PolyS: Network-based Signature Generation for Zero-day Polymorphic Worms

Sounak Paul and Bimal Kumar Mishra

Dept. of Information Technology, Birla Institute of Technology, Mesra, Ranchi, India
Dept. of Applied Mathematics, Birla Institute of Technology, Mesra, Ranchi, India

paul.sounak@gmail.com, drbimalmishra@gmail.com

Abstract

With growing sophistication of computer worms, it is very important to detect and prevent the worms quickly and accurately at their early phase of infection. Traditional signature based IDS, though effective for known attacks but failed to handle the zero-day attack promptly. Recent works on polymorphic worms does not guarantee accurate signature in presence of noise in suspicious flow samples. In this paper we propose PolyS, an improved version of Hamsa, a network based automated signature generation scheme to thwart zero-day polymorphic worms. We contribute a novel architecture that reduces the noise in suspicious traffic pool, thus enhancing the accuracy of worm's signature. Also we propose a signature generation algorithm for successfully matching polymorphic worm payload with higher speed and memory efficiency. Analysis shows that our system is fast, accurate, attack-resilient and capable of generating quality signature with low false positive and false negative.

Keywords: *polymorphic worm, worm signature, honeypot, flow-classifier, false positive*

1. Introduction

Worms are the malicious codes which propagate themselves without human intervention. They are capable of replicating themselves in the host, when required. Worms first find targets using different methods available. Most popular among them are blind-scan, hit-list, topological and web search. Once the target is found, the worm sends a copy to the next victim. Depending on the nature of the worm, propagation may be straightforward or worm payload may be downloaded from internet or from the infected host. Worm payload may also be embedded with legitimate traffic. Propagation may be epidemic as observed in many research [1, 2, 3]. It can spread faster than human response. Depending on the type of transmission, worm may either require TCP connection or UDP connections. TCP worms are found to be latency limited, whereas UDP worms are bandwidth limited [4]. The payload of worm has been evolved from traditional monomorphic to much complex polymorphic and metamorphic. Polymorphic worms can change their payload dynamically, but the functionality remains same. Metamorphic worms on the other hand not only change their appearance but they

This paper is an extended and revised version of our paper "Honeypot based signature generation for defense against polymorphic worms in networks," in proc. of IEEE IACC, Feb, 2013, DOI: 10.1109/IAdCC.2013.6514213

change their behavior too on every infection, making it difficult for traditional IDSes [5, 6] to detect them.

Most of the defense mechanism developed against these worms work reactively after full or partial damage has already occurred. Research community has continuously tried to build and improve IDS to defend against malicious code attack. Research on worm defense may be broadly classified into two categories: Detection and Containment. Further Detection algorithm can be classified into two categories; Anomaly based detection and Signature based detection [4, 10].

Anomaly based system [11, 12] observe the traffic statistics and host behavior to detect previously unknown worms. To detect malicious traffic it require to understand normal traffic behavior, which in turn require efficient training which in real time scenario is much difficult to achieve, as the behavior of legitimate activities are largely unpredictable. Though this method is found to be effective in detecting unknown worms [4], it generates high false alarm.

Signature based detection [13, 14] does not take interest in propagation or transmission scheme; neither they look into host behavior. They look for specific byte sequence in each packet. If any match found with signature stored in database it will be identified as malicious. When compared with anomaly based detection method signature based approach found to be much simpler and easier to implement online in real time. The difficulty is every signature needed to be stored in a signature pool. Over a period of time, the signature pool grows larger, making it complex for comparing any new signature with the existing one in the database [4]. Moreover the early signature based methods are mostly manual; therefore this method consumes huge system resource, reducing the overall performance of the system and largely depends on human expertise of finding signature.

Another problem with signature based approach is that it can detect only known worms with the signature generated manually by experts by carefully studying the network traces [7, 8, 15]. The slow pace of manual signature generation led the researcher focus on automatically generating signature of both known as well as unknown worms. Automated signature generation scheme need to address the following difficulties:

- It must understand well the definition of malicious and legitimate traffic, so that false alarm is minimum. There lacks a systematic solution to this problem.
- The signature must be flexible enough to defend against polymorphic worms that change their appearance in every infection. So single payload substring may not be invariant across worm connections.
- It should take care of system resource such as CPU time of generating signature and comparing them with network traffic. Signature storage space must not be stressed.
- The Signature must be reliable enough to detect wide variety of zero day attacks.
- The system must be noise tolerant, attack-resilient and accurate.
- The system must produce signature with low false positive and false negative.

Several recent research focus on defense against zero-day polymorphic worms [7, 8, 9, 26], but it remains largely an open problem due to diversity of the modern worms which employs multiple vulnerability in much faster rate than human response to control their spread. Our research attempts to address the above mentioned problems,

and provide a systematic solution to the whole problem, from isolating malicious traffic to automatically generating signature for polymorphic worms with lowest possible false positive and high coverage (true positive).

We have used two tier flow-classification to minimize the presence of normal traffic (noise) in suspicious flow pool. The first layer is a standard flow-classifier. The existing flow classifier to identify suspicious flow may be a port-scan detection method [21] or byte-frequency distribution [12, 25]. The second layer uses honeypot technology [17, 18] to capture and analyze malicious traffic and send them to signature generation module for automatic generation of polymorphic worms signature. We presents a token based signature generation algorithm, similar to hamsa [8], but our algorithm is much faster and memory efficient.

The rest of the paper is organized as follows. Section II discusses the strength and weakness of related works. Section III discusses the anatomy of polymorphic worms; Section IV proposes a novel architecture and described data control and data capture mechanism. Section V formally defines the problem. Section VI presents the signature generation algorithm for polymorphic worms, Section VII draws conclusion.

2. Related Work

Several algorithms have been proposed for anomaly based worm detection and signature based detection. But none can cover entire range of worms. A hybrid system which check for network anomaly and look for signature in worm's payload may give complete and broader view of detection with a honeypot system to collect and analyze worm activities.

One of the early work in this category is Honeycomb [20], proposed by Kreibich and Crowcroft. Honeycomb combines honeypot technology with automatic signature generation scheme. Honeycomb has implemented an extended version of open source honeypot *honeyd* for this purpose. Suffix tree has been used to implement the Longest Common Substring (LCS) algorithm in this case. Problem with Honeycomb is that it generates single contiguous substring of sufficient length of worm's payload to match the worms. This assumption is insufficient for polymorphic worm detection [7].

Hyang-Ah Kim and Brad Karp describes autograph [21] a distributed, automated worm signature generation scheme to detect polymorphic worms. Autograph takes the input from cross DMZ traffic that includes benign traffic and selects suspicious traffic using certain heuristic. The suspicious packet passes through flow reassembly. Payloads partition is done into different content block using COPP algorithm. The content blocks are analyzed and autograph selects most frequently occurring byte sequence across the flows in suspicious flow pool. Prevalence histogram is generated for each content block which acts as worm signature. Similar to Honeycomb, Autograph also relies on generating single contiguous substrings as signature of a worm's payload to match the worm instances. But polymorphic worm may change their payload in each infection. Autograph system fails to address this problem.

Recently there has been a major focus shift towards generation of signature for polymorphic worms. The signatures in this category are broadly classified as exploit-specific and vulnerability-based. Exploit-specific signatures are based on feature specific to worm's implementation. They are mostly content based. Vulnerability-based signature looks for characteristics of the vulnerability the worm exploits. Depending on deployment both the exploit specific and vulnerability based signature may be classified as host-based and network based. Although host based signature

generation is more accurate, network based approach can have better coverage and ability of protecting all user in the network as a whole, enabling detecting the worm fast and at early stage [26]. The classification is shown in Table 1 below.

Table 1. Polymorphic Worm Signature Generation Schemes

Signature Properties	Signature Generation Approach	
	Network-Based	Host-Based
Exploit - Specific	Polygraph [7] Hamsa [8] PADS [15] Nemean [35] CFG [36] LISABETH[22] CCSF[31] Polytree [9]	Taint Check [37] DACODA [38]
Vulnerability - driven	LESG [26]	COVERS [39] Packet Vaccine [40] Vulnerability Signature [41] Vigilante [42]

Our signature generation scheme is exploit-specific, network-based. We present below some of state-of-art work in this category.

J. Newsome et al. address the problems of autograph, honeycomb as discussed above in Polygraph [7]. Polygraph generates multiple disjoint content substrings to match all instances of a polymorphic worm. They observed that multiple invariant substrings often present in all variant payload of a polymorphic worm. Such invariant substrings include protocol framing bytes, return addresses and in some cases obfuscated code [7]. Based on these facts, polygraph divides signatures into tokens, a contiguous byte sequence. The system extract tokens automatically and represents each suspicious flow as a sequence of tokens. Polygraph present three classes of signature suite: namely conjunction signature, token subsequence signature and Bayes signature.

Hamsa [8], a network based automated signature generation scheme has shown substantial improvement over polygraph in terms of speed, accuracy and attack resilience. Hamsa follows the polygraph token based approach, but replaced suffix tree method of token extraction with light weight suffix array method [24] which increases the speedup of token extraction process 100 fold than Polygraph. Hamsa uses a greedy approach of generating multiset token signature.

Lorenzo Cavallaro et al. proposed Lisabeth [22], an improved version of Hamsa [8] in terms of attack resilience and accuracy.

All these systems are effective in detecting polymorphic worms in normal situation. But they can be evaded by injecting well crafted fake anomalous flows into normal traffic, thereby misleading signature generation process [30].

Polygraph and hamsa signature are both vulnerable to suspicious and innocuous pool poisoning attack [22]. Compared to hamsa, Lisabeth shows better resilience against these two attacks. Moreover token extracted in presence of noise are probably fragments, therefore polygraph and hamsa will generate wrong signature in presence of noise in suspicious pool [31]. Hamsa and LESG also proved that presence of noise in

suspicious pool makes the problem NP hard [8, 26]. Still there is little or no attention is paid in filtering noise in suspicious pool [30].

Y. Tang *et al.*, proposed position aware distribution signature (PADS)[15], a content based signature generation scheme for polymorphic worms. Expectation Minimization and Gibs Sampling algorithms are used to compute PADS from polymorphic worm samples. It claims to bridge the gap between traditional signature scheme and statistical anomaly- based approach. While hamsa and polygraph signature based on invariant part in polymorphic worms PADS consider both invariant and variant part of worm's payload to find signature. However PADS can't assure the accuracy of signature in presence of noise [8].

Y.Tang, Bin Xiao proposed Polytree [9], a signature generation for polymorphic worms using multiple sequence alignment algorithm. The signature is in the form of simplified regular expression. This method organizes signature into a tree structure. Signature in a child node is more specific than signature in parent node. Signature generated using polytree is more accurate and attack resilient than polygraph and hamsa however noise in suspicious flow pool still a major deterrent in computing correct signature.

Most of the signature generation approaches discussed above generate signature for polymorphic worms but encounter problem in presence of noise in suspicious pool. In this paper we propose PolyS, a network based a signature generation for zero day polymorphic worms. Our system is an improved version of Hamsa in terms of accuracy, speed of signature generation, memory usage and attack resilience.

3. Structure of Polymorphic worms

Typically a polymorphic worm and its instances are composed of following components [7, 8, 22, 31].

Protocol Framing: Protocol framing is necessary for branch down the code execution path, where software vulnerability exist. The protocol framing string is invariant across all instances of polymorphic worms.

Example of such protocol framing string in Apache –Knacker exploits: "GET", "HTTP/1.1", and "Host:" twice [7].

Return Address: Return address or function pointers are the values used to overwrite a jump target to redirect the server execution [7]. Typically a 32 bit integer, of which first 23-bit are normally same across all worm samples [8]. Return address is another invariant part in polymorphic worms.

Exploit Code: These invariant bytes are necessary for abusing vulnerability. It also activates decryption routines and ensures identical malicious activities in all attacks.

Encrypted worm code (Payload): It contains the code to perform malicious activities. In presence of strong encryption routines, the worm payloads take different values in different infection.

Decryption Routine: Its function is to decrypt the encrypted payload by decryption key and passes the control to worm's code to start execution. Decryption routines are obfuscated in different instances of polymorphic worms.

Decryption Key: Worm payload is encrypted by polymorphic engines by different keys in different instances. To decrypt the worm's payload, corresponding decryption key is required.

Wild Card bytes: These bytes may take any values without affecting the functioning of worms and their spreading capabilities.

In summary, polymorphic worms have two classes of bytes; invariant and variant bytes. Invariant bytes remain same across all instances of the worms while variant bytes change its value in every infection attempt. Typically variant bytes are protocol framing string, exploit code and return address. The other components are in general variant across different instances of a polymorphic worm.

4. System Model

4.1. Architecture

Figure 1 shows the typical deployment of our PolyS system that is similar to the basic framework of other network based signature generation approaches like Polygraph [7], Hamsa [8], Polytree [9], and LESG [26]. The difference of our architecture from above mentioned system is that our system uses flow classifiers in two levels. The system first sniffs the traffic from network at router/gateways/switches using a network tap. The traffic is classified as different application level protocols (TCP/UDP) based on port number and other protocol identifiers. Next for each (port, protocol) pair we filter out known worms and then using an existing worm flow classifier [12, 21, 25] at the first level, separate remaining traffic as suspicious traffic pool and normal traffic reservoir. In absence of noise constructing multiset token signature is a polynomial time problem as observed by Hamsa and LESG. However we can't expect the flow classifier to be perfect. There must be some normal traffic (noise) mixed with the worm traffic pool, thereby increasing the computational complexity of signature generation significantly [8, 26]. To further reduce the noise in suspicious traffic pool we have used second layer of flow classifier, *i.e.*, a double honeypot system. Before discussing architecture of our double-honepot system, we give a brief introduction of honeypot here. "A honeypot is an information system resource whose value lies in being probed, attacked or compromised" [16]. Depending on the level of interaction, honeypot can be classified as low-interaction honeypot and high-interaction honeypot. Low-interaction honeypot works by simulating operating- system and other services. It also populate unused IP address There are several open source and commercial low-interaction available such as `honeyd` [27], `KFSensor` [28]. High interaction honeypot on the other hand works with real operating system and services. Advantage of high-interaction honeypot system is that it can capture useful information about malicious activities of even unknown worms in detail [4, 19]. There are very few high interaction honeypot implementation is available. Notable among them are `Argos` [29].

The architecture of our double honeypot system is similar to the basic framework of [34, 29]. It is composed of two independent arrays of honeypot – inbound (honetrap-1) and outbound (honeytrap-2) honeypot. Each honeypot array consists of multiple honeypots capable of running in physical machines or virtual machines simulated by same computer. The idea is to leave a flexibility to implement physical or virtual honeypot as per requirements of the system. This architecture of double honeypot system is an open architecture so that we can plug-in different types of honeypot (low

or high interaction) as per implementation. At this point we can have two options. First the inbound honepot is implemented as high-interaction honepot and outbound honepot is implemented as low interaction honepot, second both inbound and out bound honepot system implanted as high interaction honepot. Our goal is to attract worm in inbound honeypot which will subsequently tries to make or connection with outbound honeypot.

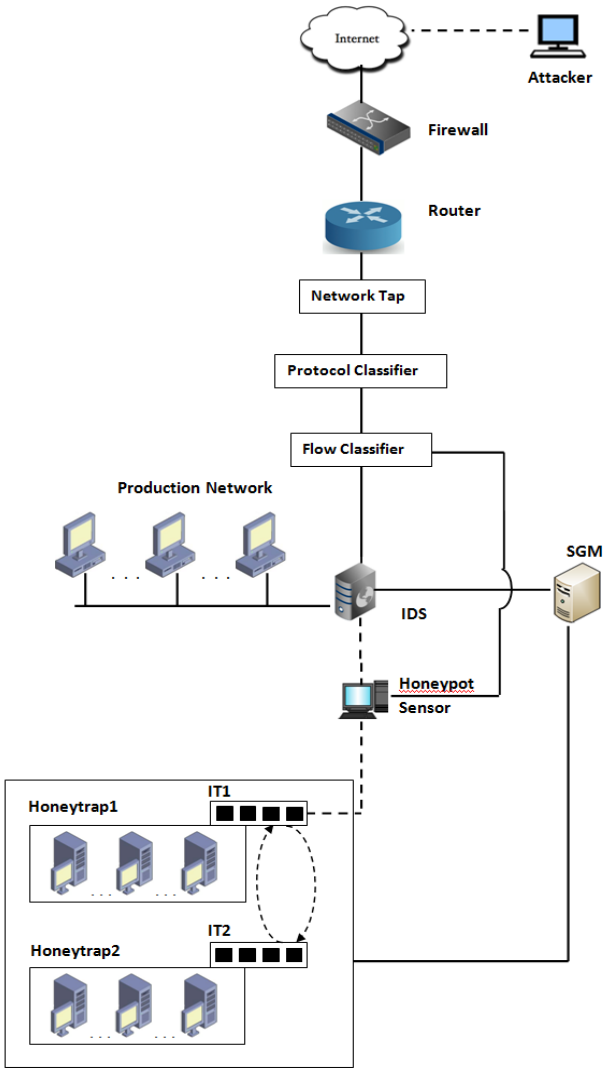

Figure 1. System Architecture

4.2. Data Control

A network tap sniffs the traffic from networking devices (Such as router, switches, gateways). The traffic then passes through protocol classifier and worm flow classifier. One level of flow classification is done at this stage. The suspicious traffic is then directed towards honepot system for further refinement. The layer two bridging device isolates the entire honeypot system from the rest of the network. The same device force all traffic going to and from honeypot system, first pass through as “invisible” layer two bridge. This bridge allows the attacker come into the honeypot system, but decide its

fate if it want to come out of the honeypot [17]. The attacker can't trace any IP address, MAC-Address, Time to Live (TTL) reducing or routing path as the bridge functions in layer two.

Worms are self replicating. When a worm infects one host it tries to infect next vulnerable host by replicating itself and sending a copy of it. In an attempt to find next victim, worm initiate outbound connections from the compromised host, *i.e.*, when honeytrap1 is compromised it will try to make an outbound connection, to the intent of spreading the attack in other systems. The traffic is redirected to honeytrap2 through the internal translator1 (IT1) implemented at router. The honeytrap2 also does the same thing when the worm tries to make an outbound connection. The traffic is redirected to honeytrap1 through internal translator2 (IT2). We can decide a threshold of the number of outbound connections allowed. We set this threshold as 10 connections per hour. We will block further connections. This will serve two purposes. First we will be able to capture enough instances of malicious traffic, second it will reduce the chances of denial of service attacks. The disadvantage of this method is it can still launch attack in outbound limits. Instead of blocking all the outbound connections at outbound threshold we will allow some of them periodically to pass through honeynet sensor to forward them to the IDS which will modify and disable the attacks. Attacker may see the failure but will unable to recognize the reason. This way we can capture necessary details regarding the attack, at the same time reducing the chances of compromising remote systems.

4.3 Data Capture

To analyze the malicious traffic effectively and generate signature from it, we require different types of information related to the attack. A single layer of defense can't capture all necessary information. Therefore we have implemented multilayer data capturing system in our proposed system. Entry layer of data capture is the firewall. Firewall logs preliminary header information such as source and destination address, attack timestamp, source and destination port number, *etc.* The second layer of data capture is network tap which sniffs the network data, protocol classifier classifies traffic based on port number and application level protocol (TCP, UDP, ICMP). The next layer is flow classifier, which separates worm traffic from normal traffic. Suspicious traffic then redirected to honeypot system. The honeypot is the final layer of data capture in our system. It further filters the noise from suspicious traffic and captures different types of worm data, necessary for generating their signature at signature generator module (SGM).

5. Problem definition

Given a suspicious flow pool X and normal flow pool Y as a input to the signature generation algorithm, our goal is to find a signature S which cover many flows in X and not so many in normal flow pool Y, so that false positive rate of signature S, determined by normal pool, $FP_S = \frac{|Y_S|}{|Y|}$ must be low and coverage of suspicious flow, i.e. true positive rate of S, $COV_S = \frac{|X_S|}{|X|}$ is high.

Our problem formulation is based on the fact that multiple invariant strings must present in all variant of a polymorphic worm [7, 8, 9, 22]. We consider each of these

strings as tokens. A token is a byte sequence that is present in a significant member of flows. A signature is a set of tokens with their occurrence number in suspicious flows. Formally a signature takes the following form:

$\{(t_1, n_1), (t_2, n_2), (t_3,n_3)\ldots\ldots\ldots(t_k,n_k)\}$, where t_j is the token and n_j is the number of occurrence.

A signature is said to match a flow F if it contains at least n_j copies of t_j as substring. For example tokens of Apache –Knackers worm are [7][8][33]

{(\'XFF\XBF', 1), ('GET', 1), ('\r\n', 5), ('HTTP/1.1\r\n', 1), ('\r\n Host', 2)}

A signature is a set of (token, number of occurrence) taken in any order.

6. Signature Generation

Our signature generation algorithm follows the following steps.

Token Extraction:

Like Polygraph [7], Hamsa [8], Tokenpair [33] our algorithm also extract tokens with minimum length $l_{\min}$ bytes that occur in atleast λ fraction in suspicious traffic. The length parameter expected to be at least two. If we set the minimum length below two almost every character will be included as a token for its frequent occurrence in a suspicious pool [9]. Hamsa used deep-shallow-sort [24], a suffix array based algorithm to extract tokens. The use of suffix array has increased the speed of token extraction manifold. We have used suffix array induced sorting (SA-IS) [32] algorithm for token extraction. The use of this suffix array algorithm further enhanced the speed of token extraction and memory efficiency [32].The optimized code of SA-IS is available in [43]. The author of this paper compared performance of SA-IS algorithm with deep-shallow sorting [24], that is used in token extraction and false positive calculation of Hamsa [8]. The experiment show that SA-IS is most time and space efficient among all linier time suffix array construction algorithm. Token extracted may appear more than once in a flow. We note the number of occurrence of a token in a flow and store them as a pair (Token, number of occurrence).

Subtoken Consideration:

We define t_1 as subtoken of t_2, if $t_1 \neq t_2$ and t_1 is a substring of t_2. We include every subtoken in signature if it satisfies the minimum length constraint and its coverage is larger than λ. Many of the token based signature generation schemes argued in favor of eliminating subtokens from signature. But we have observed by eliminating the subtokens we may miss some crucial invariants.

Sorting Tokens:

Sort the list of tokens in descending order of their length. We assign each (token, no of occurrence) pair an unique identifier. For each identifier we build a list of suspicious flows in which it occurs.

False Positive Calculation:

To estimate false positive of each identifier in partial signature set we build suffix array of normal traffic pool using the SA-IS algorithm [32], and store them in memory. We use binary search to find a token in suffix array. Compared to hamsa our suffix array algorithm is both time and space efficient. However we use the same procedure as Hamsa [8] to calculate the false positive rate of each token in partial signature list.

The flow classification in our architecture used double filtration technique. Therefore it is expected that the suspicious flow will contain mostly worms. The worm flows are constrained to include a multi-set of invariant in any order.

Say this multi-set invariant set is

$$T = \{\hat{t}_1, \hat{t}_2, \hat{t}_3 ....., \hat{t}_k\}$$

Such that, $FP_{\{\hat{t}_i\}} \le FP_{\{t_i\}} \quad \forall i$

$$FP_{\{\hat{t}_1, \hat{t}_2\}} \le FP_{\{\hat{t}_1, t_i\}} \quad \forall i > 1$$

$$FP_{\{\hat{t}_1, \hat{t}_2, ...\hat{t}_j\}} \le FP_{\{\hat{t}_1, \hat{t}_2, ...\hat{t}_{j-1}, t_i\}} \quad \forall j \forall i > j-1$$

This means $\hat{t}_1$ is the token with least false positive value among all tokens available in the list. $\hat{t}_1$ in conjunction with $\hat{t}_2$ is has least false positive than in conjunction with any other token available in the list. This greedy process will continue till we produce a signature S such that false positive of it is less than a threshold false positive rate and coverage is more than minimum threshold coverage, *i.e.*,

$FP_S \le FP_{\max}$ and $COV_S \ge COV_{\min}$

7. Conclusion

In this paper we propose PolyS, a network based, automated signature generation scheme for zero-day polymorphic worms. Our system achieves significant improvement in speed of signature matching and memory usage efficiency over hamsa [8], the state-of-art of content based, automated signature generation for polymorphic worms. While hamsa proved that, problem of generating signature in presence of noise is NP-Hard, we have tried to reduce noise significantly in suspicious traffic though our novel architecture. We have also used a faster and memory efficient algorithm for signature generation. The signature generated by PolyS, can easily be deployed in IDSes to enhance their capabilities of defense against most sophisticated polymorphic worms.

As future work currently we are preparing a test bed based on our architecture. Both the hamsa [8] and SA-IS [32, 43] implementation codes are available in public domain, therefore it will be easy to compare the efficacy of both the systems in real production network. Test may be carried out with synthetically generated polymorphic worms.

Acknowledgements

This work was partially supported by Birla Institute of Technology, Mesra, Ranchi, India and Birla Institute of Technology, International centre, Muscat, Sulatanate of Oman (Waljat College of Applied Sciences). The authors also thank the anonymous reviewers for their constructive comments and feedback in improving this paper.

References

[1] B. K. Mishra and D. K. Saini, "SEIRS epidemic model with delay for transmission of malicious objects in computer network", Applied Mathematics and Computation, Elsevier, vol. 188, **(2007)**, pp. 1476–1482.

[2] B. K. Mishra and S. K. Pandey, "Fuzzy epidemic model for the transmission of worms in computer network", Nonlinear Analysis: Real World Applications, Elsevier, vol. 11, **(2010)**, pp. 4335-4341.

[3] O. A. Toutonji, S. -M. Yoo and M. Park, "Stability analysis of VEISV propagation modeling for network worm attack", Applied Mathematical Modelling, Elsevier, vol. 36, **(2012)**, pp. 2751–2761.
[4] P. Li, M. Salour and X. Su, "A survey of internet worm detection and containment", Communications Surveys & Tutorials, IEEE, vol. 10, **(2008)**, pp. 20-35.
[5] V. Paxson, "Bro: a system for detecting network intruders in real-time", Computer Networks, vol. 31, **(1999)**, pp. 23-24.
[6] M. Roesch, "Snort: The lightweight network intrusion detection system", **(2001)**, http://www.snort.org/.
[7] J. Newsome, B. Karp and D. Song, "Polygraph: Automatically generating signatures for polymorphic worms", In proc. IEEE Security and Privacy Symposium, **(2005)**.
[8] Z. Li, M. Sanghi, Y. Chen, M. Kao and B. Chavez, "Hamsa: Fast signature generation for zero-day polymorphic worms with provable attack resilience", in Proc. IEEE S&P, **(2006)**, pp. 33–47.
[9] Y. Tang, B. Xiao and X. Lu, "Signature Tree Generation for Polymorphic Worms", IEEE transactions on computers, vol. 60, no. 4, **(2011)**.
[10] P. Vinod, V. Laxmi and M. S. Gaur, "Survey on Malware Detection Methods, In proc. Hackers workshop, IIT-Kanpur, **(2009)**.
[11] C. Kruegel and G. Vigna, "Anomaly Detection of Weh-based Attacks", in Proc. ACM Conference Computer and Communication Security, ACM , **(2003)**, pp. 251-261.
[12] K. Wang and S. J. Stolfo, "Anomalous Payload-based Network Intrusion Detection", in proc. 7th Infemtional Symposium on Recent Advances in Intrusion Detection (RAID '2004), **(2004)**.
[13] U. Lindqvist and P. Porras, "Detecting Computer and Network Misuse through the Production-Based Expert System Toolset P-BEST", in Proc of Symposium on Security and Privacy, **(1999)**.
[14] K. Ilgun, R. Kemmerer and P. Porras, "State Transition Analysis: A Rule-based Intrusion Detection Approach", IEEE Trans. Software Eng., vol. 2, **(1995)**, pp. 181-199.
[15] Y. Tang and S. Chen, "An Automated Signature-Based Approach against Polymorphic Internet Worms", IEEE Transaction on Parallel and Distributed Systems, **(2007)** July, pp. 879-892.
[16] L. Spitzner, "Honeypots: Tracking Hackers", Addison-Wesley, **(2002)**, www.tracking-hackers.com/book.
[17] L. Spitzner, "The Honeynet Project: Trapping the Hackers", In proc. IEEE S&P, **(2003)**, pp. 15-23.
[18] Y. Tang and S. Chen, "Defending against Internet Worms: A Signature-Based Approach", In Proc. IEEE INFOCOM, **(2005)**.
[19] I. Mokube and M. Adams, "Honeypots: Concepts, Approaches, and Challenges", In proc. ACMSE, **(2007)**, pp. 321-326.
[20] C. Kreibich and J. Crowcroft, "Honeycomb - creating intrusion detection signatures using honeypots", In Proc. of the Workshop on Hot Topics in Networks (HotNets), **(2003)**.
[21] H. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection", In USENIX Security Symposium, **(2004)**.
[22] L. Cavallaro, A. Lanzi, L. Mayer and M. Monga, "LISABETH: Automated Content-Based Signature Generator for Zero-day Polymorphic Worms," Proc. of the fourth international workshop on Software engineering for secure systems, Leipzig, Germany, 2008.
[23] Mohammed, M. M. Z. E., H. A. Chan, Ventura N., "Honeycyber: Automated signature generation for zero-day polymorphic worms", IEEE Military Communications Conference, MILCOM 2008, **(2008)**.
[24] G. Manzini and P. Ferragina, "Engineering a lightweight suffix array construction algorithm", Algorithmica, vol. 40, no. 1, **(2004)**.
[25] K. Wang, G. Cretu and S. J. Stolfo, "Anomalous Payload-based Worm Detection and Signature Generation", in proc. 8th Infemtional Symposium on Recent Advances in Intrusion Detection (RAID '2005), **(2005)**.
[26] L. Wang, Z. Li, Y. Chen, *et al*., "Thwarting Zero-day Polymorphic Worms with Network-Level Length-Based Signature Generation", IEEE/ACM transactions on networking, vol. 18, no. 1, **(2010)**, pp. 53-65.
[27] N. Provos, "Honeyd-A Virtual Honeypot Daemon", in 10th DFN-CERT worshop, Hamburg, Germany, **(2003)** February.
[28] http://www.keyfocus.net/kfsensor/.
[29] G. Portokalidis and H. Bos, "SweetBait: Zero hour worm detection and containment using low and high interaction honeypots", Elsevier Computer Networks, Int. Journal of Telecommunications and Networking, vol. 51, Issue 5, **(2007)**, pp. 1256-74.
[30] R. Perddisci, *et al*., "Misleading worm signature generators using deliberate noise injection", in Proc IEEE S&P, **(2006)**, pp. 17-31.
[31] J. Wang, *et al*., "An automated signature generation approach for polymorphic worms based on color coding", in proc. of IEEE ICC, **(2009)**.
[32] G. Nong, S. Zhang and W. Hong, "Two efficient algorithm for linear time suffix-array construction", IEEE transactions on computers, vol. 60, no. 10, **(2011)**, pp. 1471-1484.
[33] B. Bayoglu and I. Sogukpinar, "Polymorphic worm detection using Token-Pair signature", in proc. of ACM SecPerU-08, **(2008)**, pp. 7-13.

[34] Y. Tang and S. Chen, "Defending against internet worms: A Signature-based approach", in proc of IEEE Infocom, **(2003)**.
[35] V. Yegneswaran, *et al.*, "An architecture for generating semantic-aware signatures", in proc. USENIX Security Symp., **(2005)**, pp. 7.
[36] C. Kruegel, *et al.*, "Polymorphic worm detection using structural information of executables", in Proc. RAID, **(2005)**, pp. 207–226.
[37] J. Newsome and D. Song, "Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software", presented at the NDSS, **(2005)**.
[38] J. R. Crandall, Z. Su and S. F. Wu, "On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits", in Proc. ACM CCS, **(2005)**, pp. 235–248.
[39] Z. Liang and R. Sekar, "Fast and automated generation of attack signatures: A basis for building self-protecting servers", in Proc. ACM CCS, **(2005)**, pp. 213–222.
[40] X. Wang, *et al.*, "Packet vaccine: Black-box exploit detection and signature generation", in Proc. ACM CCS, **(2006)**, pp. 37–46.
[41] D. Brumley, *et al.*, "Towards automatic generation of vulnerability based signatures", in Proc. IEEE Security Priv. Symp., **(2006)**, pp. 2–16.
[42] M. Cost, *et al.*, "Vigilante: End-to-end containment of Internet worms", in Proc. ACM SOSP, **(2005)**, pp. 133–147.
[43] Y. Mori, "SAIS-An Implementation of the Induced Sorting Algorithm," http://yuta.256.googlepages.com/sais, 2008.

Authors

Sounak Paul is an Assistant professor with the Department of Information Technology, Birla Institute of Technology (BIT), Mesra, Ranchi, India, presently deputed at Birla Institute of Technology, International centre, Muscat, Oman (Waljat College of Applied Sciences). He received his Master in Technology in Computer science and Engineering from Indian Institute of Technology (IIT), Guwahati, India. He earned his Master in Computer Applications (MCA) from Birla Institute of Technology, Mesra, Ranchi, India. His research interests include network security specially malware analysis and defense, intrusion detection and security in wireless sensor networks.

Bimal Kumar Mishra is a Professor with the Department of Applied Mathematics and Associate Dean faculty and sponsored research, Birla Institute of Technology (BIT), Mesra, Ranchi, India. He received his Master degree in Operational Research from University of Delhi, India (1992) and Masters in Mathematics too. He earned his Ph. D. degree from Vinoba Bhave University, Hazaribag, India (1997) and subsequently earned his D.Sc. Degree from Berhampur University, Orissa, India in 2007. His research interests include Nonlinear Analysis specifically mathematical modeling of cyber attack and its defense. He is editor in chief of International Journal of mathematical modeling, Simulations & Applications and International Journal of mathematical modeling and Computing. He is the member in editorial board of several international Journals. He has published more than 90 research papers in journals of repute and conference proceedings. Presently he is working in the area of cyber attack/crime and its defense mechanism.