available at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/cose

Computers & Security

Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms☆

Yong Tang[a,b], *Bin Xiao*[b,*], *Xicheng Lu*[a]

[a]*College of Computer, National University of Defense Technology, Changsha, Hunan, 410073, CHINA*
[b]*Department of Computing, The Hong Kong Polytechnic University, Hung Hom, Kowloon, Hong Kong*

ARTICLE INFO

*Article history:*Received
2 February 2009
Received in revised form
18 May 2009
Accepted 8 June 2009

Keywords:
Exploit-based signature generation
Polymorphic worms
Sequence alignment
Simplified regular expression
One-byte invariant
Distance restriction

ABSTRACT

In this paper, we propose Simplified Regular Expression (SRE) signature, which uses multiple sequence alignment techniques, drawn from bioinformatics, in a novel approach to generating more accurate exploit-based signatures. We also provide formal definitions of what is ‘‘a more specific’’ and what is ‘‘the most specific’’ signature for a polymorphic worm and show that the most specific exploit-based signature generation is NP-hard. The approach involves three steps: multiple sequence alignment to reward consecutive substring extractions, noise elimination to remove noise effects, and signature transformation to make the SRE signature compatible with current IDSs. Experiments on a range of polymorphic worms and real-world polymorphic shellcodes show that our bioinformatics approach is noise-tolerant and as that because it extracts more polymorphic worm characters, like one-byte invariants and distance restrictions between invariant bytes, the signatures it generates are more accurate and precise than those generated by some other exploit-based signature generation schemes.

1. Introduction

Internet worms propagate over the Internet through infections in which a worm sends a payload (such as a shellcode or a copy of itself) to the target host by exploiting operation system or network service vulnerabilities (Tang et al., 2009). A polymorphic worm is a worm that changes its appearance at each infection, making detection and prevention much harder. One of the most popular and effective ways to detect worms is signature-based detection (also called content-based filtering). The generated signature can be either exploit-based or vulnerability-based: an exploit-based signature describes the characteristics of one or a number of exploits; a vulnerability-based signature describes properties of one vulnerability and can detect all possible exploits utilizing this vulnerability.

Although vulnerability-based signature could be more effective, so far both exploit-based and vulnerability-based signatures have equal importance for worm detection in current IDSs (Vulnerability, 2005) for the following reasons. First, the real vulnerability-based signature[1] can be generated only if a vulnerability is disclosed (Brumley et al., 2006); however an exploit-based signature can be fast and timely generated to detect zero-day exploits of an undisclosed

☆ The work was partially supported by China 973 Grant No. 2005CB321801, China 863 Grant No. 2009AA01Z432 and HK RGC PolyU 5311/06E.

* *Corresponding author.*
E-mail addresses: ytang@nudt.edu.cn (Y. Tang), csbxiao@comp.polyu.edu.hk (B. Xiao), xclu@nudt.edu.cn (X. Lu).

[1] A real vulnerability signature can match all possible exploits (inputs) which satisfy a vulnerability condition in a program. In fact, most automatically generated vulnerability-based signatures can only provide approximate solutions, which can detect a part of all possible exploits.

0167-4048/$ – see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cose.2009.06.003

vulnerability. Second, most IDS/anti-virus vendors generate both types of signatures. Through them, we not only know when we were attacked, but how we were attacked by investigating whether an exploit is new or a variant of a well-known exploit. Third, some vulnerability-based signatures cannot be adopted in current IDSs since they require that IDSs are able to perform profound protocol analysis (Crandall et al., 2005). However, this obviously overestimates the capability of current IDSs. On the contrary, it has no such requirement to generate exploit-based signatures and hence exploit-based signature generation systems are more compatible to current IDSs. Compared with manual signature generation, the recently studied automatic signature generation approaches can generate more accurate signatures for worms much faster, especially for polymorphic worms. Thus, this paper will only focus on automatic exploit-based signature generation for polymorphic worms.

We find that currently available exploit-based signature generation approaches may fail to create accurate signatures from collections of exploit samples of polymorphic worms. To understand why, consider a sample of a polymorphic worm (an infection flow) as a string sequence consisting of *invariant* and *wildcard* bytes. Invariant bytes have fixed values and are present in every worm sample. In contrast, wildcard bytes change their values in each sample. Fig. 1 shows an example of the polymorphic Code Red II worm, which contains a sequence of seven invariant parts: ‘‘GET’’, ‘‘.ida?’’, ‘‘XX’’, ‘‘%u’’, ‘‘%u780’’, ‘‘=’’, and ‘‘HTTP/1.0 r n’’. Typically, we would try to extract the invariant parts of polymorphic worms as their signatures, since invariant bytes in a worm flow are composed of a number of invariant parts which are crucial to the exploitation of a vulnerable server (Crandall et al., 2005; Newsome et al., 2005). But this raises two difficulties. First, some invariant parts in polymorphic worms cannot be extracted. Earlier approaches (Kreibich and Crowcroft, 2003; Kim and Karp, 2004; Singh et al., 2004) were able to generate only a single invariant part. More up-to-date approaches (Polygraph (Newsome et al., 2005) and Hamsa (Li et al., 2006)) can extract most invariant parts except for one-byte invariant parts (such as ‘‘=’’ in the Code Red II worm). Second, no approach takes into account all distance restrictions between invariant parts (like ‘‘‘%u780’ is 4 bytes after ‘%u’’’ in the Code Red II worm). Yet we might surmise that whether one-byte invariant (signature) parts and distance restrictions are valuable in worm detection. Consider Table 1, which shows part of rules (a rule represents a signature) of two well-known IDSs, Snort and Bro. It can be seen that 40.6% of rules in Snort's exploit.rules file consist of multiple invariant parts, 38.7% contain distance restrictions, and 22% contain one-byte signature parts. These figures suggest that distance restrictions could play a role in worm infections and that signatures generated by previous NSG systems may not be accurate enough to identify worms, resulting in a high rate of false positives.

Table 1 – Analysis of current IDS rules.

IDS (rule file name)	Snort (exploit.rules)	Bro (signatures.sig)
# of rules analyzed	155	2966
# (%) of rules with ordered multiple invariant parts	63 (40.6%)	365 (12.3%)
# (%) of rules with distance restriction	60 (38.7%)	312 (10.5%)
# (%) of rules with one-byte invariant part	34 (22%)	86 (2.9%)

In this paper, we model the problem of generating accurate exploit-based signature given some zero-day exploits of a new polymorphic worm. We propose a signature type – *Simplified Regular Expression (SRE) signature*. SRE can be easily transformed to rules in current IDSs to accurately capture polymorphic worms. Based on SRE, we provide formal definitions of what is ‘‘a more specific’’ and what is ‘‘the most specific’’ signature of a polymorphic worm such that we can compare the accuracy of two signatures. We show that it is an NP-hard problem to generate the most specific SRE signature of a polymorphic worm. Our approach proposed in the paper is a network-based and exploit-based scheme to generate SRE signature for a single polymorphic worm. It is a bioinformatics approach, inspired by the multiple sequence alignment techniques (used to identify motifs and domains preserved by evolution) in bioinformatics. The approach consists of three steps: multiple sequence alignment, noise elimination and signature transformation. The multiple sequence alignment step is based on the T-coffee algorithm (Notredame et al., 2000) and our newly proposed CSR (Consecutive Substrings Rewarded) algorithm. CSR is a pairwise alignment algorithm that captures contiguous invariant parts in worm samples. It is an extension of the Needleman–Wunsch algorithm and is based on rewarding consecutive matches. We minimize the impact of noise samples by using a noise elimination algorithm that is relevant to a noise tolerance rate θ. A signature transformation step is used to derive accurate SRE signatures that can be conveniently used in current IDSs. We analyze the complexity of the approach and the impact of its noise tolerance rate θ.

Experiments on a range of polymorphic worms and real-world polymorphic shellcodes show that the SRE signatures generated by our approach are more accurate than those of previous methods, extracting more polymorphic worm characters, including the one-byte invariant and distance restriction, and distinguishes between benign traffic and worm

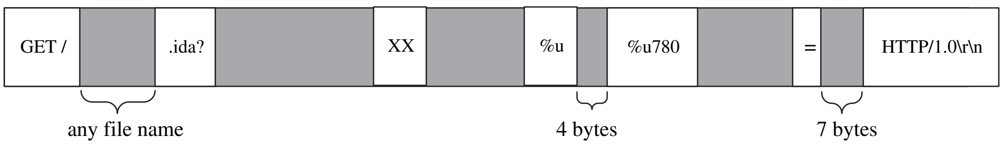


Fig. 1 – Polymorphic Code Red II worm. Shaded content represents wildcard bytes. Unshaded content represents invariant bytes.

traffic. Our approach is resilient to a small portion of noise samples. Our experiments show that our approach (combining the CSR and T-coffee algorithms) is more accurate than other sequence alignment algorithms with regard to accuracy.

The rest of the paper is organized as follows. We first review the related work in Section 2. In Section 3, we propose the SRE signature and raise the accurate exploit-based signature generation problem. In Section 4, we present a bioinformatics approach that generates accurate SRE signatures given some exploit samples from a polymorphic worm. Section 5 evaluates our approach with and without noise sample interference. We discuss the complexity, the scope and limitation of our approach in Section 6. Finally, Section 7 offers our conclusion.

2. Related work

2.1. *Polymorphism technique*

Polymorphism technique has been exploited to create worm flows and worm writers have started using polymorphic engines in recent years. Published polymorphic shellcode generators include ADMmutate, PHATBOT, Jempiscodes, PHolyP, Clet, TAPiON, and Metasploit. Common techniques used to write polymorphic shellcodes include Garbage and NOP insertions, register shuffling, equivalent code substitution, and encryption/decryption. Although it has been shown that it may be possible to create polymorphic shellcode that has no artifact (i.e. invariant bytes) (Song et al., 2007), the paper also says that signature-based methods still work on even state-of-the-art polymorphic worms while Newsome et al. observe that, in practice, most polymorphic worms must contain invariant bytes as they are crucial to exploiting vulnerable servers (Newsome et al., 2005).

2.2. *Exploit-based automatic signature generation*

There are a number of automatic signature generation schemes that can output exploit-based signatures. They can be broadly classified as either network-based or host-based. A network-based scheme relies solely on network traffic to generate exploit-based signatures while a host-based scheme needs to collect more information at a host, such as the source/binary code of a vulnerable program, the execution context of an exploitation. The categorization of existing signature generation schemes is shown in Table 2.

Table 2 – Categorization of existing signature generation schemes.

Signature generation mechanisms	Output signature type	
	Exploit-based signature	Vulnerability-based signature
Network-based	Honeycomb (Kreibich and Crowcroft, 2003), Autograph (Kim and Karp, 2004), PAYL (Wang et al., 2003), EarlyBird (Singh et al. 2004), Polygraph (Newsome et al., 2005), Hamsa (Li et al., 2006), Nemean (Yegneswaran et al., 2005), Our approach	LESG (Li et al., 2007)
Host-based	TaintCheck (Newsome et al., 2005), DACODA (Tang et al., 2007), DIRA (Smirnov, 2005)	Vulnerability Signature (Brumley et al., 2006), Vigilante (Costa et al., 2005), COVERS (Liang and Sekar, 2005a), Packet Vaccine (Wang et al., 2006), ARBOR (Liang and Sekar, 2005b), Automatic Diagnosis (Xu et al., 2005)

2.2.1. *Network-based and exploit-based schemes*

Early network-based signature generation schemes, including Honeycomb (Newsome et al., 2005), Autograph (Kim and Karp, 2004), PAYL (Wang et al., 2003), and EarlyBird (Singh et al., 2004), do not work well with polymorphic worms since their signatures contain only a single contiguous string. Polygraph (Newsome et al., 2005) and Hamsa (Li et al., 2006) are two token-based approaches that select a set of tokens that have high coverage of suspicious traffic pool and will not lead to a high false positive. However, the signatures generated by Polygraph and Hamsa are not accurate enough because they do not take account one-byte invariant bytes and distance restrictions in polymorphic worms. Nemean (Yegneswaran et al., 2005) uses a semantic analysis of network protocols to generate connection and session signatures. It requires to explicitly analyze protocols and the generated connection signatures are similar to the token-subsequence signatures in Polygraph but have no information of distance restrictions on tokens. Our approach is a network-based and exploit-based scheme. In comparison with most recent work in this category, such as Polygraph (Newsome et al., 2005), Nemean (Yegneswaran et al., 2005) and Hamsa (Li et al., 2006), our approach can generate more accurate signatures and does not rely on well classified worm flow pool and explicit protocol analysis. Compared with our previous preliminary work in (Tang et al., 2007), this paper improves the CSR algorithm and the proposed bioinformatics approach is more complex to generate accurate signatures by integrating new steps, i.e. multiple sequence alignment, noise elimination and signature transformation.

2.2.2. *Host-based and exploit-based schemes*

One host-based and exploit-based scheme, TaintCheck (Newsome and Song, 2005), uses dynamic taint analysis to detect anomaly instruction execution and then outputs a three-byte signature, which could be used to overwrite a jump target. DACODA (Crandall et al., 2005) adopts a similar mechanism and outputs a set of tokens as a signature to detect attacks. DIRA (Smirnov, 2005), a compiler, can transform arbitrary programs to a form to detect control hijacking attacks. Malicious input (packet) is identified as the signature. Compared to these schemes, our approach does not require to know some unrevealed information (e.g., TaintCheck and DACODA need to know the vulnerable program, DIRA must have source code).

Our proposed approach aims to generate more accurate signatures than theirs (three-bytes Newsome and Song, 2005, tokens Crandall et al., 2005, and whole packet Smirnov, 2005) to recognize and filter out polymorphic worm traffic.

2.3. *Vulnerability-based automatic signature generation*

In recent years, we have witnessed some vulnerability-based automatic signature generation schemes. Brumley et al. (2006) use static analysis techniques to generate two approximate vulnerability-based signatures (MEP symbolic constraint signature and regular expression signature). Vigilante (Costa et al., 2005) uses dynamic dataflow analysis to generate vulnerability-based signatures. COVERS (Liang and Sekar, 2005a) generates vulnerability-based signatures through a forensic mechanism to correlate victim server's memory with network input message. Packet vaccine (Wang et al., 2006) randomizes address-like strings in packet payloads, carries out exploit detection and vulnerability diagnosis, and finally outputs vulnerability-based signatures, called Application-level Signature. Xu et al. (2005) present a method that traces the corrupting instructions and generates signatures using data/address values embedded in the malicious input message. The common limitation of these schemes is that they need to know the vulnerable program and some need to test variant implementation versions to identify the vulnerable program. Their side-effects are witnessed, such as impasting the performance of the protected host (Brumley et al., 2006; Costa et al., 2005; Liang and Sekar, 2005a; Wang et al., 2006; Xu et al., 2005); requiring patching of kernels (Wang et al., 2006; Xu et al., 2005) or modification of run time libraries (Brumley et al., 2006; Liang and Sekar, 2005a; Xu et al., 2005); generating non-compatible signatures for network filtering (Costa et al., 2005; Liang and Sekar, 2005b; Xu et al., 2005).

3. Accurate exploit-based signature generation problem

In this section, we first propose a new signature – *Simplified Regular Expression* (*SRE*) signature. Based on this signature, we formally define what is "a more specific" signature and what is "the most specific" signature to compare SRE signatures. Then we present the accurate exploit-based signature generation problem and we show that the most specific signature generation for polymorphic worms is NP-hard.

3.1. *SRE signature*

Although regular expression has been widely employed in intrusion detection, like in Snort and Bro, many syntax rules in regular expression are rarely used for worm detection in the real-world. Hence, we propose *Simplified Regular Expression* (*SRE*) signature that is efficient in worm matching and compatible with current IDSs.

An SRE signature is a regular expression that contains only two qualifiers – ".*" and ".{k}". For simplicity, these can each be further abbreviated by replacing ".*" with "*", which represents an arbitrary string (including a zero-length string), and by replacing ".{k}" with "[k]", which represents a string consisting of k arbitrary characters and defines a *distance restriction* whose length is k. For example, "'one'[2]'two'*" is an SRE signature that is equivalent to the regular expression "one.{2}two.*". SRE signatures are still regular expressions, but contain less syntax rules. In other words, the set of SRE signatures is a subset of the set of regular expressions. Since an arbitrary SRE signature still satisfies the definition of a regular expression, it can be converted to an equivalent NFA/DFA, and be conveniently used for detection in any IDS that supports regular expression. Suppose that $\Phi = \{*, [k]\}$ is the set of the two qualifiers and that Σ^+ is the set of nonempty strings over a finite alphabet Σ. An SRE signature is defined as follows:

Definition 1. ((*SRE signature*)) *An SRE signature is a signature in the form of* $(q_0)s_1q_1s_2\ldots q_{k-1}s_k(q_k)$, *where* $q_i \in \Phi$ *is a qualifier,* $s_i \in \Sigma^+$ *is a substring* ($i \in [0, k]$), (q_0) *and* (q_k) *means* q_0 *and* q_k *are optional.*

Given that $\mathcal{X}$ is an SRE signature, we use $|\mathcal{X}|$ to denote the total length of substrings in $\mathcal{X}$, and use $||\mathcal{X}||$ to denote the total length of substrings plus the total length of distance restrictions in $\mathcal{X}$. Thus, $||\mathcal{X}|| - |\mathcal{X}|$ refers to the total length of distance restrictions in $\mathcal{X}$. For instance, if $\mathcal{X} = {''}*'a'[2]'bcd'[1]'efgh'*{''}$, then $|\mathcal{X}| = 8(1+3+4)$, $||\mathcal{X}|| = 11(1+2+3+1+4)$, and the total length of distance restrictions in $\mathcal{X}$ is 3 ($11-8$, or $2+1$). Table 3. shows rules of some current IDSs and their equivalent SRE signatures. The examined intrusion detection rules, comprising distance restriction and 1-byte invariant part, can be easily transformed to SRE signatures, and reversely.

3.2. *Comparison of signature accuracy*

We aim to generate more accurate signatures for polymorphic worms, so it is natural to ask what is "a more accurate" and

Table 3 – Current IDS rules and their corresponding SRE signatures.

	Rule of Snort	Rule of Bro
An example rule with distance restriction and 1-byte invariant part	alert udp $EXTERNAL_NET any → $HOME_NET 500 (msg:"EXPLOIT ISAKMP delete hash with empty hash attemp"; content:"|08|"; depth:1; offset:16; content:"|0C|"; depth:1; offset:28; content:"|0004|"; depth:2; offset:30; reference:bugtraq,9416; reference:bugtraq,9417; reference:cve,2004-0164; classtype:misc-attack; sid:2413; rev:9;)	signature s2b-2101-9 {ip-proto = = tcp dst-port = = 139 event "NETBIOS SMB SMB_COM_TRANSACTION Max Parameter and Max Count of 0 DOS Attempt" tcp-state established,originator payload /\x00/payload /.{3}\xFFSMB%/payload/.{42}\x00\x00\x00\x00/}
Equivalent SRE signature	[16]'\x08'[28]'\x0C'[30]'\x00\04'*	'\x00'[3]'\xFFSMB'[42]'\x00\x00\x00\x00'*

what is "the most accurate" signature for a polymorphic worm? Intuitively, the "the most accurate" signature should be "the most specific" while also being "generally" representative. It would be more specific the more features of a worm it contained. This would reduce false positives. At the same time, however, it would be generally representative in that it would not contain useless or incorrect features of a worm that would lead to false negatives. The following defines the more specific than relationship that we use to compare two SRE signatures in terms of their specificity.

Definition 2. ((*contain*, $\lhd$)) *Letting* $\mathcal{X}$ *and* $\mathcal{Y}$ *be two SRE signatures, we say that* $\mathcal{Y}$ *contains* $\mathcal{X}$*, denoted by* $\mathcal{X} \lhd \mathcal{Y}$*, if* $L(\mathcal{X}) \subseteq L(\mathcal{Y})$*. That is, the strings that match signature must also match signature* $\mathcal{Y}$*.*

Definition 3. ((*more specific than*, $\succ$)) *Let* $\mathcal{X}$ *and* $\mathcal{Y}$ *be two SRE signatures. If* $\mathcal{X} \lhd \mathcal{Y}$ *and* $\|\mathcal{X}\| > \|\mathcal{Y}\|$*, we say* $\mathcal{X}$ *is more specific than* $\mathcal{Y}$*, or* $\mathcal{Y}$ *is more general than* $\mathcal{X}$*, and this is denoted by* $\mathcal{X} \prec \mathcal{Y}$*.*

Here are some examples for Definition 2 and Definition 3: "'abc'*'bcd'" $\lhd$ "'ab'* 'c'*", "'ab'[2]'cd'" $\lhd$ "'ab'*'cd'"; "'aaa'*'b'" $\succ$ "'aa'*'b'", "'aa'[3]'c'" $\succ$ "'aa'*'c'". Note that if a string or a network flow x (as a special SRE signature with only one substring and without qualifiers) matches an SRE signature $\mathcal{Y}$, we also use "$x \lhd \mathcal{Y}$" to denote it. For instance, the string "abcdef" matches the signature "'ab'*'ef'" and can be denoted by "abcdef" $\lhd$ "'ab'*'ef'".

3.3. *Most specific signature generation problem*

Given polymorphic worm samples, that is, a set of byte sequences, from zero-day exploits, it is possible to create the most specific (or accurate) signature for worm detection. In this section, we define the most specific signature generation problem for a polymorphic worm, and show that the problem is NP-hard. Since a sequence is a string that can be treated as a special SRE signature without any qualifiers, we can define the signature of a sequence set.

Definition 4. ((*Signature of a Sequence Set*)) *Let* $R = \{s_1, s_2, \ldots, s_k\}$ *be a set of sequences.* $\mathcal{X}$ *is an SRE signature of* R *if and only if for each* $s \in R$*,* $s \lhd \mathcal{X}$ *and this is denoted by* $R < .\mathcal{X}$*.*

Definition 5. ((*the Most Specific Signature of a Sequence Set*)) *If* $\mathcal{X}$ *is an SRE signature of* $R = \{s_1, s_2, \ldots, s_k\}$*, a set of sequences, and for any other SRE signature* $\mathcal{Y}$ *such that* $R < .\mathcal{Y}$*,* $\mathcal{X} \lhd \mathcal{Y}$ *always holds, we say* $\mathcal{X}$ *is the most specific signature (MSSig) of* R*, and this is denoted by* $\mathcal{X} = MSSig(R)$*.*

MSSG (the Most Specific Signature Generation) Problem.
INPUT: $R = \{s_1, \ldots, s_n\}$ is a sample set of a polymorphic worm $\mathbf{w}$.
OUTPUT: A signature $\mathcal{X}$ such that $\mathcal{X} = MSSig(R)$.

From the SRE signature property, we know that given a set of sequences R, *MSSig*(R) is unique. By reduction from a known NP-complete problem, the **LCS** (Longest Common Subsequence) problem (22), we have the following theorem.

Theorem 1. MSSG problem is NP-hard.

Definition 6. ((*Signature of a Polymorphic Worm*)) *Given a polymorphic worm* $\mathbf{w}$*, if all possible samples of* $\mathbf{w}$ *match an SRE signature* $\mathcal{X}$*, then* $\mathcal{X}$ *is a signature of* $\mathbf{w}$*, and this is denoted by* $\mathbf{w} < .\mathcal{X}$*.*

Definition 7. ((*the Most Specific Signature of a Polymorphic Worm*)) *Given a polymorphic worm* $\mathbf{w}$ *and its SRE signature* $\mathcal{X}$*, if* $\mathbf{w} < .\mathcal{X}$ *and for any other SRE signature* $\mathcal{X}'$ *such that both* $\mathbf{w} < .\mathcal{X}'$ *and* $\mathcal{X} \lhd \mathcal{X}'$ *hold, then* $\mathcal{X}$ *is the most specific signature (MSSig) of* w*, and this is denoted by* $\mathcal{X} = MSSig(w)$*.*

Definition 6 states what a signature is and Definition 7 states what 'the most specific (accurate) signature' is for a polymorphic worm. In a network-based approach to generating exploit-based signature, given a sample set R of a polymorphic worm $\mathbf{w}$, we aim to generate *MSSig*(R), which is demonstrated to be NP-hard by Theorem 1. In practice, we could use a polynomial scheme to generate an accurate signature that is close or identical to *MSSig*(R). It is highly possible that this signature is also the most specific signature of $\mathbf{w}$ when the collected R is representative for $\mathbf{w}$ and its size is large enough.

4. A bioinformatics approach to SRE signature generation

In this section, we propose a bioinformatics approach to accurate SRE signature generation for a single polymorphic worm, inspired by some related algorithms from bioinformatics. The kernel of this approach is multiple sequence alignment, which has been broadly studied in bioinformatics where it has been used to find all preserved or common motifs and domains from a set of DNA/RNA sequences (Notredame, 2002). This application is similar to our accurate signature generation problem in which we need to identify invariant bytes from a set of polymorphic worm samples. In the rest of this section, using an example, we first provide an overview of the bioinformatics approach, which is based on a new pairwise sequence alignment algorithm proposed in Section 4.2. The approach comprises three main steps: multiple sequence alignment (Section 4.3), noise elimination (Section 4.4) and signature transformation (Section 4.5).

4.1. *Overview*

We first transform a set of samples (network flows) of a polymorphic worm into a set of character sequences, and then generate an SRE signature for this worm in three steps: multiple sequence alignment, noise elimination, and signature transformation. Fig. 2 illustrates the procedure. 'WormSample1' ("ONEwerTWOtyjfTHREEcxbfd") to 'WormSample6' ("yuiddONEnsddTWOweredsTHREEnfg") are six worm samples and 'noise1' and 'noise2' are two noise samples. The first step analyzes and aligns these worm samples and noise flows. The alignment is represented as a colored matrix, where the greater the number of identical characters in a column, the darker its color. The next step is to identify noise samples using a noise elimination algorithm. Fig. 2 shows sequences 'noise1' and 'noise2' correctly identified as noise flows. The remaining sequences are recognized as worm samples. From them, identical characters in the same columns are extracted as

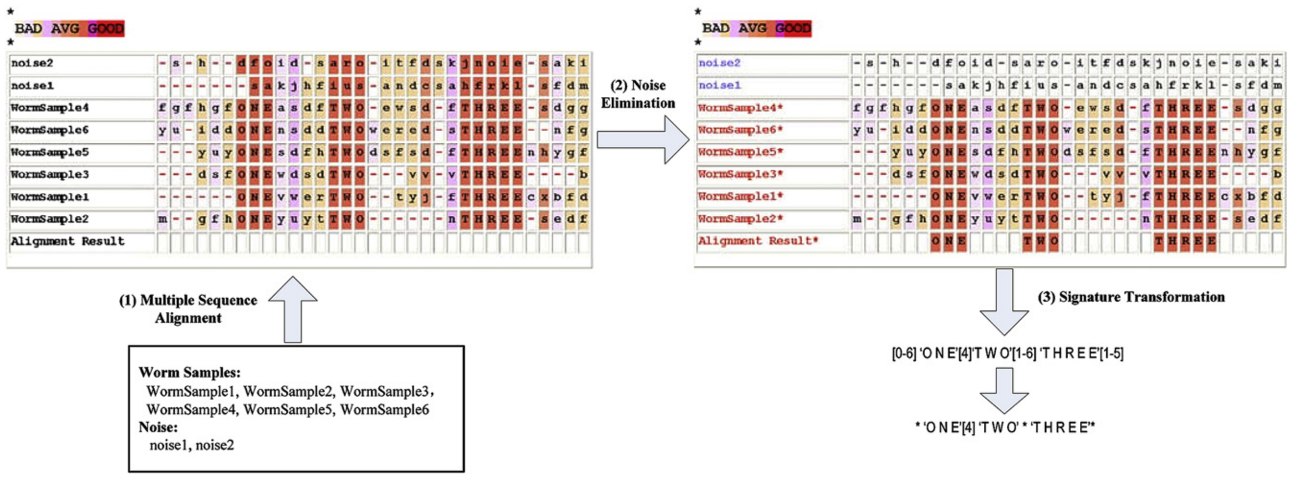

BAD AVG GOOD
(1) Multiple Sequence Alignment
(2) Noise Elimination
(3) Signature Transformation
Worm Samples: WormSample1, WormSample2, WormSample3, WormSample4, WormSample5, WormSample6
Noise: noise1, noise2
Alignment Result
[0-6]'O N E'[4]'T W O'[1-6]'T H R E E'[1-5]
'O N E'[4]'T W O''T H R E E'*

Fig. 2 – Three steps in the bioinformatics approach.

invariant bytes of the polymorphic worm. Step 3 produces an SRE signature "*'ONE'[4]'TWO'*'THREE'*" by putting distance restrictions between adjacent invariant bytes. This is the most specific signature of the worm.

4.1.1. *Step 1. Multiple sequence alignment*

Sequence alignment compares pair or multiple sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences. The definition of multiple sequence alignment is given in Definition 8. An alignment (result) is represented as a matrix $\mathcal{A}$ and the row A_p within the matrix represents the aligned sequence of s_p. Gaps ('–') are inserted between elements so that elements with identical characters from different sequences can be aligned in the same columns. As can be seen in Fig. 2, we use colors to indicate how many rows containing an identical character for each column of $\mathcal{A}$. Columns with more rows are filled with a darker color.

Definition 8. (*Multiple Sequence Alignment.*) *Given a family of sequences* $\mathcal{S} = \{s_1, \ldots, s_k\}$ *over an alphabet* Σ, $|s_p|$ *is the length of the sequence* s_p, *and an alignment of sequences in* $\mathcal{S}$ *is a* $(k \times N)$*-matrix* $\mathcal{A} = (\mathcal{A}_{p,i})_{1\le p\le k, 1\le i\le N}$ *with* $\max_{1\le p\le k}|s_p| \le N \le \sum_{1\le p\le k}|s_p|$, *if and only if:*

1. $A_{p,\,i} \in \Sigma \cup \{\text{'–'}\}$ (where '–' $\notin \Sigma$ is called a gap);
2. Upon removal of all blanks, row $A_p = (A_{p,\,i})_{1\le i\le N}$ reduces to S_p; and
3. No column consists only of blanks.

Multiple sequence alignment (MSA) is the core of our approach. Various MSA techniques have been proposed in the area of bioinformatics, including *Exact alignment*, *Progressive alignment* (Notredame et al., 2000; Thompson et al., 1994), and *Iterative alignment* (Notredame, 2002). Progressive alignment is by far the most widely used and uses pairwise alignment with sequences being aligned one by one. We propose our MSA algorithm in Section 4.3, which is based on the adopted T-Coffee algorithm and a new pairwise alignment algorithm CSR proposed by us (see Section 4.2).

4.1.2. *Step 2. Noise elimination*

Owing to imperfect suspicious flow classification or worm sample clustering, there may be noise in worm samples. This step finds the noise and removes it. As can be seen in Fig. 2 (step 2), sequences (rows in matrix) identified as noise do not have '*' at the end of the sequence name, but the sequences identified as worm samples do. The noise elimination algorithm can be found in Section 4.4.

4.1.3. *Step 3. Signature transformation*

This step outputs the final SRE signature transformed from the MSA alignment result. The transformation technique will be discussed in Section 4.5.

4.2. *Pairwise sequence alignment algorithm*

A pairwise sequence alignment is a matrix where one sequence is placed above the other to find and align common characters. Gaps ('–') are inserted to help in aligning matching characters. A mismatch occurs if elements in the same column are not identical. Fig. 3 shows results for pairwise sequence alignment between "oxnxexzxtwox" and "ytwoyoynyeyz". In the following, we first describe limitations of the well-known

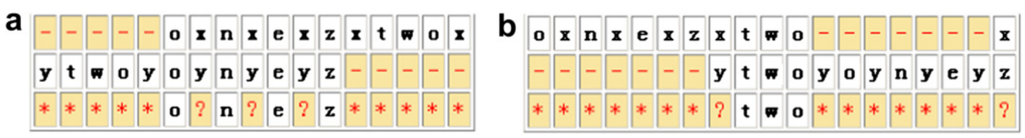


Fig. 3 – Two alignments generate different results.

Needleman–Wunsch pairwise alignment algorithm on signature generation, and then propose a new pairwise alignment algorithm – an improved CSR algorithm from (Tang et al., 2007). This algorithm is the basis of the multiple sequence alignment algorithm in our approach.

4.2.1. *Limitations of the Needleman–Wunsch pairwise alignment algorithm*

The Needleman–Wunsch algorithm (Needleman and Wunsch, 1970) is a typical global alignment algorithm that can compute an alignment between two sequences by maximizing a similarity score as in Formula (1), where k_m denotes the number of matches, W_m denotes the score for a match, k_d denotes the number of mismatches, W_d denotes the score for a mismatch, k_{gaps} denotes the number of gaps, and δ denotes the penalty score for a gap. For example, if we set $W_m = 1$, $W_d = 0$, $\delta = -1$, the similarity score of the alignment in Fig. 3(a) is $-6(4\times1+3\times0+10\times(-1))$, the similarity score of the alignment in Fig. 3(b) is -11 $(3\times1+2\times0+14\times(-1))$.

$$SC(x,y) = k_m \times W_m + k_d \times W_d + k_g \times \delta \qquad (1)$$

Although the Needleman–Wunsch algorithm can maximize the score in Formula (1), it is not suitable for polymorphic worm signature generation because it maximizes only the total number of matches but does not reward consecutive matches, yet most invariant parts in polymorphic worms are composed of consecutive characters, and consecutive characters in a signature, in turn, can improve the accuracy and efficiency of worm traffic detection. Consider the example in Fig. 3 where two strings "oxnxexzxtwox" and "ytwoyoynyeyz" are aligned in two ways. Fig. 3(a) is the alignment generated by the Needleman–Wunsch algorithm with a similarity score of -6 while Fig. 3(b) is another possible alignment with a similarity score of -11. Although the similarity score of the latter alignment is smaller than the one of the former, it contains a contiguous substring 'two', which is semantically meaningful. Thus it is more likely to be an invariant part of a polymorphic worm.

4.2.2. *CSR pairwise alignment algorithm*

Over many experiments we have discovered that no matter how the parameters were adjusted, limited to the optimization function in Formula (1) that maximizes the total number of matches instead of consecutive matches, the Needleman–Wunsch algorithm always failed to generate contiguous substrings that are much meaningful in worm traffic detection. Our proposed pairwise sequence alignment algorithm, CSR (consecutive substrings rewarded)[2] extends Needleman–Wunsch by rewarding consecutive matches, that is, more contiguous substrings in an alignment result. We do this by modifying the similarity score function from Formulas (1)–(2). In Formula (2), an additional function *enc*() is used for rewarding contiguous substrings. That is, for a possible alignment result, each contiguous substring will result in an additional (rewarding) score and the longer the substring is, the higher the additional score will be.

[2] See Algorithm 2. in Appendix A for more details.

$$SC(x,y) = k_m \times W_m + k_d \times W_d + k_g \times \delta + \sum_{\substack{\text{S is a substring}\\ \text{in analignment result}}} enc(|s|) \qquad (2)$$

For example, given that $enc(x) = 3(x-1)$ and $W_m = 1$, $W_d = 0$, $\delta = -1$, the similarity score of the alignment in Fig. 3(a) remains to be -6 $(1\times4+0\times3+(-1)\times10+3\times(1-1))$ while it is -5 in Fig. 3(b) $(1\times3+0\times2+(-1)\times14+3\times(3-1))$. Our CSR algorithm outputs the optimized alignment as shown in Fig. 3(b).

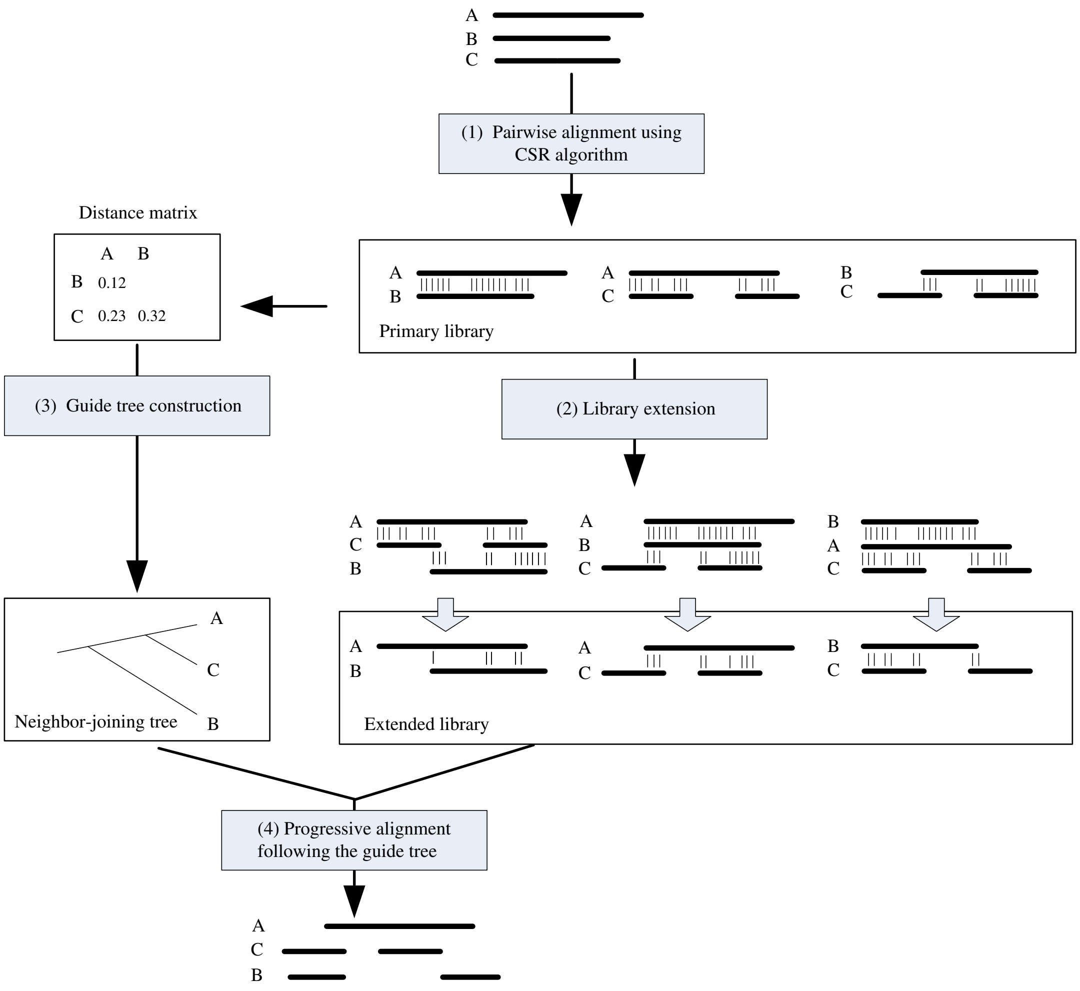
4.3. *Multiple sequence alignment algorithm*

The multiple sequence alignment algorithm in our approach derives from the T-Coffee algorithm (Notredame et al., 2000) but differs in that for the pairwise alignment applied in building a primary library, we use our CSR algorithm while T-Coffee uses the Needleman–Wunsch algorithm. A significant advantage of our approach is that our algorithm accepts any kind of sequences whereas T-Coffee accepts only limited data types, such as DNA, RNA, and protein sequences. Fig. 4 shows the four stages of our MSA algorithm, describes as follows.

1) **Pairwise alignment to build primary library.** The first stage will build a primary library by aligning every two sequences from all input sequences using our CSR algorithm. If there are N sequences to be aligned, the primary library contains alignment results of $N(N-1)/2$ sequence pairs. Each alignment in the primary library is represented as a list of pairwise element matches. For example, the pairwise alignment shown in Fig. 3(b) can be represented as "A.9 with B.2" (the 9th element of sequence A is aligned with the 2nd element of sequence B), "A.10 with B.3" and "A.11 with B.4".
2) **Library extension.** This stage extends each pairwise alignment to a triplet. For example, as shown in Fig. 4, the pairwise alignment of sequence A and sequence B is extended to the alignment between A and B through sequence C. Each pairwise element match in the extended library is given a weighted score. For example, if "A.10 with C.11" and "C.11 with B.2" are listed, then the weight score of "A.10 with B.2" will increase by 1.

Library extension is the main technique applied in the T-Coffee algorithm. The weighted element match in each pairwise alignment reflects its popularity in the whole library. The benchmark simulation results in (Notredame et al., 2000) showed that T-Coffee algorithm outperformed other algorithms like Prrp, DiAlign, and ClustalW (Thompson et al., 1994). Our experiments in Section 5.5 demonstrate similar results that library extension can improve the generated signature accuracy.

3) **Guide tree construction.** Because each pairwise alignment can output a similarity score according to Formula (2), the pairwise alignments in stage 1 can be used to produce a distance matrix in which a smaller distance represents more matches between two close sequences. We can use



A
B
C
(1) Pairwise alignment using CSR algorithm
Distance matrix
A B
B 0.12
C 0.23 0.32
Primary library
(3) Guide tree construction
(2) Library extension
Neighbor-joining tree
Extended library
(4) Progressive alignment following the guide tree

Fig. 4 – The four stages in the progressive multiple sequence alignment algorithm.

this matrix to construct a binary guide tree using the neighbor-joining method (Saitou and Nei, 1987). The tree guides the progressive alignment in which a close sequence is iteratively added to get the final multiple alignment.

4) **Progressive alignment.** The multiple sequence alignments are progressively assembled using a series of pairwise alignments and following the branch order in the guide tree. Similar to ClustalW (Thompson et al., 1994) and T-Coffee (Notredame et al., 2000), the alignment is carried out by a dynamic programming algorithm using the information in the extended library (the pairs of elements and their weights). The progressive alignment in Fig. 4 first aligns A and C, then goes with B.

4.4. *Noise elimination*

Imperfect suspicious flow classification or worm sample clustering can produce noise in worm samples which must be eliminated to derive accurate worm signatures. Since MSA in step 1 generates maximum element matches from each pairwise alignment, a limited number of noise flows will not influence the extraction of most common characters. That is, most worm samples can be aligned to get valuable matches even if there are a few noise flows. There are two noise samples at Step 1 in Fig. 2, yet the six worm samples are properly aligned to show extractable common invariant parts.

The proposed noise elimination algorithm in Algorithm 1 is designed to improve the accuracy of signature generation in a noise-tolerant way. Given the alignment of k sequences, we define a noise tolerance rate θ and select $\lceil k\cdot\theta\lceil$ sequences as noise within k sequences. The remaining $k-\lceil k\cdot\theta\lceil$ sequences will be regarded as worm samples and we will use them to output the final SRE signature. The noise elimination algorithm first determines the invariant bytes of a polymorphic worm. The invariant bytes will be characters that appear more than $\lceil k\cdot\theta\lceil$ times in one column. Then we determine $\lceil k\cdot\theta\lceil$ noise samples as those containing fewer invariant bytes.

```
Algorithm 1. (Noise elimination algorithm.)
input: $\mathcal{A}$: a $(k\times N)$ matrix, which is the alignment of $k$ sequences in $\mathcal{S}=(s_1, \ldots, s_k)$;
output: $\mathcal{N}$: a set of noise, $\mathcal{N}\subseteq\mathcal{S}$;
parameters: $\theta$: noise tolerance rate $(0\le\theta\le1)$;
Determine invariant bytes
    foreach $i\in\{1, \ldots, N\}$ do
```

```
  foreach $c \in \Sigma$ do
   $n_{i,c} \leftarrow$ the count of elements in column $i$ whose value is $c$;
   $f_{i,c} \leftarrow n_{i,c}/k$;
  end
  end
  foreach $i \in \{1, \ldots, N\}$ do
   $F_i \leftarrow \max_{c \in \Sigma} f_{i,c}$;
   if $F_i \geq 1 - \theta$ then
     $I_i \leftarrow c$, where $c$ is the character to make $F_i = f_{i,c}$; /* $I_i$ is an
     invariant byte */
     else $I_i$=' '; /* $I_i$ is not an invariant byte */
   end
```

Determine noise

```
  foreach $s_i \in \mathcal{S}$ do
/* count how many invariant bytes are included in this
 sequence */;
  $InvByt(s_i) \leftarrow$ the count of $A_{i,j} (1 \leq j \leq N)$ such that $A_{i,j} = I_j$;
  end
  $\mathcal{N} \leftarrow \emptyset$;
/* select $\lceil k \cdot \theta \lceil$ sequences as noise which contain less invariant
 bytes */;
  for $i = 1$ to $\lceil k \cdot \theta \lceil$ do
   $s_i$ is the sequence such that $s_i \in \mathcal{S}/\mathcal{N}$, and for any other
   $s' \in \mathcal{S}/\mathcal{N}$ we have $InvByt(s_i) \leq InvByt(s')$;
   $\mathcal{N} \leftarrow \mathcal{N} \cup \{s_i\}$
  end
  return $\mathcal{N}$
```

The selection of the noise tolerance rate θ is a key point in the noise elimination algorithm. A fixed value of θ could be impractical. If θ is too small, some noise may not be filtered out. In contrast, if θ is too large, some worm samples may be wrongly eliminated as noise and there will be not enough worm samples left.[3] In both cases, our approach may fail to generate accurate signatures. We propose to use an adaptive θ scheme that θ changes its value according to the total number of available sequences (k, and k is required to be over 4), as shown in Formula (3). The objective of this scheme is to ensure that for any k ($k > 4$), $k\theta \approx \lceil 0.8k - 3.2 \lceil$ sequences will be chosen as noise, $k - k\theta = k(1 - \theta) \approx 4 + \lfloor 20\%(k - 4) \rfloor$ sequences will be chosen as worm samples. In other words, at least 4 sequences will be chosen as samples. For the remaining $k - 4$ sequences we choose 20% as samples (80% as noise), a conservative policy.

$$\theta = 0.8 - 3.2/k (k > 4) \tag{3}$$

4.5. *SRE signature transformation from alignment result*

An alignment result can be easily transformed into an SRE signature. Given an alignment of multiple worm samples, after the noise elimination and invariant byte extraction, we can get the distance restriction for adjacent invariant bytes by counting the number of in-between none-blank positions. Taking Fig. 2 as an example, the corresponding SRE signature is ‘‘[0,6]‘ONE’[4]‘TWO’[1,6] ‘THREE’[1,5]’’, where ‘[1,6]’ is a distance bound restriction meaning that there are at least one and at most six elements between ‘TWO’ and ‘THREE’ in worm samples (WormSample1–WormSample6).

Although using ‘[k1, k2]’ to express a distance bound restriction (instead of ‘*’ in SRE signature) is a more precise way to express the range distance restriction, this paper does not adopt such a distance bound restriction in generated signatures. This is because even though there are some range distance restrictions in polymorphic worms that can be exactly expressed by a bound of ‘[k1, k2]’, we may not be able to extract a perfect bound given inadequate worm samples. For instance, suppose that a polymorphic worm should have a real range distance restriction with the bound of ‘[1, 100]’. If the worm samples are inadequate, as a result, we may only get an over-specific bound, like ‘[12,50]’, which will result in a high false negative rate. The last step in our approach outputs SRE signatures by extending each range bound of ‘[k1, k2]’ to ‘*’, i.e., extending ‘‘[0,6]‘ONE’[4]‘TWO’[1,6]‘THREE’[1,5]’’ to ‘‘*‘ONE’[4]‘TWO’* ‘THREE’*’’, as in Fig. 2. Note that we still keep the fixed distance restriction in generated SRE signatures (like [4] in Fig. 2).

5. Experiments

In this section, we evaluate our bioinformatics approach under different scenarios. After introducing metrics used to measure signature quality, we first evaluate our approach using synthetic polymorphic worms and compare it with previous approaches, like Polygraph (Newsome et al., 2005) and Hamsa (Li et al., 2006). Then, we show the signature generation quality for polymorphic shellcodes, which were constructed by some real-world shellcode engines. We also carried out experiments to test the noise-tolerance ability of our approach where there were noise in worm samples. We also compare some well-known multiple sequence alignment (MSA) algorithms with our approach (CSR + T-Coffee) in terms of signature generation accuracy. Our signature generation approach was implemented in VC6.0 with about 7000 lines of C++ code. All experiments were run on a desktop machine with 2.0 GHz Intel Core 2T7200 processor, running Windows XP SP2.

5.1. *Evaluation metrics*

We measure the quality of a generated signature for variant approaches in terms of: (1) the false positive (FP) in normal traffic and false negative (FN) in tested polymorphic worm samples. (2) the *invariant-byte deviation* and *distance-restriction deviation* from the most specific signature. The measurement details are described below.

5.1.1. *FP and FN*

To test the FP of a generated signature in our experiments, we used a normal traffic pool, in which the data was collected from two sources: (1) 24GB network traffic at the campus network gateway of Hong Hong Polytechnic University, including 5-day HTTP traces (45,111 flows,

[3] The worm sample size should be larger than 4 for accurate signature generation as shown in experiments in Section 5.4.

Table 4 – The synthetic polymorphic worms for evaluation.

Polymorphed worm name	TSIG	IISPrinter	CodeRedII	Witty	Zotob	ISAKMP
Real-world worm/malware	Lion worm	Beavuh malware	Code Red II worm	Witty worm	Zotob worm	N/A
Bugtraq ID	17692	2674	2880	9913	14513	9416
CVE ID	CVE-2006-2073	CVE-2001-0241	CVE-2001-0500	CVE-2004-0362	CVE-2005-1983	CVE-2004-0164
Target system	BIND DNS server	ATPhttpd 0.4	Microsoft IIS 5.0	RealSecure Server Sensor	PnP service	Check Point Software Firewall
(total length of substrings, total length of distance restrictions)	(6,216)	(39,0)	(31,11)	(12,15)	(20,82)	(4,28)

15 GB) and 3-day DNS traces (1.5 GB); and (2) the first week's network TCP dump data from 1999 DARPA Intrusion Detection Evaluation Data Sets (Lippmann et al., 2000) (10 GB), which is guaranteed no attack traffic. To test FN of a generated signature for a polymorphic worm, we obtained FN result by detecting another 5000 synthetic samples of the same worm.

5.1.2. *Invariant-byte deviation and distance-restriction deviation*

Ideally the generated signature should be the most specific for a polymorphic worm. In our evaluation experiments, if we know the most specific signatures of a polymorphic worm in advance, we can quantify the quality of a generated signature by measuring how far it deviates from the most specific one, represented as *the invariant-byte deviation* and *the distance-restriction deviation.*

Given that $\mathcal{X}$ is a generated signature and $\mathcal{Y}$ is the most specific signature of a polymorphic worm w, we use $STR(\mathcal{X})$ and $STR(\mathcal{Y})$ to denote strings converted from $\mathcal{X}$ and $\mathcal{Y}$ respectively by removing all qualifiers. Obviously, we can find a set of operations that convert $STR(\mathcal{X})$ to $STR(\mathcal{Y})$ and minimize the total operation cost. The operations can be character insertion, deletion and replacement. The minimum operation cost is commonly known as *the Levenshtein distance* (Levenshtein, 1966). Here we call it *the invariant-byte deviation* between $\mathcal{X}$ and $\mathcal{Y}$. For example, given $\mathcal{X} = ''c*aa*bb*''$ and $\mathcal{Y} = ''[3]aa*bbb*''$, their invariant-byte deviation is 2, since $STR(\mathcal{X}) = ''caabb''$ can be converted to $STR(\mathcal{Y}) = ''aabbb''$ by two operations that deleting the 'c' at the start and adding a 'b' to the end of $STR(\mathcal{X})$. The *distance-restriction deviation* states the absolute value of difference between the distance restrictions in $\mathcal{X}$ and $\mathcal{Y}$, that is, $|(||\mathcal{X}|| - |\mathcal{X}|) - (||\mathcal{Y}|| - |\mathcal{Y}|)|$. For example, given $\mathcal{X} = ''c*aa*bb*''$ and $\mathcal{Y} = ''[3]aa*bbb*''$, the distance restriction deviation is $|(5-5)-(8-5)| = 3$.

5.2. *Signature generation for synthetic polymorphic worms*

In this section, we first illustrate synthetic polymorphic worms whose samples without noise are used to evaluate the generated signature accuracy of our approach. Though a few false positives are produced, the generated signatures are very accurate in that they are all specific. The accuracy is also related to the number of worm samples investigated. We also compare the accuracy of our approach with others.

5.2.1. *Synthetic polymorphic worm*

Although some polymorphic engines have been released in recent years, there are no well-known polymorphic worm traffic reported on the Internet. Most NSG-based detection methods (like Polygraph Newsome et al., 2005 and Hamsa Li et al., 2006) evaluated their performance using synthetically generated polymorphic worms based on real-world exploits. In the first series of experiments, we follow their approach using synthetic polymorphic worms to evaluate our approach. We choose six real-world exploits and develop some Python scripts to make them polymorphic. Thus we can create arbitrary number of synthetic polymorphic worm samples for signature generation and FN testing. Among the six synthetic polymorphic worms, Beavuh can launch real-world malware exploits, four others (Lion, Code Red II, Witty and Zotob) can launch real-world worm exploits, and ISAKMP can develop particularly selected exploits whose invariant parts contain only one or two bytes.

Table 4 summarizes the six synthetic polymorphic worms, the first row giving their names and the second row indicating the real-world worms/malwares that make use of the same exploit. The rows entitled "Bugtraq ID" and "CVE ID" show the corresponding vulnerabilities exploited by these worms and the "Target system" row exhibits the targeted vulnerable systems. The last row gives the real total length of substrings

Table 5 – Generated signatures for five worm samples and their tested false positives and false negatives.

Worm (average sample length)	Signature	Run time (seconds)	FP	FN
TSIG (569)	*'\xFF\xBF'[200]'\x00'[14]'\x00\x00\xFA'[2]	18.7	0	0
IISprinter (460)	'GET http://'* ': '* '\r\n'* 'null.printer?'*' HTTP/1.0\r\n'	10.78	0	0
CodeRedII (309)	'GET/'*'.ida?'*'XX'[15]'%u'*'%u780'* '='[7]'HTTP/1.0\r\n'	4.7	0	0
Witty (109)	'\x05\x00'[5]'\x12\x02'*'\x05\x00'[5]'n\x00'*'\x05\x00'[5]'\xDE\x03'*	0.45	0	0
Zotob (253)	'\x00'[3]'\FFSMB%'[56]'&\x00'[4]'\PIPE\\x00\x05\x00\x00'[19]'6\x00'*	2.56	0	0
ISAKMP (77)	*'\x08'[11]'\x0c'[1]'\x00\x04'*	0.22	0.00735%	0

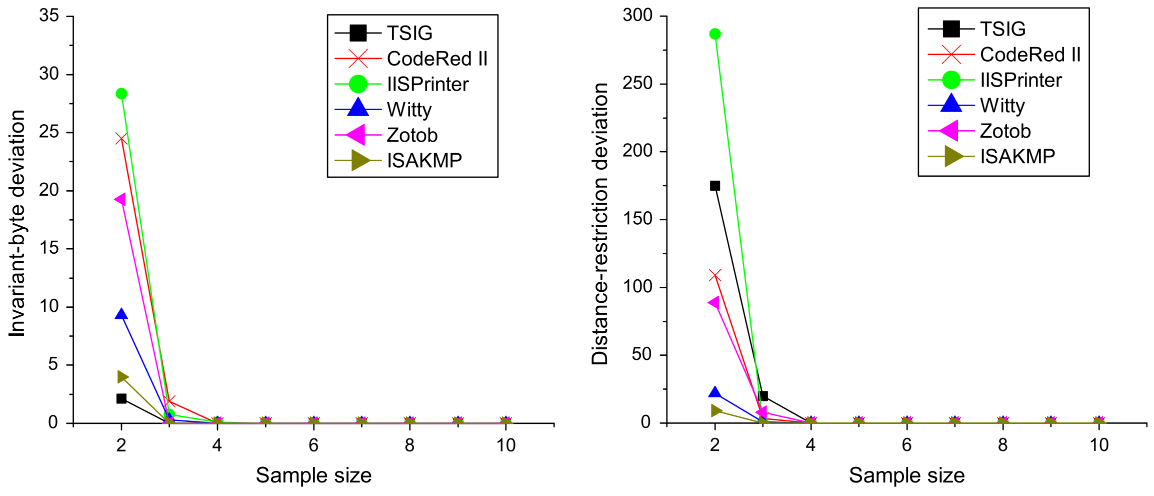


Fig. 5 – The average invariant-byte deviation and average distance-restriction deviation of the generated signatures given different numbers of samples.

and total length of distance restrictions for the most specific signature of each polymorphic worm.

5.2.2. *Signature quality*

We tested our signature generation approach for each synthetic polymorphic worm without adding noise. The generated signatures from five randomly selected worm samples are shown in Table 5. We observe that they are exactly the same as the most specific signatures as we know in advance. These (the most specific) signatures consequently lead to almost zero false positives on normal traffic, and zero false negative on tested polymorphic worm samples. Note that our approach is robust enough to generate the most specific signature of ISAKMP worm, which contains two 1-byte and one 2-byte invariant parts. This suggests that although our approach (CSR algorithm) rewards contiguous (long) substring extraction, it is also effective in capturing short substrings, even some 1-byte or 2-byte invariant parts. The FP of the signature for ISAKMP worm is not zero, since it matches multimedia (.rm) files on HTTP flows five times, however these flows are not ISAKMP worm.

5.2.3. *Number of worm samples needed*

To determine how many worm samples are needed to generate an accurate signature, we tested our signature generation algorithm with samples varied from 2 to 10. Each test was conducted ten times and we computed the average invariant-byte deviation and the average distance-restriction deviation, which are shown in Fig. 5. We can see that a sample containing more than four worms has an average invariant-byte deviation and average distance-restriction deviation of zero, meaning that no invariant bytes or distance restrictions were lost in the generated signatures and that they are maximally specific.

5.2.4. *Signature accuracy comparison*

Table 6 compares our approach with most recent exploit-based and network-based signature generation methods, testing the polymorphic Code Red II worm. Our approach can extract more valuable information (such as 1-byte invariant parts and distance restrictions) from worm exploits, improve signature accuracy and can potentially benefit current IDSs for prompt worm detection.

5.3. *Signature generation for real-world polymorphic shellcodes*

In this series of experiments, we used some popular real-world polymorphic engines (Metasploit Team, 2007, CLET DeTristan et al., 2003 and Admmutate Ktwo, 2001) to create polymorphic shellcodes by taking a single shellcode sample and encrypting it with each engine 1000 times to generate

Table 6 – Accuracy of signatures for Red Code II worm.

Signature type (approach)	Generated signature	Limitations in accuracy
Single substring (earlier systems (Kreibich and Crowcroft, 2003; Kim and Karp, 2004; Wang et al., 2003; Singh et al., 2004)	‘.ida?’ or ‘%u7801’	lost of most invariant parts and the distance restrictions of invariant parts
Token-subsequence (Polygraph Newsome et al., 2005)	GET /.*.ida? .*XX.*%u.*%u7801.*HTTP /1.0\r\n	lost of “=” and the distance restrictions of invariant parts
Multi-sets of tokens (Hamsa Li et al., 2006)	{‘.ida? ’: 1, ‘%u780’: 1,‘HTTP/1.0\r\n’:1, ‘GET/’: 1,‘%u’: 2}	lost of “=”, the order and the distance restrictions of invariant parts
SRE signature (Ours)	‘GET/’*‘.ida? ’*‘XX’[15]‘%u’*‘%u780’* ‘=’[7]‘HTTP/1.0\r\n’	none

Table 7 – Signatures generated for 10 shellcode sequences from each polymorphic engine.

Shellcode (polymorphic engine)	Generated signature	FP	FN
shell_bind_tcp[a] (Metasploit)	*‘\x89’[3]‘\xd9’[1]‘\xf4’*‘CCCCCC7’[2]‘jAXP0A0AkAAQ2AB2BB0BBABXP8ABuJI’ [8]‘KOKOKL’[2]‘JKPM’[4] ‘KOKOKO’[2]‘LKBL’[4]‘LK’[2]‘GLLKCL’[6]‘JOLKPO’[2]‘LKQO’ [4]‘JK’[2]‘LK’[2]‘LK’[2]‘JN’[2]‘IP’[2]‘NL’[2]‘IP’[4]‘IQ’[2]‘DM’[4]‘JK’[2]‘GK’[10]‘LKQO’[4]‘JK’[2]‘LKDLPKLKQOEL’ [2]‘JK’[2]‘FLLK’[2]‘BL’[2]‘EL’[6]‘IK’[2]‘LK’[4]‘LK’[2]‘DLLK’[2]‘ELNMLK’[4]‘QN’ [2]‘LNPNDNJL’[2]‘KO’[22]‘QO ’[2]‘KO’[4]‘HKJMKLGK’[2]‘KO’[2]‘QO’[8]‘JM’[10]‘KO’[10]‘NM’[2]‘KO’[20]‘KO’ [8]‘QL’[22]‘KO’[10]‘- KO’[6]‘NMFN’[4]‘KO’[6]‘KO’[14]‘KO’[10]‘KO’[16]‘KO’[4]‘KO’[14]‘BM’[12]‘HL’[14]‘IP’[2]‘NJKN’ [22]‘FMKN’[2]‘FL’[2]‘LM’[4]‘NKNKNK’[4]‘KN’[4]‘KO’[4]‘KO’[2]‘QK’[22]‘KO’[4]‘NM’[4]‘HN’[2]‘KO’[4]‘KOKO’ [2]‘KO’[2]‘LK’[2]‘KL’[6]‘KO’[4]‘KO’[4]‘CN’[8]‘KO’[2]‘KO’[2]‘AA\x0d\x0a’	0	0
exec_add_user[b] (Clet)	[600]‘\xeb’[2]‘1’[2]‘\xf0\x8b’*‘\xff\xff\xff’ *‘t\x07\xeb’[1]‘\xe8’[1]‘\xff\xff\xff’* ‘\xfe\x01\x1e\xc9\xfe \x01\x1e\xc9\xfe\x01\x1e\xc9\xfe\x01\x1e\xc9’	0	0
exec_bind_tcp[c] (Admmutate)	*‘\xe8’[1]‘\xff\xff\xff’[344]‘s\x90\xf0\xd1\xbf\xbf \xf0\xd1\xbf\xbf’	0.00147%	0

a A shellcode that sets up a socket, binding to a specific port and listening for a connection; Upon accepting a connection, it spawns a new shell.
b A shellcode that adds a user account in a remote system.
c A shellcode that listens for a connection and allows us to remotely executes an arbitrary command.

1000 unique shellcode sequences. Table 7 shows the signatures generated by our approach for 10 shellcode sequences from each engine. The results also demonstrate that the signatures generated are able to identify normal traffic for near-zero FP (there is only one FP for wrongly classifying a multimedia file as a worm flow), and detect other 990 shellcode sequences for zero FN perfectly.

5.4. *Signature generation with noise*

Next we show that our approach can generate accurate signatures even if the input sequences contain noise. Firstly, we randomly selected 6 worm samples from the Code Red II worm and created noise sample strings with length of 500 bytes (so-called low-similarity noise). Each test has been run 5 times to get an average result. Fig. 6(a) shows the generated signature quality. When the number of noise samples is small (fewer than 9, the noise ratio is less than 57%), the generated signatures are very accurate to be the most specific one. When the number of noise samples increases from 9 to 13, the generated signatures are less accurate because a few invariant bytes could be lost. When the number of noise samples is large (larger than 13), the generated signatures become very inaccurate. The sudden increase appears in Fig. 6(a) because most invariant bytes and distance restrictions are lost as a result of too many noise samples wrongly identified as worm samples.

Secondly, we used 6 samples from Code Red II worm while noise samples were HTTP request flows randomly collected from normal traffic (so-called high-similarity noise). As Fig. 6(b) shows, when there are fewer than four noise samples (a noise ratio of 40%), the signatures that are generated are most specific. However, when there are more than four samples, the average invariant-byte deviation increases dramatically. This

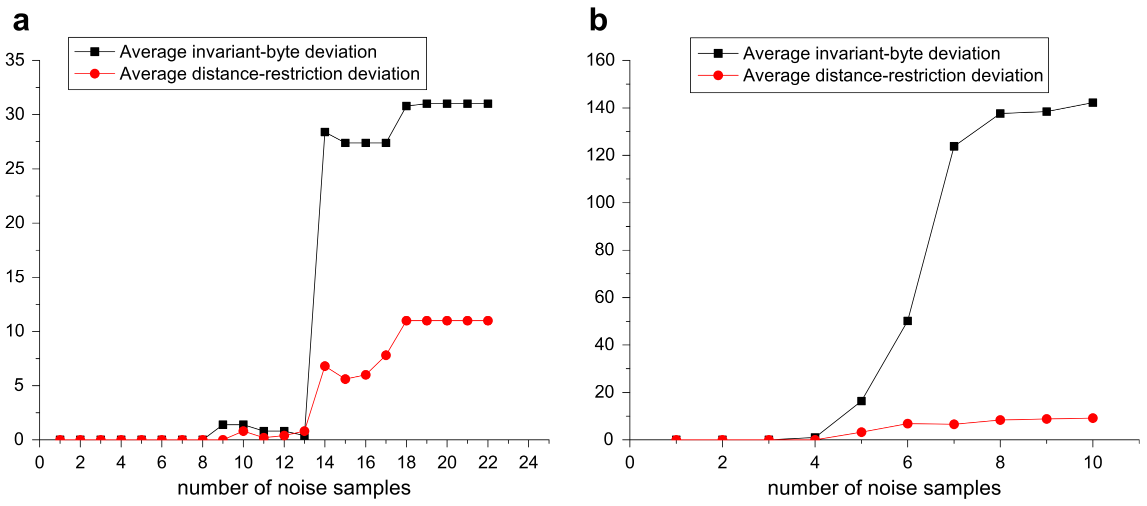


Fig. 6 – The average invariant-byte deviation and distance-restriction deviation of the generated signatures for variant noise samples.

is because some common bytes or keywords, frequently witnessed in HTTP request flows, like ''mozilla/'', are incorrectly extracted as invariant bytes and thus our approach tends to classify those flows as worm samples.

5.5. *MSA algorithm comparison*

We also compare our Multiple Sequence Alignment (MSA) algorithm (CSR + T-Coffee algorithm) with others in experiments to illustrate the impact of MSA on the signature accuracy. Our approach uses CSR algorithm, instead of traditional Needleman–Wunsch (NW) algorithm, for the pairwise alignment, and adopts T-coffee for the progressive MSA. ClustalW (Thompson et al., 1994) is an alternative MSA algorithm. We carried out comparison experiments by replacing the MSA algorithm in step 1 of our approach with other possible combinations. There are four MSA algorithm combinations with different pairwise and MSA algorithms: NW + ClustalW, NW + T-Coffee, CSR + ClustalW, and CSR + T-Coffee. As in the setting described in Section 5.2, we tested the signature quality generated for each algorithm combination using samples of different sizes and worm samples from Code Red II, Witty, and IISPrinter. As Fig. 7 shows, the CSR + T-Coffee algorithm (adopted in our approach) yields the best performance, the only one to generate accurate signatures without losing any invariant bytes and distance restrictions when the number of worm samples is larger than 4.

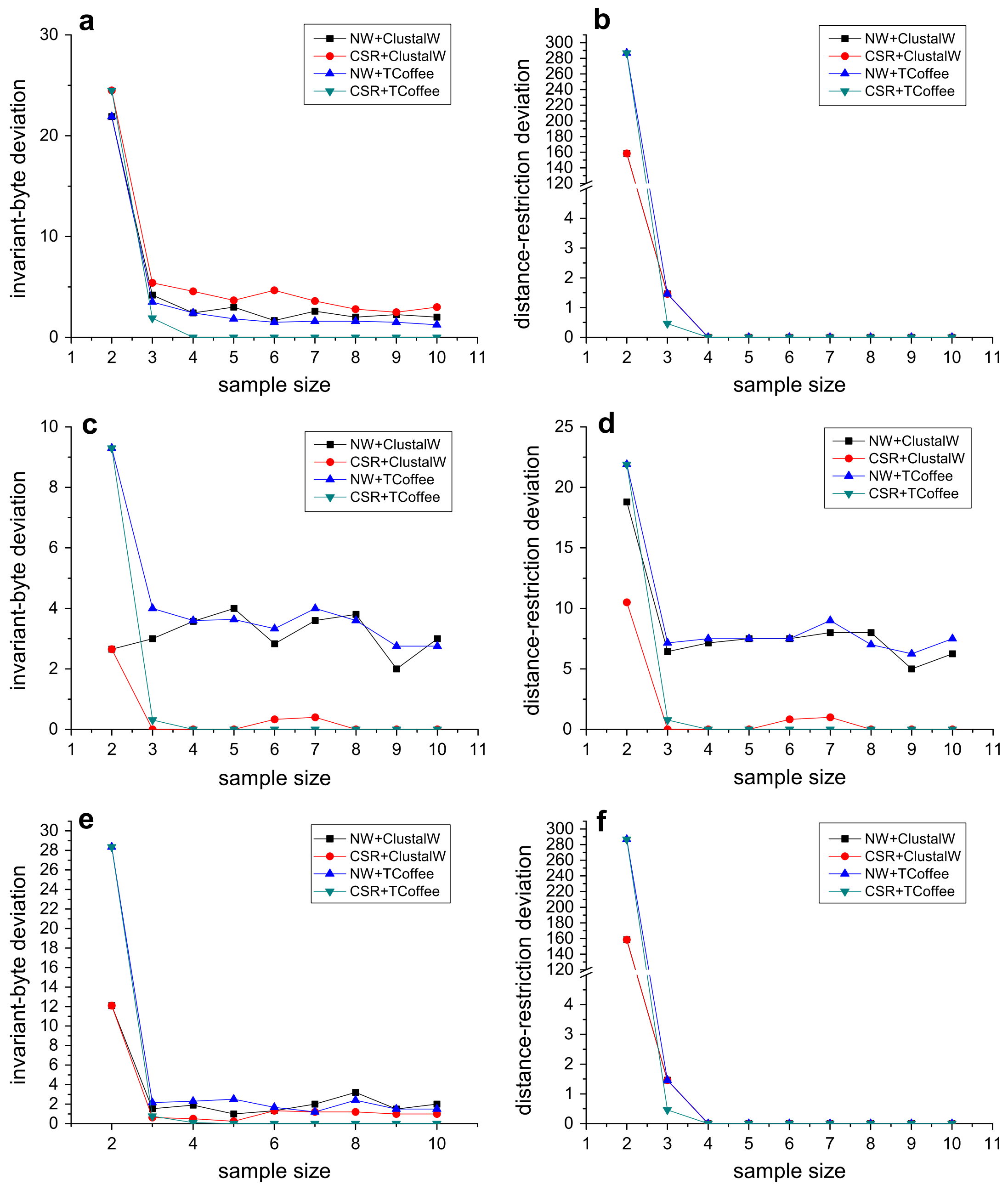


Fig. 7 – Performance of different MSA combinations.

6. Discussion

6.1. Complexity analysis

Suppose that there are N sequences of length L as the input to our signature generation approach. The time complexity of the CSR algorithm is $O(L^2)$ and the time complexity of multiple sequence alignment is $O(N^2L^2)+O(N^3L)+O(N^3)+O(NL^2)$, where $O(N^2L^2)$ represents the computation involved in the pairwise alignment to build primary library, $O(N^3L)$ for library extension, $O(N^3)$ for guide tree construction and $O(NL^2)$ for the computation of the progressive alignment. Since the time complexity of noise elimination in step 2 and the signature transformation in step 3 is trivial (respectively $O(N)$) and normally $L \gg N$ in practice, the total time complexity of our signature generation approach is $O(N^2L^2)+O(NL^2)$. Table 5 shows the run time of signature generation with 5 worm simples from each polymorphic worm.

6.2. Scope and limitations

Our approach generates exploit-based SRE signatures. It can work well for the early stage detection of zero-day exploits before new polymorphic worms widely propagate on the Internet, since it can be generated when only a small number of samples but not other information are available. Though our approach may fail for some worms if they do not have any artifacts (i.e. invariant bytes) (Crandall et al., 2005) and hence can not be described by any exploit-based signatures, the approach is effective for a significant portion of the state-of-the-art polymorphic worms and exploits in the wild.

In this paper, we assume that worm samples are from one polymorphic worm or they are from multiple polymorphic worms but have been well clustered. This assumption is practical since a number of worm sample clustering methods have been proposed, such as the hierarchical clustering method in (Newsome et al., 2005), the online star clustering algorithm in (Yegneswaran et al., 2005), and the incremental online clustering algorithm in (Wang et al., 2003). Even if the clustering technique is not perfect, a few samples from other polymorphic worms can be regarded as noise and this will not affect the final SRE signature generation because the proposed approach is noise-tolerant to filter out a small part of noise.

Our approach is subject to the limitations of all exploit-based signature generation approaches, that is: (a) the generated signatures cannot detect all potential exploits of a vulnerability, (b) the worm traffic detection effect could be mitigated due to making a significant number of mistakes of both false positives and false negatives, as a result of specially designed adversary attacks (Perdisci et al., 2006; Newsome et al., 2006). In fact, all learning-based signature generation approaches (including some vulnerability-based signature generation approaches) could suffer from these kinds of attacks (Venkataraman et al., 2008). Fortunately, no such attacks in the wild have been reported so far.

7. Conclusion

In this paper, we addressed the problem of generating accurate exploit-based signature for a single polymorphic worm and proposed a novel signature generation method based on multiple sequence alignment – a bioinformatics approach. This approach provides a more powerful method to accurately analyze the intrinsic similarities of worm samples. An IDS can employ such approach to locally generate accurate exploit-based signatures for polymorphic worms and such signatures can be distributed to other IDSs to circumvent further worm damage. The experiments on a range of polymorphic worms and real-world data traffic show that this approach is noise-tolerant and the signatures are more accurate and precise than other exploit-based signature generation methods. In future work we will exploit fast multiple sequence alignment algorithms (such as MAFFT) for use in signature generation.

Appendix. CSR Algorithm

The proposed CSR algorithm in Algorithm 2 is extended from the Needleman–Wunsch algorithm. It uses the divide-and-conquer strategy and dynamic programming technique to construct the optimal alignment that maximizes the similarity score in Formula (2). The maximum similarity score is achieved by iteratively calculating two matrices: the score matrix F and the traceback matrix PTR. Given two sequences X and Y, suppose $X[1..i]$ and $Y[1..j]$ are prefix subsequences of X and Y, $F_{i,\ j}$ stores the possible maximum similarity score for the two prefix subsequences, and $PTR_{i,\ j}$ records how to traceback to get the corresponding optimal alignment with the highest score $F_{i,\ j}$. The optimal alignment of X and Y will be stored in $F_{N,\ M}$ and $PTR_{N,\ M}$ after F and PTR have been achieved. The CSR algorithm consists of three main steps:

1. Initialize the iterative matrix F and PTR.
2. Recursively calculate the score matrix F and the traceback matrix PTR. As shown in Algorithm 2, $F_{i,\ j}$ is computed from $F_{i-1,\ j-1}$, $F_{i-1,\ j}$, and $F_{i,\ j-1}$ in three cases. Correspondingly, it means that the optimal alignment of subsequence $X[1..i]$ and $Y[1..j]$ is constructed in the following three cases: Case 1, optimal alignment of subsequence $X[1..i-1]$ and $Y[1..j-1]$, $X[i]$ to $Y[j]$; Case 2, optimal alignment of subsequence $X[1..i-1]$ and $Y[1..j]$, $X[i]$ to a gap; Case 3, optimal alignment of subsequence $X[1..i]$ and $Y[1..j-1]$, $Y[j]$ to a gap. Note that in case 2, we use the *enc*() function to reward consecutive matches, which is the core difference of our CSR algorithm from the Needleman–Wunsch algorithm.
3. Reduce the alignment from the traceback matrix PTR.

Algorithm 2. (*CSR algorithm.*)

```
input: sequence $X$, $Y$,
output: alignment result sequence $R$, similarity score $SC$
parameters: $W_m$: match score; $W_d$: mismatch score; $\delta$: gap penalty; enc(): contiguous substring rewarding function ;
```

```
Initialization
$N \leftarrow$ the length of $X$; $M \leftarrow$ the length of $Y$;
foreach $i$ such that $1 \le i \le N$ do
    $F_{i,0} \leftarrow i\delta; T_{i,0} \leftarrow 0; PTR_{i,0} \leftarrow$ Up;
end
foreach $j$ such that $1 \le j \le M$ do
    $F_{0,0} \leftarrow 0; T_{0,0} \leftarrow 0; PTR_{0,0} \leftarrow$ Traceend;
Iteration
    foreach $i$ such that $0 \le i \le N$ do
      foreach $j$ such that $0 \le j \le M$ do
        if $X_i = Y_j$ then
        $S_{xi,Yj} = W_m$ ; $T_{i,j} = T_{i-1,j-1} + 1$;
        else $S_{xi,Yj} = W_d$ ; $T_{i,j} = 0$
                       $F_{i-1,j-1} + S_{X_i,Y_j} + Enc(T_{i,j})$   [case 1]
        $F_{i,j} \leftarrow \max\{$ $F_{i-1,j} + \delta$                  [case 2] ;
                       $F_{i,j-1} + \delta$                  [case 3]
        $PTR_{i,j} \leftarrow \{$ Dial   if [case 1]Leftif [case 2]Upif [case 3]

    end
    end
Traceback
    $t \leftarrow PTR_{N,M}$ ; $i \leftarrow N$; $j \leftarrow M$;
    Allocate a empty sequence $R$;
    Whilet $\ne$ TraceEnd do
      Allocate a new character $r$ and add it to the head of
      sequence $R$;
      Switch $t$ do
        Case Dial $r \leftarrow X_i$; $i \leftarrow i-1$; $j \leftarrow j-1$
        Case Up $r \leftarrow$ 'space'; $i \leftarrow i-1$
        Case Left $r =$ 'space'; $j \leftarrow j-1$;

    end
    $t \leftarrow PTR_{i,j}$;
end
Return
$SC \leftarrow F_{N,M}$; return $SC$, $R$
```

REFERENCES

Brumley D, Newsome J, Song D, Wang H, Jha S. Towards automatic generation of vulnerability-based signatures. In: the 2006 IEEE symposium on security and privacy; 2006. pp. 2–16.

Costa M, Crowcroft J, Castro M, Rowstron A, Zhou L, Zhang L, Barham P. Vigilante: end-to-end containment of internet worms. In: ACM symposium on operating systems principles; 2005. pp. 133–47.

Crandall JR, Su Z, Wu SF, Chong FT. On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In: the 12th ACM conference on computer and communications security; 2005. pp. 235–48.

DeTristan T, Ulenspiegel T, Malcom Y, von Underduk MS. Polymorphic shellcode engine using spectrum analysis. Phrack 2003;11:61–9.

Kim HA, Karp B. Autograph: toward automated, distributed worm signature detection. In: USENIX security symposium; 2004. pp. 271–86.

Kreibich C, Crowcroft J. Honeycomb – creating intrusion detection signatures using honeypots. In: The second workshop on hot topics in networks (Hotnets II), Boston; 2003.

Ktwo, Admmutate: Shellcode mutation engine; 2001.

Levenshtein VI. Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 1966;10(8): 707–10.

Li Z, Sanghi M, Chen Y, Kao MY, Chavez B. Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. In: The 2006 IEEE symposium on security and privacy; 2006.

Li Z, Wang L, Chen Y, Fu Z. Network-based and attack-resilient length signature generation for zero-day polymorphic worms. In: The 15th IEEE International Conference on Network Protocols (ICNP 2007); 2007.

Liang Z, Sekar R. Fast and automated generation of attack signatures: a basis for building self-protecting servers. In: The 12th ACM conference on Computer and communications security; 2005a. pp. 213–22.

Liang Z, Sekar R. Automatic generation of buffer overflow attack signatures: an approach based on program behavior models. In: The 21st annual computer security applications conference; 2005b. pp. 215–24.

Lippmann R, Haines JW, Fried DJ, Korba J, Das K. The 1999 darpa off-line intrusion detection evaluation. Comput Networks 2000;34(4):579–95.

Needleman SB, Wunsch CD. A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology 1970;48: 443–53.

Newsome J, Song D. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In: The 12th annual network and distributed system security symposium; 2005.

Newsome J, Karp B, Song D. Polygraph: automatically generating signatures for polymorphic worms. In: the 2005 IEEE symposium on security and privacy; 2005. pp. 226–41.

Newsome J, Karp B, Song D. Paragraph: thwarting signature learning by training maliciously. RAID; 2006. pp. 81–105.

Notredame C. Recent progress in multiple sequence alignment: a survey. Pharmacogenomics 2002;3:131–44.

Notredame C, Higgins DG, Heringa J. T-coffee: a novel method for fast and accurate multiple sequence alignment. Journal of Molecular Biology 2000;302(1):205–17.

Perdisci R, Dagon D, Lee W, Fogla P, Sharif M. Misleading worm signature generators using deliberate noise injection. In: IEEE symposium on security and privacy; 2006.

Saitou N, Nei M. The neighbor-joining method: a new method for reconstructing phylogenetic trees. Molecular Biology and Evolution 1987;4(4):406–25.

Singh S, Estan C, Varghese G. Savage, Stefan, Automated worm fingerprinting. In: 6th USENIX OSDI; 2004.

Smirnov A, Chiueh, Tzi-cker. Dira: automatic detection, identification and repair of control-hijacking attacks. In: Proceedings of NDSS; 2005.

Song Y, Locasto ME, Stavrou A, Keromytis AD, Stolfo SJ. On the infeasibility of modeling polymorphic shellcode. In: ACM conference on computer and communications security (CCS); 2007.

Tang Y, Lu X, Xiao B. Generating simplified regular expression signatures for polymorphic worms. In: The 4th international conference on autonomic and trusted computing (ATC-07); 2007. pp. 478–88.

Tang Y, Luo J, Xiao B, Wei G. Concept, characteristics and defending mechanism of worms. IEICE Transactions on Information and Systems 2009;E92-D(5):799–809.

Team MD. Metasploit project; 2007.

Thompson JD, Higgins DG, Gibson TJ. Clustal w: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties

and weight matrix choice. Nucleic Acids Research 1994;22(22): 4673–80.
Venkataraman S, Blum A, Song D. Limits of learning-based signature generation with adversaries. NDSS; 2008.
Vulnerability vs. exploit signatures and ips. URL, http://archives.neohapsis.com/archives/sf/ids/2005-q2/0130.html; 2005.
Wang K, Cretu G, Stolfo SJ. Anomalous payload-based worm detection and signature generation. In: Recent Advances in Intrusion Detection (RAID); 2003.
Wang XF, Li Z, Xu J, Reiter MK, Kil C, Choi JY. Packet vaccine: black-box exploit detection and signature generation. In: The 13th ACM conference on computer and communications security; 2006. pp. 37–46.
Xu J, Ning P, Kil C, Zhai Y, Bookholt C. Automatic diagnosis and response to memory corruption vulnerabilities. In: The 12th ACM conference on computer and communications security; 2005. pp. 223–34.
Yegneswaran V, Giffin JT, Barford P, Jha S. An architecture for generating semantics-aware signatures. In: The 14th USENIX security symposium; 2005. pp. 97–112.

Yong Tang received the B.Sc, M.Sc and Ph.D. degrees in computer science from College of Computer, National University of Defense Technology, China, in 1998, 2002 and 2008 respectively. Now he is a lecturer in the College of Computer, National University of Defense Technology, China. His primary research field is network security, especially on Internet worm attacks, intrusion detection and Honeypot.

Bin Xiao received the B.Sc and M.Sc degrees in Electronics Engineering from Fudan University, China. in 1997 and 2000 respectively, and Ph.D. degree from University of Texas at Dallas, USA, in 2003 from Computer Science. Now he is an Assistant Professor in the Department of Computing of Hong Kong Polytechnic University, Hong Kong. His research interests include communication and security in computer networks, peer-topeer networks, wireless mobile ad hoc and sensor networks.

Xicheng Lu received the B.Sc. degree in computer science from Harbin Military Engineering Institute, China, in 1970. He was a visiting scholar at the University of Massachusetts between 1982 and 1984. He is now a professor in College of Computer, National University of Defense Technology, Changsha, China. His research interests include distributed computing, computer networks, parallel computing, network security, etc. He has served as member of editorial boards of several journals and cochaired many professional conferences. He is the joint recipient of more than a dozen academic awards, including four First Class National Scientific and Technological Progress Prize of China. He is an academician of the Chinese Academy of Engineering.