection is computed and compared with the derived byte frequency distribution of the normal traffic. A large deviation will be deemed as suspicious. The problem is that an intelligent attacker could easily cheat the system by attach the worm body to a lengthy normal, legitimate session. Since the majority of the payload is from legitimate operations, its byte frequency distribution will not vary much from the normal traffic. As the worm byte sequence is diluted in normal traffic, its statistic characters are smoothed out.

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

Our system inherits the positive aspects of both 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 so that the system has the flexibility against certain polymorphism. The new signature is called the *position-aware distribution signature* (PADS for short). It includes 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 in specific positions.

Consider a polymorphic worm with register reassignment (Section III). 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.

The goal of our system is to use double honeypots to capture the worm attack traffic, based on which PADS is derived and used to detect inbound worm variants that are targeted at the local systems. It provides a quick and automatic response that complement the existing approaches involving human experts. Based on PADS, the defense system will be able to identify the new variant of a worm at its first occurrence, even if such a variant has not been captured by the system previously.

## B. *Position-Aware Distribution Signature (PADS)*

We first describe what is a PADS signature, then explain how to match a byte sequence against a signature, and finally motivate how to compute such a signature based on captured worm sequences.

At each byte position $p$ of a PADS signature, the probabilistic byte-frequency distribution is a function $f_p(b)$, where $b \in [0..255]$, which is the set of possible values for a byte. $\sum_{b\in[0..255]} f_p(b) = 1$. We use $(f_1, f_2, ...f_W)$ to characterize the probabilistic 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 the legitimate traffic. The PADS signature is defined as $\Theta = (f_0, f_1, f_2, ...f_W)$, which consists of a *normal signature* $f_0$ and an *anomalous signature* $(f_1, f_2, ...f_W)$. Table I gives 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 |

TABLE I

AN EXAMPLE OF A PADS SIGNATURE WITH WIDTH $W = 10$

Consider a set of byte sequences $\mathcal{S} = \{S_1, S_2, ..., S_n\}$, where $S_i$, $1 \leq i \leq n$, is the byte sequence of an incoming connection. We want to decide whether $S_i$ is a variant of the worm by matching it against a signature $\Theta$. Let $l_i$ be the length of $S_i$. Let $s_{i,1}$, $s_{i,2}$, ..., $s_{i,l_i}$ be the bytes of $S_i$ at position 1, 2, ..., $l_i$, respectively. The value of each byte belongs to [0..255]. 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

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

which is the probability for $seg(S_i, a_i)$ to occur, given the two-dimensional distribution $\Theta$. Similarly, the matching score of $seg(S_i, a_i)$ with the normal signature is defined as

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

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 $seg(S_i, a)$ and the anomalous signature, and the "dissimilarity" between $seg(S_i, a_i)$ and the normal signature. $\Lambda(\Theta, S_i, a_i)$ is the matching score of $seg(S_i, a_i)$ with the PADS signature.

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

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,

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

Alternatively, we could use the logarithm of $\Lambda$ as the score, which makes it easier to plot our experiment results. Our final matching score of $S_i$ with the PADS signature $\Theta$ is defined as

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

The $W$-byte segment starting from $a_i$ that maximizes $\Omega(\Theta, S_i)$ is called the *significant region* of $S_i$, which is denoted as $R_i$. The matching score of the significant region is also the matching score of the whole byte sequence by definition.

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 $\Omega$, above zero means that $S_i$ is closer to the anomalous signature $(f_1, f_2, ...f_W)$; below zero means that $S_i$ is closer to the normal signature $f_0$.

Next we discuss how to calculate $\Theta$ based on the previously collected instances of a worm. Suppose we have successfully obtained a number $n$ of variants of a worm from the double-honeypot system. Each variant is a byte sequence with a variable length. It contains one copy of the worm, possibly embedded in the background of a normal byte sequence. Now let $S = \{S_1, S_2, ..., S_n\}$ be the set of collected worm variants and we will reuse the notations defined previously. 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 underlying 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 showed how to compute the significant region of a byte sequence if the signature $\Theta$ is know. Next we describe how to compute the signature if the significant regions of the variants are known.

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

$$\overline{f_p(x)} = \frac{c_{p,x}}{n}$$

One problem is that $\overline{f_p(x)}$ will be zero for those byte values $x$ that never appear at position $p$ of any significant region. However, consider that our calculation is based on a limited collection of the variants and $\overline{f_p(x)}$ is only the maximum likelihood estimation of the frequency, we are not absolutely confident that the actual frequencies are zero unless we obtain all variants of the worm. For better flexibility, we apply a "pseudo-count" to the observed byte count $c_{p,x}$. That is, the byte frequency $f_p(x)$ is defined as

$$f_p(x) = \frac{c_{p,x} + b}{n + 256 \cdot b}$$

where $b$ is a small predefined pseudo-count 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 components of the significant regions, computing the signature, using the signature to compute the new components of the significant regions, and repeating the process until convergence.

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 captured by our double-honeypot system. We want to stress that, though comparing the signature with the payload of the incoming connections is online, the signature itself is computed off-line. There is no real-time requirement.

## C. Expectation-Maximization Algorithm

Expectation-Maximization (EM) [18] is an iterative procedure to obtain the maximum-likelihood parameter estimations. Given a set of byte sequences $S$, we lacks the significant regions $R_1, R_2, ..., R_n$ of $S_1, S_2, ..., S_n$, which are the missing data in our problem. The underlying parameter $\Theta$ of our data set is also unknown. The EM algorithm is to first calculate the expected value of the missing data from the estimate of the parameter. The expectation of the missing data is then used obtain the new maximum likelihood estimate of the parameter. The expectation step and the maximization step are iterated until convergence after the initialization.

In our case, we need to calculate the expectation of the significant region from the estimate of $\Theta$. However, for any sequence $S_x$ with length $l_x$, every possible position $a_x \in [1..l_x - W + 1]$ can be considered as a candidate of the starting position for the significant region of $S_x$. As is defined in (1), the matching score for each candidate position is $\Lambda(\Theta, S_x, a_x)$, given estimate $\Theta$. The probability that a position $a_x$ is the starting position of the significant region in $S_x$ is proportional to $\Lambda(\Theta, S_x, a_x)$ [18]. That is,

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

Therefore, the expectation of the significant region can be described as

$$\begin{aligned}E(R_x) &= \sum_{a_x=1}^{l_x-W+1} R_x \times \Pr(a_x) \\ &= \sum_{a_x=1}^{l_x-W+1} R_x \times \frac{\Lambda(\Theta, S_x, a_x)}{\sum_{a_x=1}^{l_x-W+1} \Lambda(\Theta, S_x, a_x)}\end{aligned}$$

To better understand the concept, we give a simple example to show how to calculate the expectation of the significant region $R_x$ . Suppose sequence $S_x$ contains 3 bytes: 0x00, 0x02 and 0x04. The width of the signature is assumed as 2 for simplicity. There are two possible starting positions of the significant reion $R_x$, namely 0 and 1, in sequence $S_x$. We can obtain the matching scores that correspond to these two positions given the estimate of $\Theta$. The probability that the significant region starts at 0 and 1 can also be calculated. We assume $\Pr(0) = 0.25$, $\Pr(1) = 0.75$. Now the expectation of the significant reigon for $S_x$ is actually a position-aware byte-frequency distribution as well. At position $p = 1$ of the expected significant region, it contains 0.25 count of byte 0x00 and 0.75 count of byte 0x02. At position $p = 2$ of the expected significant region, it contains 0.25 count of byte 0x02 and 0.75 count of byte 0x04. In the previous subsection, we described how to compute $\Theta$ when count of the byte can only be integer. It can be easily accomodated to compute $\Theta$ when the expected significant regions are used, where the count of the byte are not integer. Therefore, from the expectation of the significant region, we are able to compute the maximum likelihood of the parameter $\Theta$.

The formal description of EM algorithm is presented in the following:

**Initialization**. The starting positions $a_1$, $a_2$, ..., $a_n$ of the significant regions for worm variants $S_1$, $S_2$, ..., $S_n$ are assigned randomly. They define the initial guess of the significant regions $R_1$, $R_2$, ..., $R_n$. The maximum likelihood estimate of the signature $\Theta$ is calculated based on the initial significant regions.

**Expectation**. The new guess of the significant regions is calculated based on the estimated PADS signature $\Theta$. We use the expectation $E(R_1)$, $E(R_2)$, ..., $E(R_n)$ to replace the initial significant region $R_1$, $R_2$, ..., $R_n$, based on the method discussed above.

**Maximization** The new maximum likelihood estimate of the signature (parameter) $\overline{\Theta}$ is calculated based on the current expected significant regions. The old estimate $\Theta$ is replaced with the new estimate $\overline{\Theta}$. Refer to Section IV-B for how the PADS signature is computed.

The algorithms terminates if the average matching score $\Omega$ is within $(1 + \varepsilon)$ of the previous iteration, where $\varepsilon$ 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 region deviates greatly from the average score, we repeat the algorithm with a smaller $W$. This process repeats until we reach a signature that matches well with the significant regions of all collected worm variants.

## D. Gibbs Sampling Algorithm

One 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 the problem, many

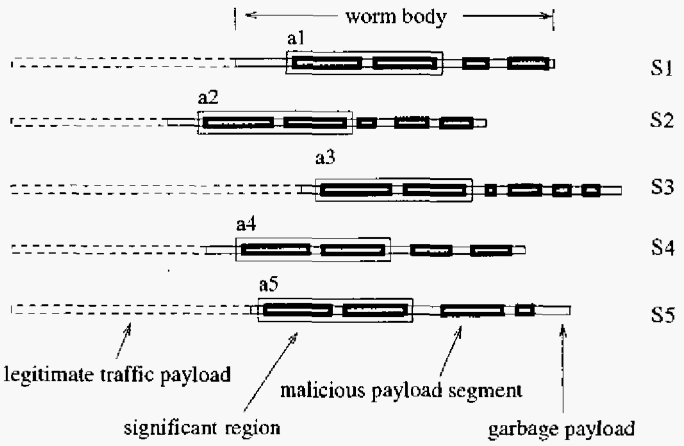


Fig. 2. Variants of the polymorphic worm

strategies have been proposed. One approach is to start with multiple random parameter configurations and look for the best among different results obtained. Another is to pre-process the data with some other methods and choose "good" initial configuration. In recent years, the simulated annealing [24] approach attracted great attention. Simply speaking, each time when the hidden data are updated, there is a small probability that a worse case is made randomly, which provides a chance for the algorithm to jump out of a local maxima. One example of the simulated annealing is the Gibbs Sampling Algorithm [19], which we will use to compute the PADS signature below.

The algorithm is initialized by assigning random starting positions for the significant regions of the worm variants. Then one variant is selected randomly. This selected variant is temporarily excluded from $S$. 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 probabilistic distribution based on the matching scores at different positions. Note that the chosen position may not be the best one. The algorithm will proceed with many iterations until a convergence criterion is met.

The detailed description of the Gibbs sampling algorithm is given below.

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

**Predictive update**. One of the $n$ worm variants, $S_x$, is randomly chosen. The signature $\Theta$ is calculated based on the other variants, $S - S_x$.

The algorithms terminates if the average matching score is within $(1 + \varepsilon)$ of the previous iteration, where $\varepsilon$ is a small predefined percentage.

**Sampling step**. Every possible position $a_x \in [1..l_x - W + 1]$ is considered as a candidate for the next starting position for 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,

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

The next iteration starts.

### *E. Signature with Multiple Separated Strings*

Thus far the signature is assumed to be a continuous string (where each string position is associated not with a byte value but with a byte frequency distribution). The definition can be easily extended for a signature to contain $k(\geq 1)$ separated strings, which may have different lengths. The significant region of a byte sequence also consists of multiple separated segments, each having a starting position and corresponding to a specific string in the signature. The matching score $\Lambda(\Theta, S_i, a_{i1}, a_{i2}, \ldots)$ should now be a function of a set of starting positions, and the significant region is defined by the set of starting positions that maximizes the matching score. Because it remains that the signature and the significant regions can be computed from each other, the EM algorithm and the Gibbs Sampling algorithm can be easily modified to compute a signature with $k$ strings.

## V. Experiments

The effectiveness of our algorithms in detecting polymorphic worms is demonstrated by experiments. The malicious payload of the MS Blaster worm, which is 1.8KB long, is used in the experiments. It exploits a vulnerability in Microsoft's DCOM RPC interface. Upon successful execution, MS Blaster worm attempts to retrieve a copy of the file msblast.exe from a previously infected host [25]. In the experiments, we artificially generate the variants of the MS Blaster worm based on the olymorphism techniques discussed in Section III.

S1, S2, ..., S5 in Figure 2 shows the examples of the polymorphic worm design. Each variant of the polymorphic worm consists of three different types of regions. The black regions are segments of the malicious payload in MS Blaster worm. Garbage payloads, which are represented as the regions with solid lines, are generated and inserted into different locations of the malicious payloads randomly. The ratio of the malicious payload and the garbage payload is 9:1[1]. In addition to garbage payload, we preceed each variant with a legitimate traffic payload from a normal session, represented by the regions with dotted lines. The length of the legitimate traffic varies from 2KB to 20KB.

During the experiments, EM and Gibbs sampling algorithm are used to identify the significant regions in different samples of the polymorphic worm. The significant regions start at a1, a2, ... a5 in Figure 2. By combining the significant regions together we obtain the PADS signature of the polymorphic

[1]This ratio is not shown proportionally in Figure 2 for better illustration.

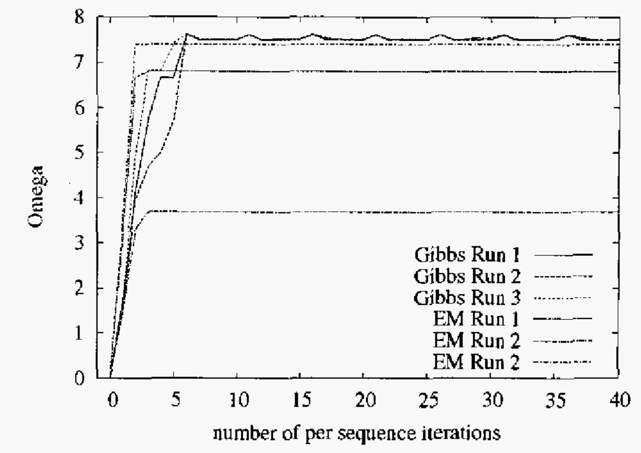


Fig. 3. The influence of different initial configurations to EM and Gibbs Sampling algorithms.

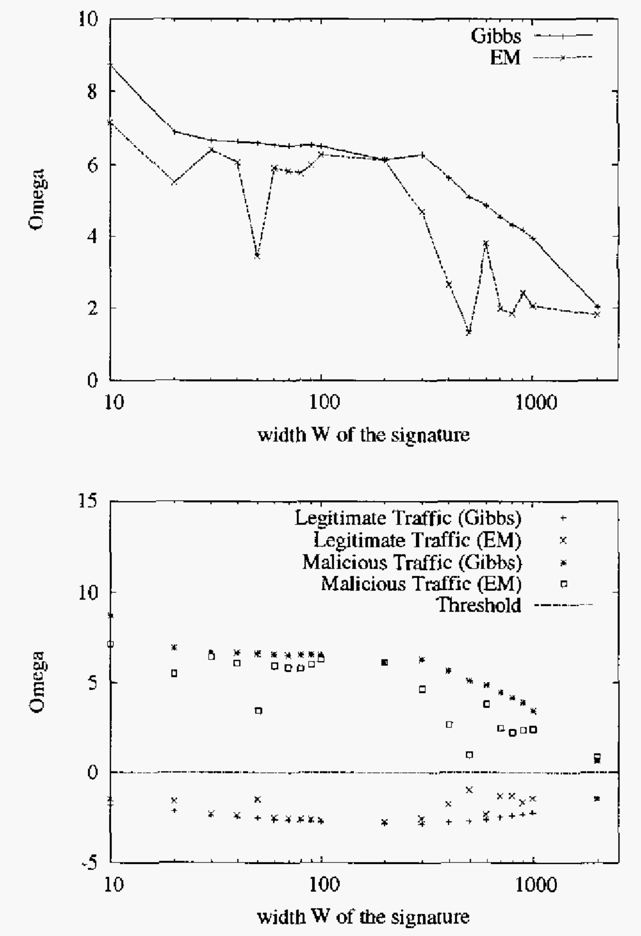


Fig. 4. The influence of different widths of the signatures to EM and Gibbs Sampling algorithms.

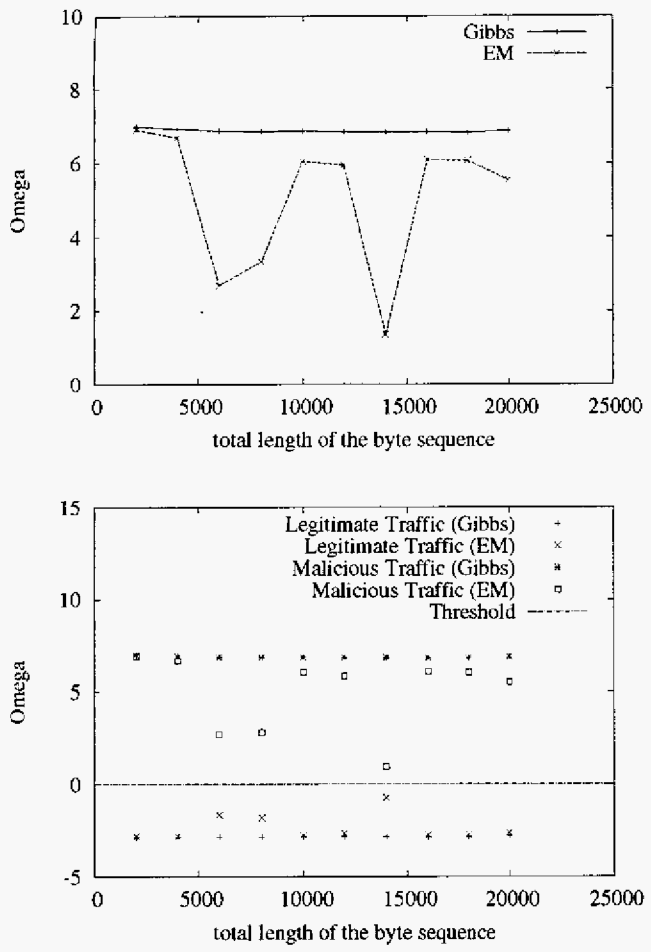


Fig. 5. The influence of different total lengths of the variants EM and Gibbs Sampling algorithms.

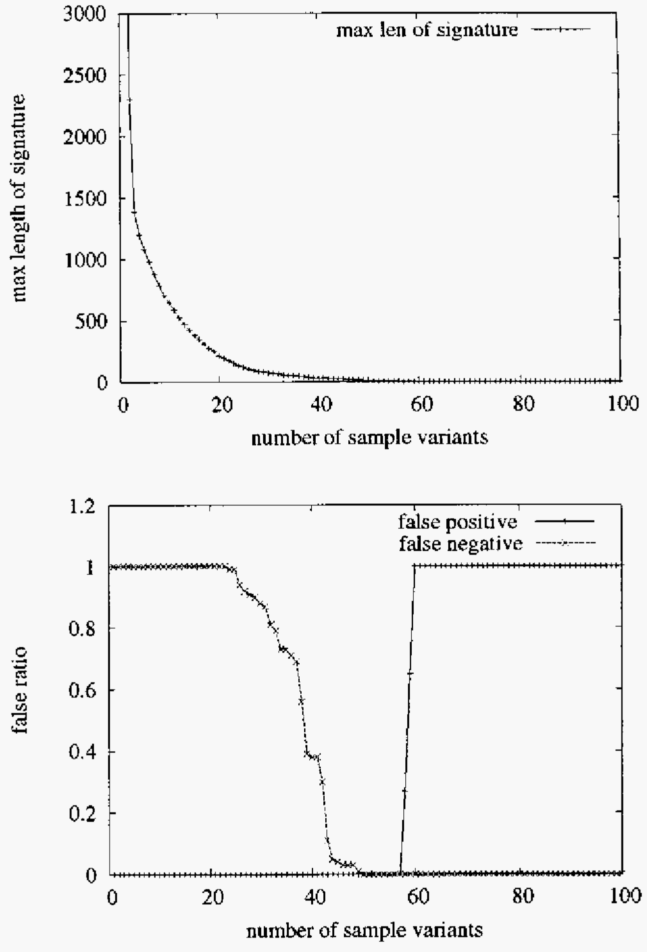


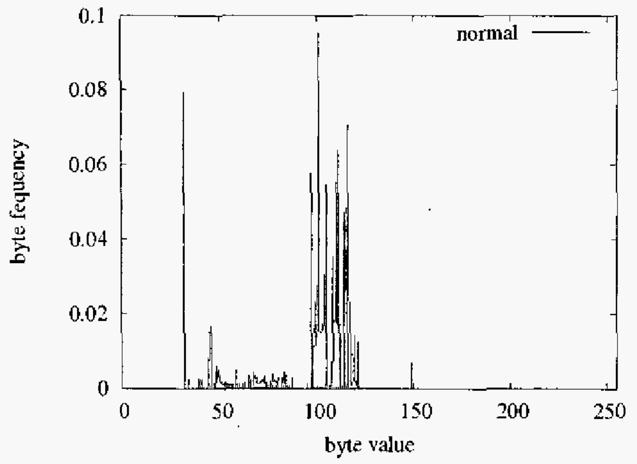
Fig. 6. The performance of signature-based system using longest common substrings.

worm. The signature is then used to detect new variants of the worm from a mixture of worm connections and normal connections.

Figure 3 presents the performance of EM and Gibbs sampling algorithm with respect to iterative cycles. As is mentioned in Section IV, during each iterative cycle EM algorithm updates the significant regions of all variants, while Gibbs sampling algorithm only adjusts one starting position of the significant region from a randomly selected variant. To make a fair comparison, we use the number of *per sequence iterations*, which is defined as the average number of the iterations each sequence undergoes, to count the iterative cycles during the process. Both EM and Gibbs sampling algorithms have been initialized randomly. Compared with EM algorithm, Gibbs sampling algorithm can always find the global maxima eventually, though takes a longer time. In addition to that, Gibbs sampling algorithm will not stabilize even in global maxima. Due to the intrinsic nature of the algorithm, there is still a slight possibility that the start position of any sequence can be changed to non-maximal places, thus leads to the fluctuation of the matching score ($\Omega$) during the process of the iterations.

Figure 4 and Figure 5 show the matching scores ($\Omega$) under different signature widths and different byte-sequence lengths, respectively. In each experiment, we generate 200 variants of the MS Blaster worm. We use 100 of them to serve as the training data and run iterative algorithms to identify the PADS signature. The rest 100 variants are mixed with 100 legitimate traffic payloads to test the performance of the signature. The upper figures in Figure 4 and Figure 5 show the average $\Omega$ of the signatures against the training data, the lower figures show, the average $\Omega$ of the signatures against the testing data, with legitimate traffic always below zero and malicious traffic always above zero. Therefore, with a threshold of 0, worm variants are distinctively seperated from the legitimate traffic. Our experiment shows that our methods are able to successfully identify the variants of the worm with no misidentification.

Figure 4 also shows that increasing the width of the signature $W$ will decrease the average matching score of the signature against new variants of the worm. The reason is that increasing the width of the signature means the significant



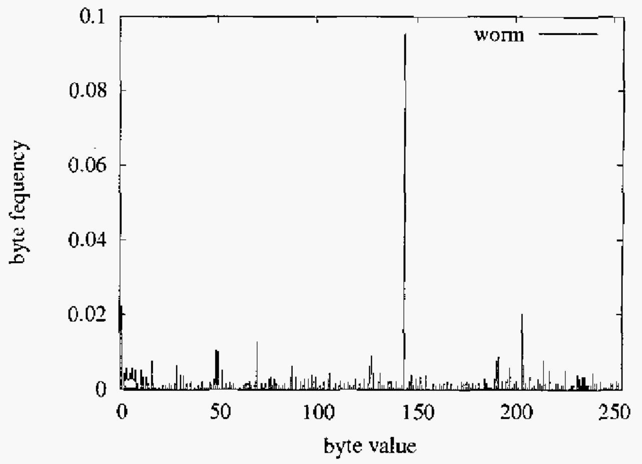
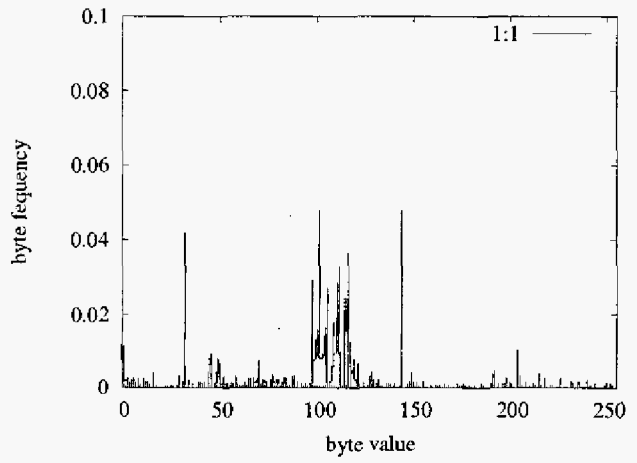


Fig. 7. The byte frequency distributions of normal (left) traffic and worm body (right) traffic.



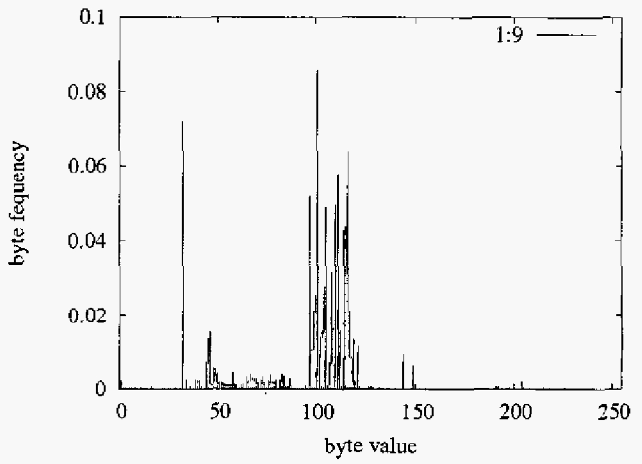


Fig. 8. The byte frequency distributions of variants of the worm. Left: worm body is attached to an equal length legitimate traffic payload. Right: worm body is attached to a legitimate traffic payload with 9 times the length of worm body.

region inside a variants is able to contain more normal traffic or garbage payload and decreases the matching score. Figure 5 shows that increasing the length of the attached normal traffic payload in the worm, which has been widely used by some polymorphic worms to elude the anomaly-based systems, provides no help to avoid detection in our system. The reason is that only the best match of the significant region is used in our algorithm.

In comparision, we also perform experiments with some existing methods. Figure 6 shows the results based on the *longest common substrings* method [20]. In these experiments, we identify the longest common substrings using the worm variants from our training data. The longest common substring is then served as the signature to detect the worm variants in the test data. As we can see, as the number of the training variants increase, the length of the longest common substring decreases. As a result, while the false negative ratio decreases, the false positive ratio increases dramatically. Without the requirement of exact matching, a PADS signature is able to retain much more (particularly statistical) characteristics of a polymorphic worm.

Figure 7 and Figure 8 show the position-*unaware* byte frequency distributions over the entire byte sequences of the training data. The left of Figure 7 is the byte frequency distribution from the legitimate connection session. The right of Figure 7 is from the payload of the MS Blaster worm. As we can see, these two distributions are greatly different. There should be no trouble to distinguish the malicious payload from the normal traffic. However, if we attach the worm body to a legitimate traffic payload, the difference is not so appreant. As is shown In Figure 8, when we attach the worm body to a really long legitimate traffic, e.g., 9 times the length of the malicious traffic (right of Figure 8), it will not be a easy task to distinguish the variants of the worm from the payloads of normal, legitimate connection sessions. Even if some methods can distinguish these two under such ratio, the attackers can certainly make the situation worse by increasing the ratio to a very large number. Such an example further demonstrates the importance using position-*aware* distribution signature in defending against polymorphic worm attacks.

## VI. Conclusion and Future work

In this paper, we provide a new defense system to detect the attacks of malicious Internet worms. The key idea is to capture the samples of the Internet worm using proposed double-honeypot system before the protected server has been compromised. Those IP addresses that are unreachable from the outside are used to attract and trap the attackers. The system is especially useful in large networks where large number of unreachable IP addresses exist.

Our system is able to defend against polymorphic worms. After collecting a number of variants of polymorphic worm, the system uses iterative algorithms to find the PADS signature of the worm, which is used to detect future worm attacks even if new variants have not been captured before. In our experiment, a 100% accuracy has been achieved to detect the variants of MSBlaster worm which means all malicious traffic can be detected and all legitimate traffic can pass through the system with no false positives.

The system is completely automatic. It requires no involvement of human experts, which is typically the drawback of the regular signature-based system. The system also tolerates some modifications of the worm where both signature- and anomaly-based systems may fail.

In our future work, we plan to evaluate the system in a live environment. We also need some further improvement of our proposed iterative algorithms. For example, what should we do to distinguish several different worms from a mixture collection of the variants of these worms. The research in these directions will provide a more robust and reliable system to defend against future worm attacks.

## References

[1] S. Staniford, V. Paxson, and N. Weaver, "How to 0wn the Internet in Your Spare Time," in *Proceedings of the 11th USENIX Security Symposium (Security '2002)*, San Francisco, California, USA, Aug. 2002.

[2] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Internet Quarantine: Requirements for Containing Self-Propagating Code," in *Proceedings of the* $22^{nd}$ *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM '2003)*, San Francisco, California, USA, Apr. 2003.

[3] S. Chen and Y. Tang, "Slowing Down Internet Worms ," in *Proceeding of the* $24^{th}$ *International Conference on Distributed Computing and Systems (ICDCS '2004),*, Tokyo.Japan, Mar. 2004.

[4] C. Kruegel and G. Vigna, "Anomaly Detection of Web-based Attacks," in *Proceedings of the* $10^{th}$ *ACM Conference on Computer and Communication Security (CCS'2003).* Washington D.C., USA: ACM Press, Oct. 2003, pp. 251–261.

[5] D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, "Inside the Slammer Worm," *IEEE Magazine of 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," in *Proceedings of the* $7^{th}$ *USENIX Security Conference (Security '1998)*, San Antonio, Texas, USA, Jan. 1998, pp. 63–78.

[7] M. Eichin and J. Rochlis, "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," in *Proceedings of the 1989 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 1989, pp. 326–344.

[8] C. C. Zou, W. Gong, and D. Towsley, "Code Red Worm Propagation Modeling and Analysis," in *Proceedings of the* $9^{th}$ *ACM Conference on Computer and Communications Security (CCS '2002).* Washington, DC, USA: ACM Press, Nov. 2002, pp. 138–147.

[9] C. C. Zou, L. Gao, W. Gong, and D. Towsley, "Monitoring and Early Warning for Internet Worms," in *Proceedings of the* $10^{th}$ *ACM Conference on Computer and Communication Security (CCS '2003).* Washington D.C., USA: ACM Press, Oct. 2003, pp. 190–199.

[10] Z. Chen, L. Gao, and K. Kwiat, "Modeling the Spread of Active Worms," in *Proceedings of the* $22^{nd}$ *Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM' 2003)*, San Francisco, California, USA, Mar. 2003, pp. 1890–1900.

[11] David Moore and Colleen Shannon and K Claffy, "Code-Red: A Case Study on the Spread and Victims of an Internet Worm," in *Proceedings of the* $2^{nd}$ *Internet Measurement Workshop (IMW '2002)*, Marseille, France, Nov. 2002, pp. 273–284.

[12] J. O. Kephart and S. R. White, "Directed-Graph Epidemiological Models of Computer Viruses," in *Proceedings of the 1991 IEEE Symposium on Security and Privacy*, Oakland, California, USA, May 1991, pp. 343–361.

[13] M. M. Williamson, "Throttling Viruses: Restricting Propagation to Defeat Malicious Mobile Code," in *Proceeding of the* $18^{th}$ *Annual Computer Security Applications Conference (ACSAC '2002)*, Las Vegas, Neveda, USA, Oct. 2003.

[14] H. Javitz and A. Valdes, "The NIDES Statistical Component Description and Justification," Computer Science Laboratory, SRI International, Menlo Park, California, USA, Tech. Rep., 1994.

[15] K. Wang and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection," in $7^{th}$ *International Symposium on Recent Advances in Intrusion Detection (RAID '2004)*, Sophia Antipolis, French Riviera, France, Sept. 2004.

[16] K. Ilgun, R. Kemmerer, and P. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach," *IEEE Trans. Software Eng.*, vol. 2, pp. 181–199, 1995.

[17] U. Lindqvist and P. Porras, "Detecting Computer and Network Misuse Through the Production-Based Expert System Toolset (P-BEST)," in *Proceedings of the 1999 Symposium on Security and Privacy*, Oakland, California, USA, May 1999.

[18] 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.

[19] 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.

[20] C. Kreibich and J. Crowcroft, "Honeycomb Creating Intrusion Detection Signatures Using Honeypots," in $2^{nd}$ *Workshop on Hot Topics in Networks (HotNets-II)*, Cambridge, Massachusetts, USA, Nov. 2003.

[21] N. Provos, "A virtual Honeypot Framework," Center for Information Technology Integration, University of Michigan, Ann Arbor, Michigan, USA, Tech. Rep. CITI Technical Report 03-1, Oct. 2003.

[22] C. Kaufman, R. Perlman, and M. Speciner, *Network Security: Private Communication in a Public World.* Upper Saddle River, NJ, USA: Prentice Hall, Inc., 2002.

[23] M. Christodorescu and S. Jha, "Static Analysis of Executables to Detect Malicious Patterns," in *Proceedings of the 12th USENIX Security Symposium (Security '2003)*, Washington D.C., USA, Aug. 2003.

[24] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 6, pp. 721–741, 1984.

[25] CERT/CC, "CERT Advisory CA-2003-20 - W32/Blaster worm," 2003. [Online]. Available: http://www.cert.org/advisories/CA-2003-20.html