A Novel Signature Generation Approach in Noisy Environments for Detecting Polymorphic Worm

Jie Wang[1(✉)] and Jie Wu[2]

[1] School of Information Science and Engineering, Central South University, Changsha 410083, China
jwang@csu.edu.cn

[2] College of Science and Technology, Temple Unversity, Philadelphia, PA 19122, USA
jiewu@temple.edu

Abstract. Polymorphic worms can change their patterns dynamically, that makes the generation of worm signatures a challenging task. In noisy environments the task is more difficult. In this paper, we propose a novel approach CGNRS to generate worm neighborhood-relation signatures (NRS) from suspicious flow pool with noisy sequences. CGNRS divides n sequences into m groups and each group contains 20 sequences. CGNRS identifies worm sequences for each group by adopting color coding and computing NRS. Then all identified worm sequences are used to generate NRS. We have carried out extensive experiments to evaluate the quality of signatures generated by CGNRS. In comparison with signatures generated by existing approaches, the experiment results show that NRS generated by our approaches can be used to detect effectively polymorphic worm when the suspicious flow pool contains noise sequences.

Keywords: Signature generation · Polymorphic worm detection · Neighborhood relation · Color coding · Intrusion detection

1 Introduction

Worms are self-replicating malicious programs and represent a major security threat for the Internet. They can infect and damage a large number of vulnerable hosts at timescales where human responses are unlikely to be effective [1]. According to an empirical study, a typical zero-day attack may last for 312 days on average [2]. Polymorphic worms are characterized by their ability to change their byte sequence when they replicate and propagate, and they can change their appearances with every instance. They have caused great damage to Internet in recent years.

In order to evade detection system, polymorphic worms use some techniques to remove any static signature which may be obtained from the payload.

A polymorphic worm can be implemented by some methods such as instructions rearrangement, garbage-code insertion, register-reassignment, instruction-substitution and encryption [3–5]. No matter what method is used the worms produced can always be classified into two classes. The first class of worm signatures exhibits constant subsequences and we call these polymorphic worm PWCS (polymorphic worm with constant subsequences). The second class can not be identified based on constant subsequences but can be identified based on other kinds of regularities in the worm sequences, e.g. worm byte frequency distribution and the difference between adjacent bytes of the worm sequences. We call the second class of polymorphic worm PWVS (polymorphic worm with variant subsequences).

Polymorphic worm detection are mainly based on signature because of their simplicity and ability to operate online in real time [6]. However, most deployed worm signature based detection systems are ineffective to detect PWVS [7]. Generating high quality and accurate signatures based on the characteristic of polymorphic worm remains largely an open problem [8,9].

Additionally, suspicious flow pool, from which worm signatures are generated, often includes noise flows. Noise flows often are normal flow sequences or crafted worm-like flow sequences. These noise sequences can lead to existing signature generation methods to generate useless signature. For example, Fogla [10] introduced polymorphic blending attacks, which evade signature-based intrusion detection systems by blending lots of normal flow sequences with worm sequences. Perdisci [11] proposed an attack against worm signature generation systems. This attack uses deliberate noise injection and misleads these systems to generate useless signatures.

Because of the polymorphism of worm and noise problem in suspicious flow pool, existing work for defending against polymorphic worms and generating their signatures have inadequateness because they either can not handle noise well in the process of generating worm signature or can not generate worm signature to detect PWVS successfully. In this paper, for avoiding the situation that signature of polymorphic worms can not be generated when any static signature or invariant substring are removed from worms, we propose a neighborhood-relation signatures (NRS) to detect polymorphic worm. NRS is a collection of distance frequency distributions between neighbor byte. Moreover, for solving noise problem during generating worm signature, we propose CGNRS algorithm by combining color coding methods to generate NRS from suspicious flow pool with noise sequences. CGNRS divides n sequences into m groups and each group contains 20 sequences, and then identifies worm sequences for each group by adopting color coding. Finally, all identified worm sequences are clustered to generate NRS. The rest of the paper is organized as follows. Related works are introduced in Sect. 2. CGNRS algorithm is proposed in Sect. 3. Experimental results are illustrated in Sect. 4. Section 5 draws the conclusions.

2 Related Work

Anomaly-based detection is one of defending worm techniques. For example, a novel machine learning based framework is presented in [12] to detect known and newly emerging malware at a high precision using layer 3 and layer 4 network traffic features. This framework can detect Conficker worm successfully. Another technique for defending worm is signature-based detection. Signature-based detection techniques look for specific byte sequences (called attack signatures) that are known to appear in the attack traffic. Their efficiency of defending against worms depends on the quality of worm signatures that can be generated.

Recently, there have been many research efforts on generating signature for worms. Some of them only can detect single worms. For example, G. Portokalidis et al. [13] implemented SweetBait system, which automatically generates worm signatures. S. Ranjan et al. [14] presented DoWicher, which extracts the worm content signature via a LCS algorithm applied over the flow payload content of isolated flows. M. Cai et al. developed a collaborative worm signature generation system (WormShield) [15] that employs distributed fingerprint filtering. In these systems, a worm is assumed to have a long invariant substring used as a signature to detect the worm. However, many polymorphic worms do not contain a long enough common substring. Systems mentioned above are not applicable to detect polymorphic worms.

Some methods to generate worm signatures are more complicated than that based on LCS algorithm. For example, J. Newsome et al. [16] presented Polygraph, a signature generation system. Polygraph extracted multiple invariant substrings in all worm variants as worm signature. Z. Li et al. [17] developed the Hamsa, an improved system over Polygraph in terms of both speed and attack resilience. Hamsa takes the number of occurrences of a substring token into a part of signature. Lorenzo Cavallaro et al. [18] proposed LISABETH, an improved version of Hamsa, an automated content-based signature generation system for polymorphic worms that uses invariant bytes analysis of network traffic content. Burak Bayogle et al. [19] proposed Token-Pair Conjunction and Token-Pair Subsequence signature for detecting polymorphic worm threats. Y. Tang et al. [6] proposed Simplified Regular Expression (SRE) signature, and used multiple sequence alignment techniques to generate exploit-based signatures. A graph based classification framework of content based polymorphic worm signatures is presented in [20]. Based on the defined framework, a new polymorphic worm signature scheme, Conjunction of Combinational Motifs (CCM), is proposed. CCM utilizes common substrings of polymorphic worm copies and also the relation between those substrings through dependency analysis. Above these methods assumed that multiple invariant substrings must be present in all instances of polymorphic worm. They can not produce signature for PWVS because there is no same byte sequences exists in different copies of PWVS. In this paper, we propose the CGNRS algorithm to solve the noise problem by using color coding.

3 CGNRS Algorithm

3.1 Worm Signature

In this paper, we use NRS [21] as worm signature. $S = \{S_1, S_2, ..., S_n\}$ is a set of worm sequences, where $S_i = c_1c_2...c_m$. There is at least a significant region in worms, which infects victim. Suppose $a_1, a_2, ..., a_n$ are the starting positions of significant region in n sequences, and the width of significant region is w. The number of worm sequences, in which neighbor distance of position p in the significant regions is d, is denoted as $count(p, d)$. Neighbor distance distribution $f_p(d)$ is as follows.

$$f_p(d) = \frac{count1(p, d)}{n} \tag{1}$$

where $\sum_{d\in[0...255]} f_p(d) = 1$, and $p = 1, 2, ..., w-1$. NRS signature of n sequences is defined as $(f_1, f_2, ..., f_{w-1})$. The process of computing NRS (GNRS) was described in [21].

3.2 Process of Identifying Worm Sequences from 20 Sequences by Applying Color Coding

Consider an n-size suspicious flow pool with k worm sequences, CGNRS is designed to generate worm signature from the pool. The suspicious flow pool comprises of worm sequences and noise sequences. Worm sequences are PWCS, PWVS or both of them. Noise sequences contain normal flow sequences from real network and sequences generated by using special methods discussed in [11]. Since the position of worm sequence is unknown, we have to extract signature from each k-combinations of n sequences. In other words, procedure of extracting signature will be run C_n^k times. However, when the number of sequences in suspicious flow pool is very large, for example, when n=2000, C_n^k is too large to run for extracting signature. In this paper, we use divide-and-conquer method. n sequences are divided into m g-sequences, and $g = \lfloor \frac{n}{m} \rfloor$. Then we identify worm sequences by running identification procedure for each u-combinations sequences of g-sequences. Here the identification procedure is run mC_g^u times. If g is too large, so is C_g^u, and if g is too small, it is difficult to distinguish worm sequences. From above analysis we can see that the larger the value of g is, the easier distinguishing worm sequences will be, and the greater complexity of computation is. Assuming $g = 20$, $C_{20}^{15} = 15504$, $C_{20}^{14} = 38760$, $C_{20}^{13} = 77520$, $C_{20}^{12} = 15504$, and $C_{20}^{11} = 167960$. From these data, it can be seen that computation complexity is still large when $g = 20$. Therefore, we adopt color coding to reduce the number of times C_g^u of running identification procedure when $g = 20$. The detailed introduction about color coding was described in [8]. In this paper, $Coloring(20, u)(u = 11, 12, \cdots, 19)$ is used. Table 1 shows the comparison between the size of $Coloring(20, u)$ and the number of u-combinations of 20.

When the number of noise sequences in suspicious flow pool is larger than the number of worm sequences, it is hard to identify worm sequences. Therefore we

Table 1. Comparison between the size of $Coloring(20, u)$ and the number of u-combinations of 20

	Coloring(20,u)	u-combinations of 20
u=19	10	20
u=18	50	190
u=17	170	1140
u=16	403	4845
u=15	862	15504
u=14	1220	38760
u=13	2036	77520
u=12	2085	125970
u=11	3250	167960

only consider how to identify worm sequences when g-size suspicious flow pool includes $u(u > \frac{g}{2})$ worm sequences. The procedure of building $Coloring(g, u)$ is denoted as Build coloring(g,u). Assuming f is the number of (g, u)-colorings in the set $Coloring(g, u)$. For each (g, u)-coloring, sequences with the same color are merged. So g sequences are converted to u sequences. Then algorithm GNRS generates NRS_i signature for the u sequences. After NRS_i is generated, it is evaluated in one filter flow pool, which contains n normal sequences. We compute the matching score Θ_j of the jth sequence in the filter flow pool with NRS_i based on Eq. (1). If $\Theta_j > 0(1 \leq j \leq n)$, p_i adds 1. If $p_i/n < \varepsilon$, the u sequences used to generate NRS_i is considered as u worm sequences in the suspicious flow pool, where ε is a small predefined percentage. If all $NRS_i(1 \leq i \leq f)$ are not satisfied, u worm sequences are not identified. The process is described as GeWS algorithm, which is illustrated in Fig. 1.

In Fig. 1, $g = 20$ and $11 \leq u \leq 20$. Results returned by GeWS are u sequences. According to the method of color coding, the set $Coloring(g, u)$ can cover all u-combinations of g sequences. Therefore signatures generated from u sequences are the same as signatures extracted from each u-combinations of g sequences.

3.3 Description of Algorithm CGNRS

Given a suspicious flow pool with n sequences. Firstly, n sequences are divided into many groups, each of which includes 20 sequences. If the number of sequences in the last group is less than 20, copies of other sequences are put into the last group. Assuming that n sequences are divided into $m = \lceil \frac{n}{20} \rceil$ groups $G_1, G_2, ..., G_m$. Algorithm GeWS is applied to generate NRS signatures for each group, and returns u worm sequences or returns "can not identify worm sequences" for each group.

For group G_i, the parameter u of GeWS algorithm is 20. Assuming that NRS_i is signature generated by GeWS algorithm, and S_i' is u worm sequences

```
**Algorithm GeWS**($S_g$, $N$, $u$)
Input: $g$ sequences $S_g = \{y_1, \cdots, y_g\}$, $n$ normal sequences $N = N_1, N_2, \cdots, N_n$, $u$, Width of signature $w$;
Output: $u$ sequences $S_u$, NRS for $g$ sequences;
Build $Coloring(g, u)$;
For each $(g, u)$-coloring of $Coloring(g, u)$
    {$g$ sequences of $S_g$ are colored with the $(g, u)$-coloring;
    $S_u = \{x_1, \cdots, x_u\}$ is generated by merging sequences with the same color in $S_u$;
    $NRS_i \leftarrow GNRS(S_u, w)$;
    For $j = 1$ to $n$
        Computing the matching score $\Theta_j$ of $N_j$ with $NRS_i$;
        If $(\Theta_j > 0)$ then $p_i + +$;
    If $p_i/n < \varepsilon$
        $NRS = NRS_i$; Flag="True"; break;
    Flag="False";
    }
If(Flag=="True") then
    Return ($S_u$,$NRS$)
else return("can not identify worm sequences");
```

Fig. 1. Algorithm GeWS

returned by the algorithm. If NRS_i is not generated, CGNRS calls algorithm GeWS with $u = u - 1$ again until NRS_i is generated or $u < 11$. After dealing with all groups, CGNRS can obtain a set sequences S' by merging all not-null S'_i. Then CGNRS employs GNRS algorithm to generate NRS for S'. The CGNRS algorithm is illustrated in Fig. 2.

As shown in Fig. 2, the parameter a is set to be 20 and b is set to be 11 in CGNRS. GeWS algorithm aims to generate signatures when the number of worm sequences is larger than that of noise sequences in each group. If there is at least one group which generates signature and obtains the set of worm sequences, CGNRS can generate worm signature with probability 1.

4 Experiments and Results

Four kinds of worm are used in the experiments. They are MS Blaster worm, SQL Slammer worm, Apache-Knacker worm and Conficker worm. The MS Blaster worm exploits a vulnerability in Microsoft's DCOM RPC interface. Upon successful execution, the MS Blaster worm retrieves a copy of the file msblast.exe from a previously infected host [22]. SQL Slammer worm exploits a buffer overflow vulnerability in Microsoft's SQL Server. Apache-Knacker worm are based on the real world Apache-Knacker exploit. Conficker exploits a stack corruption vulnerability, the MS08-067 server service vulnerability, to introduce and

```
**Algorithm CGNRS**($S$, $N$, $w$, $a$, $b$)
**Input:** $n$ sequences $S = \{y_1, \cdots, y_n\}$, $l$ normal sequences $N = N_1, N_2, \cdots, N_l$,
the width of signature $w$;
**Output:** worm signature NRS;
$S \rightarrow S_1, S_2, \cdots, S_m$;
For $i = 1$ to $m$
    $\{u = a$; $Flag = True$;
    While ($Flag == True$)
        {GeWS($S_i$, $N$, $u$, $w$);
        If $NRS_i$ and $S_i'$ are generated by algorithm GeWS, Then
            $\{Flag = FALSE$;
            Combining the set of sequences $S'$ and $S_i'$;}
        Else $\{u = u - 1$;
            If ($u < b$) Then
                $\{S_i' = NULL$; $Flag = FALSE$;}}}
If $S'$ is not null, then
    $\{NRS \leftarrow GNRS(S', w)$;
    return ($NRS$);}
Else return ("not generating signature");
```

Fig. 2. Algorithm CGNRS

execute shellcode on affected Windows system [23]. In our experiments, we apply polymorphism techniques to generate PWCS and PWVS for above 4 type worms.

4.1 Comparison of NRS and Other Signature Generated from Suspicious Flow Pool Without Noise Sequences

In the following experiments, NRS generated by algorithm CGNRS is compared with three worm signatures, PADS signature [24], token subsequence signature from Polygraph [16] and signature generated by using multiple sequence alignment algorithm (MSA) [6]. We use 200 MS Blaster worm variants and 200 SQL Slammer worm variants for generating signatures. Then we use 10000 correspondent worm sequences and 10000 normal flow sequences from real network as test variants. The false positive ratio and the false negative ratio are get respectively. The false positive ratio is defined as the number of normal flow sequences misclassified as worm variants divided by the total number of normal flow sequences. The false negative ratio is defined as the number of worm test variants misclassified as normal traffic divided by the total number of worm test variants. Experiments are run in the following two scenarios:

(1) Worm samples and worm test variants contain only PWCS. Since PWCS exhibits some constant subsequences, four kinds of worm signatures can all be generated. Because NRS with shorter signature length has better quality and so does PADS [24], the length of NRS and PADS is 10. The experi-

Table 2. The false positive ratio and the false negative ratio of different worm signatures when worm samples and test variants include PWVS variants

	Worm name	The false positive ratio	The false negative ratio
NRS	Blaster	0	0.0002
	Slammer	0	0
	Conficker	0	0
PADS	Blaster	0	0.2839
	Slammer	0	0.1346
	Conficker	0	0.0347
Polygraph	Blaster	None	None
	Slammer	None	None
	Conficker	None	None
MSA	Blaster	None	None
	Slammer	None	None
	Conficker	None	None

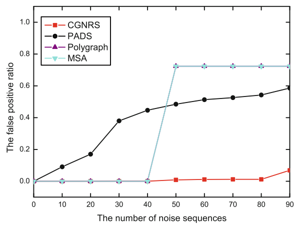
ment results show that these worm signatures are no false negative and false positive.

(2) Worm samples and worm test variants all contain PWVS. The experiment results are illustrated in Table 2.

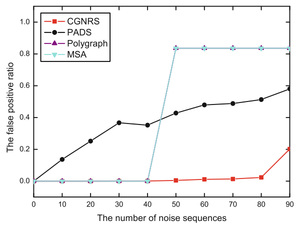
It is easy to see from Table 2 that NRS and PADS can be generated when worm samples include PWVS. Since PWVS can not be identified by finding constant subsequences, Polygraph [16] and MSA [6] can not generate signature for PWVS. In Table 2, the false positive ratio of NRS and PADS is 0. NRS and PADS can distinguish the normal flow sequences well. The false negative ratio of NRS is lower than that of PADS. The reason is that PADS is based on bytes themselves. If polymorphic techniques, such as Encryption techniques, are used in worm variants, PADS will suffer from difficulties to detect such worms. NRS is based on neighbor relationship, and it is more flexibility.

4.2 Comparison of NRS and Other Signature Generated from Suspicious Flow Pool with Noise Sequences

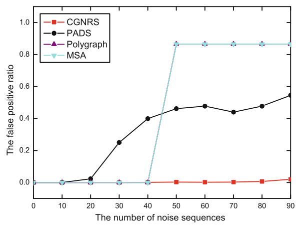
In the experiments, the suspicious flow pool includes 200 worm samples. When the number of noise sequences is l, we randomly replace l worm sequences with l noise sequences in the suspicious flow pool. Here worm sequences belong to PWCS. Noise sequences are generated by method discussed in [11]. Every noise sequence has some common substrings with one worm sequence of the suspicious pool. There also have some common substrings among noise sequences and these common substrings come from normal flow sequences. But noise sequence does not contain the true invariant parts of the worm. NRS with different length are generated by CGNRS from the suspicious flow pool. Other kinds of worm



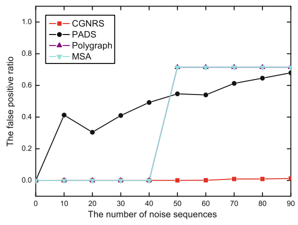
(a) Blaster



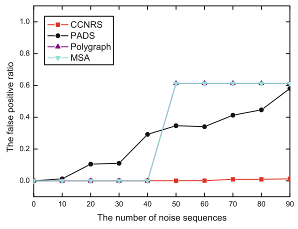
(b) SQL Slammer



(c) Apache-Knacker



(d) ATPhttpd



(e) Conficker

Fig. 3. The false positive ratio of CCNRS, PADS, Polygraph and MSA

signatures, including PADS, signatures of Polygraph and signatures of MSA, also are generated.

We use 10,000 worm variants and 10,000 normal flow sequences as test variants to measure NRS, where worm variants belong to PWCS. NRS is compared with other kinds of worm signatures. Comparison results of NRS and other kinds of signatures are illustrated in Figs. 3 and 4. In the experiments, the length of NRS and PADS is set to be 10.

From Fig. 3, it can be seen that when there are no noise sequences in suspicious flow pool, the false positive ratio of these worm signatures is 0. However, with the number of noise increasing gradually, the false positive ratio of PADS grows. PADS is collection of position-aware byte frequency distributions. If there are noise sequences in the suspicious flow pool, PADS will be generated by computing position-aware byte frequency of worm variants and noise sequences. Therefore, when the PADS is used to detect worms, PADS will classify some noise sequences into worm variants. When the number of noise adds to 50, signature generated by Polygraph and MSA also obtain higher false positive ratio. Because of the disturbance of craft noise, common substrings among noise sequences are extracted for composing worm signatures. If these signatures are used to detect noise sequences, noise sequences are considered as worm variants. Since CGNRS adopts color coding, NRS generated by CGNRS obtained lower false positive ratio.

Moreover, for signatures generated by Polygraph and MSA are common substrings of noise sequences, these signatures can not detect worm variants. So, the false negative ratio is 0 when they detect above 5 kind of worm variants. Since NRS and PADS are more flexible, they can detect correctly all worm variants. Therefore, we use one figure, Fig. 4, to show the false negative ratio of above 4 worm signatures in detecting different kind of worm variants. From the

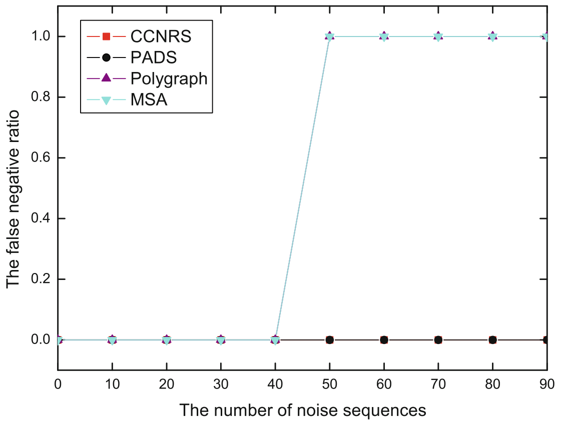


Fig. 4. The false positive ratio of CCNRS, PADS, Polygraph and MSA

Figs. 3 and 4, it can be seen that, compared with PADS, Polygraph and MSA, CGNRS is better in generating worm signature with high quality.

5 Conclusion

In this paper, we propose NRS signatures based on neighborhood relation for polymorphic worms. NRS are generated by GNRS algorithm based on distances between neighbor bytes. We perform extensive experiments to demonstrate the effectiveness of NRS. In order to deal with noise problem, we proposed a novel algorithm CGNRS by introducing color coding into our approaches. CGNRS is able to generate NRS signature automatically for polymorphic worms in the environments with noises. CGNRS are tested and compared with PADS, Polygraph and MSA. According to the results of comparison, we draw the following conclusions:

(1) If there are only PWCS in the suspicious flow pool without noise sequences, NRS, PADS, and signatures generated by Polygraph and MSA can be used to detect effectively polymorphic worm.
(2) When the suspicious flow pool without noise sequences contains PWVS, NRS can obtain lower the false negative ratio compared with PADS. Polygraph and MSA can not generate worm signature.
(3) When suspicious flow pool include noise sequences and PWCS, compared with PADS and signature generated by Polygraph and MSA, NRS by CGNRS can obtain lower false positive ratio and lower false negative ratio. Therefore, CGNRS is a better approach to generate signature for polymorphic worm.

Acknowledgment. This work is supported by National Natural Science Foundation of China under Grant No. 61202495 and No. 61573379.

References

1. Antonatos, S., Akritidis, P., Markatos, E.P., Anagnostakis, K.G.: Defending against hitlist worms using network address space randomization. Comput. Netw. **51**(12), 3471–3490 (2007). Elsevier
2. Bilge, L., Dumitras, T.: Before we knew it: an empirical study of zero-day attacks in the real world. In: Proceedings of ACM conference on Computer and communications security (CCS 2012), New Carolina, pp. 833–844, October 2012
3. Talbi, M., Mejri, M., Bouhoula, A.: Specification and evaluation of polymorphic shellcode properties using a new temporal logic. J. Comput. Virol. **5**(3), 171–186 (2009)
4. Stephenson, B., Sikdar, B.: A quasi-species approach for modeling the dynamics of polymorphic worm. In: IEEE Infocom, Barcelona, Catalunya, pp. 1–12 (2006)
5. Song, Y., Locasto, M.E., Stavrou, A., Keromytis, A.D., Stolfo, S.J.: On the infeasibility of modeling polymorphic shellcode. In: Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, Virginia, USA, pp. 541–551 (2007)

6. Tang, Y., Xiao, B., Lu, X.: Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms. Comput. Secur. **28**, 827–842 (2009). Elsevier, Available online 17 June 2009
7. Modi, C., Patel, D., Borisaniya, B., Patel, H., Patel, A., Rajarajan, M.: A survey of intrusion detection techniques in cloud. J. Netw. Comput. Appl. **36**(1), 42–57 (2013)
8. Wang, J., Wang, J.X., Chen, J.E., Zhang, X.: An automated signature generation approach for polymorphic worm based on color coding. In: IEEE ICC 2009, Dresden, Germany, pp. 1–6 (2009)
9. Tang, Y., Xiao, B., Lu, X.: Signature tree generation for polymorphic worms. IEEE Trans. Comput. **60**(4), 565–579 (2011)
10. Fogla, P., Sharif, M., Perdisci, R., Kolesnikov, O., Lee, W.: Polymorphic blending attacks. In: Proceedings of 15th USENix Security Symposium, Vancouver, B.C., Canada, pp. 241–256 (2006)
11. Perdisci, R., Dagon, D., Lee, W., Fogla, P., Sharif, M.: Misleading worm signature generators using deliberate noise injection. In: Proceedings of 2006 IEEE Symposium on Security and Privacy, Atlanta, GA, USA, pp. 17–31 (2006)
12. Comar, P.M., Liu, L., Saha, S., Tan, P.N., Nucci, A.: Combining supervised and unsupervised learning for zero-day malware detection. In: Proceedings of 32nd Annual IEEE International Conference on Computer Communications (INFOCOM 2013), Turin, Italy, pp. 2022–2030, April 2013
13. Portokalidis, G., Bos, H.: SweetBait: zero-hour worm detection and containment using low- and high-interaction honeypots. Comput. Netw. **51**(11), 1256–1274 (2007)
14. Ranjan, S., Shah, S., Nucci, A., Munafo, M., Cruz, R., Muthukrishnan, S.: DoWitcher: effective worm detection and containment in the internet core. In: IEEE Infocom, Anchorage, Alaska, pp. 2541–2545 (2007)
15. Cai, M., Hwang, K., Pan, J., Christos, P.: WormShield: fast worm signature generation with distributed fingerprint aggregation. IEEE Trans. Dependable Secure Comput. **5**(2), 88–104 (2007)
16. Newsome, J., Karp, B., Song, D.: Polygraph: automatically generation signatures for polymorphic worms. In: Proceedings of 2005 IEEE Symposium on Security and Privacy Symposium, Oakland, California, pp. 226–241 (2005)
17. Li, Z., Sanghi, M., Chen, Y., Kao, M., Chavez, B.: Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. In: Proceedings of IEEE Symposium on Security and Privacy, Washington, DC, pp. 32–47 (2006)
18. Cavallaro, L., Lanzi, A., Mayer, L., Monga, M.: LISABETH: automated content-based signature generator for zero-day polymorphic worms. In: Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems, Leipzig, Germany, pp. 41–48 (2008)
19. Bayoglu, B., Sogukpinar, L.: Polymorphic worm detection using token-pair signatures. In: Proceedings of the 4th International Workshop on Security, Privacy and Trust in Pervasive and Ubiquitous Computing, Sorrento, Italy, pp. 7–12 (2008)
20. Bayoglu, B., Sogukpinar, L.: Graph based signature classes for detecting polymorphic worms via content analysis. Comput. Netw. **56**(2), 832–844 (2012)
21. Wang, J., Wang, J.X., Sheng, Y., Chen, J.E.: Polymorphic worm detection using signatures based on neighborhood relation. In: Proceedings of the 11th IEEE International Conference on High Performance Computing and Communications, pp. 347–353 (2009)
22. CERT Advisory CA-2003-20: W32/Blaster worm, Computer Emergency Response Team (2003). http://www.cert.org/advisories/CA-2003-20.html

23. Leder, F., Werner, T.: Know your enemy: containing conficker to tame a Malware. The Honeynet Project (2009). http://honeynet.org
24. Tang, Y., Chen, S.: An automated signature-based approach against polymorphic internet worms. IEEE Trans. Parallel Distrib. Syst. **18**, 879–892 (2007)