Generating Simplified Regular Expression Signatures for Polymorphic Worms*

Yong Tang[1], Xicheng Lu[1], and Bin Xiao[2]

[1] College of Computer, National University of Defense Technology, Changsha Hunan, 410073, P.R. China
{Ytang, Xclu}@nudt.edu.cn
[2] Department of Computing, Hong Kong Polytechnic University, Hong Kong
csbxiao@comp.polyu.edu.hk

Abstract. It is crucial to automatically generate accurate and effective signatures to defense against polymorphic worms. Previous work using conjunctions of tokens or token subsequence could lose some important information, like ignoring 1 byte token and neglecting the distances in the sequential tokens. In this paper we propose the *Simplified Regular Expression (SRE)* signature, and present its signature generation method based on the multiple sequence alignment algorithm. The multiple sequence alignment algorithm is extended from the pairwise sequence alignment algorithm, which encourages the contiguous substring extraction and is able to support wildcard string alignment and to preserve the distance of invariant content segment in generated SRE signatures. Thus, the generated SRE signature can express distance information for invariant content in polymorphic worms, which in turn makes even 1 byte invariant content extracted from polymorphic worms become valuable. Experiments on several types of polymorphic worms show that, compared with signatures generated by current network-based signature generation systems (NSGs), the generated SRE signatures are more accurate and precise to match polymorphic worms.

1 Introduction

Signature-based intrusion detection system(IDS) is one of the most deployed and effective way for worm defense. Todate, the signatures used by these IDSs for detecting worms are manually generated by security experts, which is too slow (typically days after worm released) in contrast with the speed of worm propagation (usually outbreak in zero day). Motivated by increasing the rate of signature generation, a number of automatic signature generation systems or methods have been proposed in recent years, which could be classified into two

* The work was partially supported by the National Basic Research Program of China (973) under Grant No. 2005CB321801, and the National Natural Science Foundation of China under Grant No. 90412011.

B. Xiao et al. (Eds.): ATC 2007, LNCS 4610, pp. 478–488, 2007.

categories - network-based signature generation (NSG) [1,2,3,4,5,6] and host-based signature generation (HSG) [7,8,9,10]. NSG systems have the advantage that they have no influence on the protecting hosts or networks. They usually firstly collect suspicious flows that contain the samples of worms through flow classifier or honeypot, then output content-based signatures for the worms by analyzing these suspicious flows.

Accuracy of outputted signatures is the most important criteria to evaluate NSG systems. The earlier NSG systems [1,2,3] generate single contiguous byte string signatures, which have been proven to be not effective [4,5] for matching worms. Some up-to-date NSG systems [4,6] generate token-based signatures, where a token is a byte sequence that occurs in a significant number of suspicious flows. But as we shall show in later in the paper, these token-based signature could lose some important information, like ignoring 1 byte token and neglecting the distances in the sequential tokens.

Regular expression have significant advantages for intrusion detection, in terms of flexibility, accuracy, and efficiency [11,12]. In this paper we propose *Simplified Regular Expression (SRE)* signature, a more expressive and accurate signature type. Given the samples of a polymorphic worm, we propose to generate its SRE signature using the multiple sequence alignment, which encourages the contiguous substring extraction and is able to support wildcard string alignment and to preserve the distance of invariant content segment in generated SRE signatures. Thus, the generated SRE signature can express distance information for invariant content in polymorphic worms, which in turn makes even 1 byte invariant content extracted from polymorphic worms become valuable. Experiments on several types of polymorphic worms show that our approach outperforms previous works in terms of signature accuracy.

The rest of paper is organized as follows. We first introduce the anatomy of polymorphic worms and summarize the limitation of the signature types outputted by current NSG systems. In Section 3 we provide the formal definition of SRE signature. Next in Section 4 we describe how to generate SRE signature for single polymorphic worm using sequence alignment and provide the corresponding algorithms. We present the evaluation and limitation of our approach in Section 5 and Section 6.

2 Background: Polymorphic Worms and Limitation of Current Signature Types

Polymorphic worms employ polymorphism technique to change their byte sequence at every instance for evading detection. Within a polymorphic worm body, there are two classes of bytes. *Invariant content* are those bytes fixed in value and *must* be present in every worm sample to ensure infection successful. *Wildcard bytes* are those which will change value for each different worm sample. For example, a polymorphic Code Red II worm is presented in Fig. 1, which in turn contains seven invariant content: “GET”, “.ida?”, “XX”, “%u”, “%u7801”, “=”, and “HTTP /1.0\r\n”. Within these invariant content, “%u7801” is 4 bytes

after "%u", "HTTP /1.0\r\n" is 7 bytes after "=". We call these relations, like "possible start position of a substring" or "how many characters between two substrings", *distance restrictions*. In practice, distance restrictions are critical for successful worm infections.

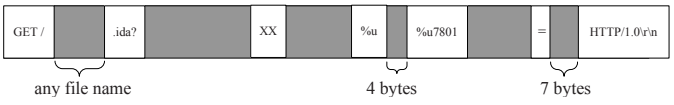

Fig. 1. Polymorphic Code Red II worm. Shaded content represents wildcard bytes, unshaded content represents invariant bytes.

Next we use this example to explain the limitations of the signature types outputted by current NSG systems. The earlier NSG systems [1,2,3] generate contiguous byte string signatures, like "HTTP /1.0\r\n" or "%u7801", which are not accuracy enough to characterize this worm. Polygraph[4] and Hamsa[6] generate token (substring with a minimum length and a minimum coverage in the suspicious flows) based signatures. But we found that token-based signatures are still not accurate enough. First, tokens can not have the length of 1, like "="; otherwise, every possible character (0-255 in value) will be extracted as a "token". Second, Polygraph and Hamsa can not express the distance restrictions of invariant content, like "'%u7801' is 4 bytes after '%u"'. Y. Tang et al. [13] propose PADS signature. A PADS is a position-aware frequency distribution of characters in a fixed length region. But there is not an fixed length region that contains all of invariant content in this example.

3 SRE Signature

A regular expression describes a set of strings without enumerating them explicitly. It is widely believed that regular expression have significant advantages for intrusion detection, in terms of flexibility, accuracy, and efficiency [11,12]. Motivated by the insufficiency of the signature types of previous NSG systems, we propose a novel signature type–*Simplified Regular Expression (SRE) signature*.

A SRE signature is a simplified form of regular expression, in which there are only three repeating qualifiers: "$*$", "$/k_1, k_2/$" and "$/k/$". We replace the ".*" in regular expression by "$*$" to represent an arbitrary string (including zero-length string), replace ".$\{k_1, k_2\}$" by "$/k_1, k_2/$" to represent any string with a length from k_1 to k_2, and replace ".$\{k\}$" by "$/k/$" to represent a string consisting of k number of arbitrary character. For example, *"'one'*'two'[2]'three'[3,5]"* is a SRE signatures that is equal to the regular expression *"one.*two.{2}three.{3,5}"*.

Suppose $\Phi = \{*, [k], [k_1,k_2]\}$ is the set of the three repeating qualifiers we just introduced. Σ^+ is the set of not empty strings over a finite alphabet Σ. The formal definition of SRE signature is provided by Definition 1.

Definition 1 SRE Signature. A SRE signature $=(p_0)s_1p_1s_2p_2s_3\ldots p_{k-1}s_k(p_k)$, where $p_i \in \Phi$ is a *repeating qualifier*, $s_i \in \Sigma^+$ is a *substring* ($i \in [0,k]$), (p_0) and (p_k) means p_0 and p_k are optional.

Within a SRE signature, the *substrings* are used to express the invariant content in polymorphic worms, the *repeating qualifiers* are used to express the distance restrictions between invariant content. For the previous example, we can use the SRE signature *"'GET /'*'.ida?'*'XX'*'%u'[4]'%u7801'*'='[7]'HTTP / 1.0\r\n"'* to precisely express the characteristic of Code Red II worm.

4 Generating SRE Signature for Polymorphic Worm Using Sequence Alignment

4.1 Overview

Sequence alignment is the procedure of comparing two (pairwise) or more (multiple) sequences by searching for a series of individual characters or character patterns that are in the same order in the sequences. Sequence alignment is widely used to quantify and visualize similarity between sequences, and it has been most prominently applied in bioinformatics [14,15]. A pairwise sequence alignment is a scheme of writing one sequence on top of another where the residues in one position are deemed to have a common character. Fig. 2 illustrates the pair-wise sequence alignment between "ONExxxTWOxxxxTHREExx xx" and "dsfONEdsdTWOvvvTHREEb". By inserting some gap('-'), the common characters are deemed to the same columns.

-	-	-	O	N	E	x	x	x	T	W	O	x	x	x	x	T	H	R	E	E	x	x	x	x	
d	s	f	O	N	E	d	s	d	T	W	O	-	v	v	v	T	H	R	E	E	-	-	-	b	
*	*	*	O	N	E	?	?	?	T	W	O	*	?	?	?	T	H	R	E	E	*	*	*	?	

Fig. 2. Example of pair-wise sequence alignment

The alignment result can be described by a sequence with wildcards, in which the question wildcard '?' presents one character, the asterisk wildcard '*' presents one or zero character. The alignment result in Fig. 2 is expressed by the sequence "***'ONE'???'TWO'???*'THREE'***?". As we shall introduce later, such alignment result can be converted to a SRE signature according to its semantic. For instance, "***'ONE'???'TWO'???*'THREE'***?" can be converted to SRE signature "[0,3]'ONE'[3]'TWO'[3,4]'THREE'[1,3]".

If we have captured a number of network flows that are the instances of a polymorphic worm, we generate the worm's signature by the following steps.

1) Transform the flows to character sequences. In the rest of the paper, these character sequences are referred to *samples* of a worm.

2) Analyze these samples by multiple sequence alignment that arranges these samples in a scheme where positions believed to be invariant bytes are written in a common column. For example, suppose A="oxnxexzxtwoxxw", B ="ytwoyown

yeyz", C ="cvcvcvtwovcwc" are three samples of a polymorphic worm, as Fig. 4 shows, aligning these three samples will get a result "*******?'two'??'w'*****".

3) Transfer the alignment result to a SRE signature as the final signature for this worm. For previous example, the alignment result "*******?'two'??'w'*****" can be converted to the SRE signature "[1,8]'two'[2]'w'[0,5]".

4.2 Problem of Current Sequence Alignment Algorithm

The Needleman-Wunsch algorithm [16] is a typical global alignment algorithm that computes the optimal alignment between two sequences by maximizing a similarity score function as Formula (1) gives, where k_m denotes the number of matches, W_m denote the score for character match, k_d denotes the number of mismatches, W_d denote the score for character mismatch, k_{gaps} denotes the number of gaps, δ denote the penalty score for a gap. If we set $W_m = 1$, $W_d = 0$, $\delta = -1$, for the example in Fig. 2, this algorithm outputs an alignment with similarity score 4 $(11 \times 1 + 7 \times 0 + 7 \times -1)$.

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

If a *piece* is a substring in alignment result with a length of only 1, we found that the Needleman-Wunsch algorithm is likely to output a large number of pieces in resulting alignment, instead of outputting the invariant content of polymorphic worms. Consider the simple example provided by Polygraph [4] that align two strings "oxnxexzxtwox" and "ytwoyoynyeyz". Fig. 3(a) shows the alignment result of the Needleman-Wunsch algorithm, which contains four pieces('o','n','e','z'). Creating too many trivial and useless pieces will prevent finding contiguous invariant content we are concerning about. Obviously Fig. 3(b) is a better alignment since the substring 'two' is semantical.

(a) (b)

Fig. 3. Two Alignment results for different algorithm

Through a plenty of experiments, we found that no matter the parameters are adjusted, the Needleman-Wunsch algorithm always tend to product a large number of pieces and hence lost some invariant content in polymorphic worms.

4.3 Pairwise Sequence Alignment Algorithm

As Algorithm 1 shows, we propose a new pairwise sequence alignment algorithm for our approach–*CSR (contiguous substrings rewarded)* algorithm, which is extended from the Needleman-Wunsch algorithm by the following three means:

1.Rewarding contiguous substrings: Motivating by reducing pieces, we modify the similarity score function of the Needleman-Wunsch algorithm from Formula (1) to Formula (2) by introducing a score function $enc()$ to reward contiguous substrings. For the example given in Fig. 3, if we define $enc(x) = 3(x-1)$ and set $W_m = 0.5$, $W_d = 0$, $\delta = -1$, the similarity score of Fig. 3(a) is -8 $(0.5 \times 4 + 0 \times 3 + -1 \times 10)$, the similarity score of Fig. 3(b) is -6.5 $(0.5 \times 3 + 0 \times 2 + -1 \times 14 + 3 \times (3-1))$. Hence our CSR algorithm will output the better alignment Fig. 3(b).

$$SC(x,y) = k_m \times W_m + k_d \times W_d + k_g \times \delta + \sum_{\substack{s \text{ is substring} \\ \text{in alignment result}}} Enc(|s|) \tag{2}$$

2.Supporting wildcards: The CSR algorithm allows the input sequences contain two previously introduced wildcard–'?' and '*'. We provide a set of character comparison rules, as Table 1 depicts, where 'α' and 'β' denote two different characters, '$-$' is a gap in alignment. By calling the function LookupCharCompTab(.) (line 27), the CSR algorithm lookup this table to determine the value of a position in result sequence.

Table 1. Character comparison rules for CSR algorithm. x and y are the characters in input sequences in the same column, r is the character will appear in the same column of alignment result sequence.

x	α	α	α	?	α	?	α	$-$	$*$	α	?
y	α	β	?	?	?	$*$	$*$	$*$	$*$	$-$	$-$
r	α	?	?	?	?	$*$	$*$	$*$	$*$	$*$	$*$

3.Preserving distance restriction: In order to preserve the distance restrictions during the alignments of polymorphic worm samples, we assign every character in sequences a length area $[min, max]$ during the alignment process, where min is the low-bound, max is the up-bound. As the line 4, 7 shows, we first initialize the length range of every character in input sequences to $[1, 1]$. During the alignment, we set the length range of a inserted gap to $[0, 1]$. As the algorithm line 28, 29 shows, finally the length range of a character in alignment result is calculated by minimizing the low-bound and maximizing the up-bound of the characters with the same column.

Convert alignment result to SRE signature. The alignment result of the CSR algorithm is a sequence with wildcards. Notice that each wildcard carries a length range, thus we can easily convert alignment result to a SRE signature by merging the wildcards between two substrings and accumulating their length range. For example, consider the alignment result in Fig. 3(b), the length range of every '*' is [0,1], the length range of every '?' is [1,1], we merge the eight wildcards before substring "two" to one repeating qualifier, and calculate its low-bound of length range to $0 \times 7 + 1 = 1$, up-bound to $1 \times 7 + 1 = 8$. Hence, the repeating qualifier before the substring 'two' is '[1,8]'. Repeat this process, we can get the converted SRE signature "[1,8]'two'[1,8]" finally.

input : sequence X, Y,
output : alignment result sequence R, similarity score SC
Parameters: W_m: match score; W_d: mismatch score; δ: gap penalty ; `enc` (): contiguous substring rewarding function ; `LookupCharCompTab` (.): lookup the character comparison table to determine the character placed in alignment result.

1 Initialization
2 $N \leftarrow$ the length of X; $M \leftarrow$ the length of Y;
3 **foreach** i *such that* $1 \leq i \leq N$ **do**
4 $F_{i,0} \leftarrow i\delta$;$T_{i,0} \leftarrow 0$; $PTR_{i,0} \leftarrow$ Up; $X_i.min \leftarrow 1$; $X_i.max \leftarrow 1$;
5 **end**
6 **foreach** j *such that* $1 \leq j \leq M$ **do**
7 $F_{0,j} \leftarrow j\delta$; $T_{0,j} \leftarrow 0$; $PTR_{0,j} \leftarrow$ Left; $Y_j.min \leftarrow 1$; $Y_j.max \leftarrow 1$;
8 **end**
9 $F_{0,0} \leftarrow 0$; $T_{0,0} \leftarrow 0$; $PTR_{0,0} \leftarrow$ TraceEnd;
10 Iteration
11 **foreach** i *such that* $0 \leq i \leq N$ **do**
12 **foreach** j *such that* $0 \leq j \leq M$ **do**
13 **if** X_i, Y_j *are not wildcard* **and** $X_i = Y_j$ **then**
14 $S_{X_i,Y_j} = W_m$; $T_{i,j} = T_{i-1,j-1} + 1$;
15 **else** $S_{X_i,Y_j} = W_d$; $T_{i,j} = 0$;
16 $$F_{i,j} \leftarrow \max \begin{cases} F_{i-1,j-1} + S_{X_i,Y_j} + \texttt{enc}(T_{i,j}) & [case1] \\ F_{i-1,j} + \delta & [case2] \\ F_{i,j-1} + \delta & [case3] \end{cases}$$
17 $$PTR_{i,j} \leftarrow \begin{cases} \text{Dial} & \textbf{if } [case1] \\ \text{Left} & \textbf{if } [case2] \\ \text{Up} & \textbf{if } [case3] \end{cases}$$
18 **end**
19 **end**
20 Traceback
21 $t = PTR_{N,M}$; $i \leftarrow N$; $j \leftarrow M$;
22 Allocate an empty sequence R;
23 **while** $t \neq$ TraceEnd **do**
24 Allocate a new character r and add it to the head of sequence R;
25 **switch** t **do**
26 **case** Dial
27 $r \leftarrow$ `LookupCharCompTab`(X_i, Y_i);
28 $r.min \leftarrow \min(X_i.min, Y_j.min)$; $r.max \leftarrow \max(X_i.max, Y_j.max)$;
29 $i \leftarrow i - 1$; $j \leftarrow j - 1$;
30 **case** Up
31 $r \leftarrow$ '*'; $r.min \leftarrow 0$; $r.max \leftarrow \max(X_i.max, 1)$; $i \leftarrow i - 1$;
32 **case** Left
33 $r =$ '*'; $r.min \leftarrow 0$; $r.max \leftarrow \max(Y_j.max, 1)$; $j \leftarrow j - 1$;
34 **end**
35 $t \leftarrow PTR_{i,j}$;
36 **end**
37 Return
38 $SC \leftarrow F_{N,M}$; **return** SC, R

Algorithm 1. CSR algorithm

4.4 Multiple Sequence Alignment Algorithm

Given the samples of a polymorphic worm, we aim to generate its SRE signature using multiple sequence alignment. Because the CSR algorithm supports wildcard characters and can preserve distance restrictions, we simply design our multiple sequence alignment algorithm by progressively employing CSR algorithm, as Algorithm 2 gives.

```
   input  : sequence set S,
   output: alignment result sequence R
1  repeat
2      randomly select two sequences X and Y from S; S ← S \ {X, Y}; employ the CSR
       algorithm to align X and Y, the result sequence is A_{X,Y}; S ← S ∪ {A_{X,Y}}
3  until |S| = 1 ;
4  return A_{X,Y}
```

Algorithm 2. Multiple sequence alignment algorithm

For example, if the sequences A ="oxnxexzxtwoxxw", B ="ytwoyownyeyz", and C ="cvcvcvtwovcwc" are three samples of a polymorphic worm, as Fig.4 shows, we first align A and B, and then use the result (denoted as MALIGN(A, B)) to align with C, finally get the alignment result "*******?'two'??'w'*****", which can be converted to the SRE signature "[1,8]'two'[2]'w'[0,5]" at last. We can see that the distance restrictions, such as "'w' is 2 bytes after 'two'", are correctly preserved in this SRE signature.

Fig. 4. Example of multiple sequence alignment algorithm

5 Evaluation

Experiment Settings. Similar to Polygraph [4] and Hamsa [6], we use three synthetically generated polymorphic worms (ATPhttpd exploit, Code Red II exploit, and BIND-TSIG exploit) to evaluate our approach. In order to test the false positive rate of generated signatures, we use the first week's network TCPdump data of the 1999 DARPA Intrusion Detection Evaluation Data Sets [17] (1.8GB) as the normal traffic data, and set $enc(x) = 3(x - 1)$, $W_m = 0.5$, $W_d = 0$, $\delta = -1$ as the parameters of CSR algorithm.

Signature Quality. For each worm, we generate it signature by analyzing its eight samples using our multiple sequence alignment algorithm. Table. 2 gives the generated signatures of the three worms. Table 5 gives the signature comparison for Code Red II worm between our approach and Polygraph. we can see that our SRE signature is more precise than Polygraph's *conjunction signature*, because our signature preserves the distance restrictions for invariant content through the repeating qualifiers '[4]' and '[7]'; in addition, our signature contains '=', which is an invariant content with a length of 1, however Polygraph does not.

Performance Study. Suppose all flows are l bytes long, although the CSR algorithm added some sentences (line 13-15) each with a runtime of $O(1)$ in the main iteration of Needleman-Wunsch algorithm, the time and apace overhead

Table 2. Generating signatures for three worms

Worm	Signature	False positive	False negative	Speed (secs)	Memery usage (MB)
BIND-TSIG	*‘\xFF\xBF’[200]‘\x00\x00\xFA’[2]	0	0	2.1	4.0
ATPhttpd	‘GET /’* ‘\xFF\xBF’*‘HTTP/1.1\r\n’	0	0	9.2	4.3
Code Red II	‘GET /’*‘.ida?’*‘XX’*‘%u’[4] ‘%u7801’*‘=’[7]‘HTTP /1.0\r\n’	0	0	1.1	3.9

Table 3. Signature type comparison for Polymorphic Code Red II worm

Signature	Generated signature
Conjunction signature (Polygraph)	GET /.*.ida?.*XX.*%u.*%u7801.*HTTP /1.0\r\n
SRE signature	‘GET /’*‘.ida?’*‘XX’*‘%u’[4]‘%u7801’*‘=’[7]‘HTTP /1.0\r\n’

of CSR algorithm is still $O(l^2)$. Aligning θ flows using our multiple sequence algorithm takes $O(\theta l^2)$ time and $O(l^2)$space. Table 2 shows the time and memory consumption of generating signatures for the three worms. All experiments were executed on a PC with single 3.0GHz Intel Pentium IV processor.

6 Limitation and Future Work

In this work we focus on signature generation, how to capture the samples of polymorphic worms is beyond the scope of this paper. And we only consider generating single signature given the samples of a polymorphic worm, which is the base of the fully general case that generating signatures for mixed several different worms.

In our future work, we plan to design a system that can automatic generate signatures for multiple polymorphic worms in a live environment. Given the suspicious flows that contain the samples of several polymorphic worms, we first need to cluster them into some clusters. Fortunately there are already some researches on worm samples clustering [4,18]. After clustering, we can generate a signature for each cluster using the method presented in this paper.

7 Conclusion

In this paper, we propose a new signature type–SRE signature. A SRE signature is a simplified form of regular expression, which is more effective for characterizing polymorphic worms. We present a multiple sequence alignment based method to generate SRE signature for single polymorphic worm. In order to overcome the problem that the typical Needleman-Wunsch algorithm will product a large number of useless pieces, we propose a novel pairwise sequence alignment algorithm–CSR algorithm. Experiment results indicate that our approach is effective for automatic signature generation of polymorphic worms.

References

1. Kreibich, C., Crowcroft, J.: Honeycomb - creating intrusion detection signatures using honeypots. In: Proceedings of the Second Workshop on Hot Topics in Networks (Hotnets II), Boston (November 2003)
2. Kim, H.A., Karp, B.: Autograph: Toward automated, distributed worm signature detection. In: USENIX Security Symposium, pp. 271–286 (2004)
3. Singh, S., Estan, C., Varghese, G., Savage, S.: Automated worm fingerprinting. In: Proc. 6th USENIX OSDI, San Francisco, CA (December 2004)
4. Newsome, J., Karp, B., Song, D.: Polygraph: Automatically generating signatures for polymorphic worms. In: Proceedings of the 2005 IEEE Symposium on Security and Privacy, pp. 226–241. IEEE Computer Society Press, Washington (2005)
5. Crandall, J.R., Su, Z., Wu, S.F., Chong, F.T.: On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits. In: Proceedings of the 12th ACM conference on Computer and communications security, pp. 235–248. ACM Press, New York (2005)
6. Li, Z., Sanghi, M., Chen, Y., Kao, M.-Y., Chavez, B.: Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience. In: Proceedings of the 2006 IEEE Symposium on Security and Privacy (S&P'06), IEEE Computer Society Press, Washington (2006)
7. Newsome, J., Song, D.: Dynamic taint analysis for automatic detection, analysis, and signaturegeneration of exploits on commodity software. In: NDSS (2005)
8. Liang, Z., Sekar, R.: Fast and automated generation of attack signatures: a basis for building self-protecting servers. In: CCS '05: Proceedings of the 12th ACM conference on Computer and communications security, pp. 213–222. ACM Press, New York (2005)
9. Xu, J., Ning, P., Kil, C., Zhai, Y., Bookholt, C.: Automatic diagnosis and response to memory corruption vulnerabilities. In: CCS '05: Proceedings of the 12th ACM conference on Computer and communications security, pp. 223–234. ACM Press, New York (2005)
10. Wang, X., Li, Z., Xu, J., Reiter, M.K., Kil, C., Choi, J.Y.: Packet vaccine: black-box exploit detection and signature generation. In: CCS '06: Proceedings of the 13th ACM conference on Computer and communications security, pp. 37–46. ACM Press, New York (2006)
11. Sommer, R., Paxson, V.: Enhancing byte-level network intrusion detection signatures with context. In: CCS '03: Proceedings of the 10th ACM conference on Computer and communications security, pp. 262–271. ACM Press, New York (2003)
12. Kumar, S., Dharmapurikar, S., Yu, F., Crowley, P., Turner, J.: Algorithms to accelerate multiple regular expressions matching for deep packet inspection. In: Proceedings of ACM SIGCOMM'06, vol. 36, pp. 339–350. ACM Press, New York (2006)
13. Tang, Y., Chen, S.: Defending against internet worms: A signature-based approach. In: Proceedings of the 24th Annual Conference IEEE INFOCOM 2005 (March 2005)
14. Gelfand, M.S., Mironov, A., Pevzner, P.: Gene recognition via splices sequence alignment. In: Proc. Natl.Acad. Sci. USA, pp. 9061–9066 (1996)
15. Goad, W.B., Kanehisa, M.I.: Pattern recognition in nucleic acid sequences: a general method for finding local homologies and symmetries. Nucleic Acids Research 10, 247–263 (1982)

16. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48, 443–453 (1970)
17. Lippmann, R., Haines, J.W., Fried, D.J., Korba, J., Das, K.: The 1999 darpa off-line intrusion detection evaluation. Comput. Networks 34(4), 579–595 (2000)
18. Yegneswaran, V., Giffin, J.T., Barford, P., Jha, S.: An Architecture for Generating Semantics-Aware Signatures. In: Proceedings of the 14th USENIX Security Symposium, Baltimore, MD, USA, pp. 97–112 (August 2005)