An Automated Signature Generation Approach for Polymorphic Worms Using Principal Component Analysis

Mohssen M. Z. E. Mohammed[1], H. Anthony Chan[1], Neco Ventura[1]
Mohsin Hashim[2], Eihab Bashier[2]
University of Cape Town, South Africa[1]
University of Khartoum, Sudan[2]

Abstract

Internet Worms pose a major threat to Internet infrastructure security. Security experts manually generate the IDS signatures by studying the network traces after a new worm has been released, a rather costly, laborious, and time consuming task. In this paper we propose automated signature generation system for polymorphic worms. We apply Principal Component Analysis (PCA) to determine the most significant substrings that are shared between polymorphic worm instances and use them as signatures. The experimental results show that the PCA has successfully detected polymorphic worms with zero false positives and low false negatives.

1. Introduction

Worms are computer programs that self-replicate without requiring any human intervention, by sending copies of their code in network packets and ensuring the code is executed by the computers that receive it. When computers become infected, they spread further copies of the worm and possibly perform other malicious activities. A polymorphic worm is a worm that changes its appearance with every instance [1].

It has been shown that multiple invariant substrings must often be present in all variants of worm payload. These substrings typically correspond to protocol framing, return addresses, and in some cases, poorly obfuscated code [8]. Intrusion detection systems serve three essential security functions: they monitor, detect, and respond to unauthorized activities. There are two basic types of intrusion detection: host-based and network-based. Host-based IDSs examine data held on individual computers that serve as hosts, while network-based IDSs examine data exchanged between computers [17, 18]. Security experts manually generate the IDS signatures by studying the network traces after a new worm has been released. Unfortunately, this job takes a lot of time. Researchers have recently given attention to automating the generation of signatures for IDSs to match worm traffic.

Our research is based on Honeypot technique. Developed in recent years, honeypot is a monitored system on the Internet serving the purpose of attracting and trapping attackers who attempt to penetrate the protected servers on a network. Honeypots fall into two categories. A high-interaction honeypot such as (Honeynet) operates a real operating system and one or multiple applications. A low-interaction honeypot such as (Honyed) simulates one or multiple real systems. In general, any network activities observed at honeypots are considered suspicious [1, 10].

Security experts need a great deal of information to perform signature generation. Such information can be captured by tools such as honeynet. Honeynet is a network of standard production systems that are built together and are put behind some type of access control device (such as a firewall) to watch what happens to the traffic [1]. We assume the traffic captured by honeynet is suspicious. Our system reduces the rate of false alarms by using honeynet to capture traffic destined to a certain network. This paper is organized as follows: Section 2 reviews worm anatomy and

techniques. Section 3 discusses the related work regarding automated signature generation systems. Section 4 introduces the proposed system architecture to address the problems faced by current automated signature systems. Signature generation algorithms for Polymorphic Worm will be discussed in section 5. Principal Component Analysis (PCA) processes will be discussed in Section 6. Section 7 gives the details of the experiments followed by the results. Section 8 concludes the paper.

2. Anatomy and Examples for Worm

A worm may have any or all of the components stated in [13:

- Reconnaissance: The worm network has to hunt out other network nodes to infect. This component of the worm is responsible for discovering hosts on the network that are capable of being compromised by the worm's known methods.

- Attack Components: The Attack Components are used to launch an attack against an identified target system. Attacks can include the traditional buffer or heap overflow, string formatting attacks, Unicode misinterpretations (in the case of IIS attacks), and misconfigurations.

- Communication Components: Nodes in the worm network can talk to each other. The communication components give the worms the interface to send messages between nodes or some other central location.

- Command Components: Once compromised, the nodes in the worm network can be issued operation commands using this component. The command element provides the interface to the worm node to issue and act on commands.

- Intelligence Components: To communicate effectively, the worm network needs to know the location of the nodes as well as characteristics about them. The intelligence portion of the worm network provides the information needed to be able to contact other worm nodes, which can be accomplished in a variety of ways.

2.1. Polymorphic Worm Techniques

The attackers will try every possible way to extend the life time of Internet worms. In order to evade the signature-based system, a polymorphic worm appears differently each time it replicates itself. This subsection discusses the polymorphism of Internet worms. There are many ways to make polymorphic worms [10]. One technique relies on self encryption with a vari-able key. It encrypts the body of a worm, which erases both signatures and statistical characteristics of the worm byte string. A copy of the worm, the decryption routine, and the key are sent to a victim machine, where the encrypted text is turned into a regular worm program by the decryption routine. The program is then executed to infect other victims and possibly damage the local system. If the same decryption routine is always used, the byte sequence in the decryption routine can serve as the worm signature.

A more sophisticated method of polymorphism is to change the decryption routine each time a copy of the worm is sent to another victim host. This can be achieved by keeping several decryption routines in a worm. When the worm tries to make a copy, one routine is randomly selected and other routines are encrypted together with the worm body. The number of different decryption routines is limited by the total length of the worm. Given a limited number of decryption routines, it is possible to identify all of them as attack signatures after enough samples of the worm have been obtained. Another polymorphism technique is called garbage-code insertion. It inserts garbage instructions into the copies of a worm. For example, a number of nop (i.e., no operation) instructions can be inserted into different places of the worm body, thus making it more difficult to compare the byte sequences of two instances of the same worm. However, from the statistics point of view, the frequencies of the garbage instructions in a worm can differ greatly from those in normal traffic. If that is the case, anomaly-detection systems can be used to detect the worm. Furthermore, some garbage instructions such as nop can be easily identified and removed. For better obfuscated garbage, techniques of executable analysis can be used to identify and remove those instructions that will never be executed.

The instruction-substitution technique replaces one instruction sequence with a different but equivalent sequence. Unless the substitution is done over the entire code without compromising the code integrity (which is a great challenge by itself), it is likely that shorter signatures can be identified from the stationary portion of the worm. The code-transposition technique changes the order of the instructions with the help of jumps. The excess jump instructions provide a statistical clue, and executable-analysis techniques can help to remove the unnecessary jump instructions. Finally, the register-reassignment technique swaps the usage of the registers, which causes extensive "minor" changes in the code sequence.

3. Related Work

The honeypots are an excellent source of data for intrusion and attack analysis. Levin et al. described how honeypot extracts details of worm exploits that can be analyzed to generate detection signatures [4]. The signatures are generated manually. One of the first systems proposed was Honeycomb developed by Kreibich and Crowcroft. Honeycomb generates signatures from traffic observed at a honeypot via its implementation as a Honeyd [5] plugin. The longest common substring (LCS) algorithm, which looks for the longest shared byte sequences across pairs of connections, is at the heart of Honeycomb. Honeycomb generates signatures consisting of a single, contiguous substring of a worm's payload to match all worm instances. These signatures, however, fail to match all polymorphic worm instances with low false positives and low false negatives.

Kim and Karp [6] described the Autograph system for automated generation of signatures to detect worms. Unlike Honeycomb, Autograph's inputs are packet traces from a DMZ that includes benign traffic. Content blocks that match "enough" suspicious flows are used as input to COPP, an algorithm based on Rabin fingerprints that searches for repeated byte sequences by partitioning the payload into content blocks. Similar to Honeycomb, Autograph generates signatures consisting of a single, contiguous substring of a worm's payload to match all worm instances. These signatures, unfortunately, fail to match all polymorphic worm instances with low false positives and low false negatives.

S. Singh, C. Estan, G. Varghese, and S. Savage [7] described the Early bird system for generating signatures to detect worms. This system measures packet-content prevalence at a single monitoring point such as a network DMZ. By counting the number of distinct sources and destinations associated with strings that repeat often in the payload, Earlybird distinguishes benign repetitions from epidemic content. Early bird, also like Honeycomb and Autograph, generates signatures consisting of a single, contiguous substring of a worm's payload to match all worm instances. These signatures, however, fail to match all polymorphic worm instances with low false positives and low false negatives.

New content-based systems like Polygraph, Hamsa and LISABETH [8, 11 and 12] have been deployed. All these systems, similar to our system, generate automated signatures for polymorphic worms based on the following fact: there are multiple invariant substrings that must often be present in all variants of polymorphic worm payloads even if the payload changes in every infection. All these systems capture the packet payloads from a router, so in the worst case, these systems may find multiple polymorphic worms but each of them exploits a different vulnerability from each other. So, in this case, it may be difficult for the above systems to find invariant contents shared between these polymorphic worms because they exploit different vulnerabilities. The attacker sends one instance of a polymorphic worm to a network, and this worm in every infection automatically attempts to change its payload to generate other instances. So, if we need to capture all polymorphic worm instances, we need to give a polymorphic worm chance to interact with hosts without affecting their performance. So, we propose new detection method "Double-honeynet" to interact with polymorphic worms and collect all their instances. The proposed method makes it possible to capture all worm instances and then forward these instances to the Signature Generator which generates signatures, using a particular algorithm.

An Architecture for Generating Semantics-Aware Signatures by Yegneswaran, J. Giffin, P. Barford, and S. Jha [9] described Nemean, Nemean's incorporates protocol semantics into

the signature generation algorithm. By doing so, it is able to handle a broader class of attacks. The coverage of Nemean is wide which makes us believe that our system is better in dealing with polymorphic worms specially.

An Automated Signature-Based Approach against Poly-morphic Internet Worms by Yong Tang and Shigang Chen[10] described a system to detect new worms and generate signatures automatically. This system imple-mented double-honeypots (inbound honeypot and outbound honeypot) to capture worms' payloads. The inbound honeypot is implemented as a high-interaction honeypot, whereas the outbound honeypot is imple-mented as a low-interaction honeypot. This system has limitation. The outbound honeypot is not able to make outbound connections because it is implemented as low-interaction honeypot which is not able to capture the remaining instances of the polymorphic worm. Our system overcomes this disadvantage by using a double-honeynet system (high-interaction honeypot), which enables us to make unlimited outbound connections between them, so we can capture the remaining instances of the polymorphic worm.

4. Double- Honeynet System

We propose a double-honeynet system to detect new worms automatically. A key contribution of this system is the ability to distinguish worm activities from normal activities without the involvement of experts. Figure 1 shows the main components of the double-honeynet system.

Firstly, the incoming traffic goes through the Gate Translator which samples the unwanted inbound connections and redirects the samples connections to Honeynet1. The gate translator is configured with publicly-accessible addresses, which represent wanted services. Connections made to other addresses are considered unwanted and redirected to Honeynet 1 by the Gate Translator.

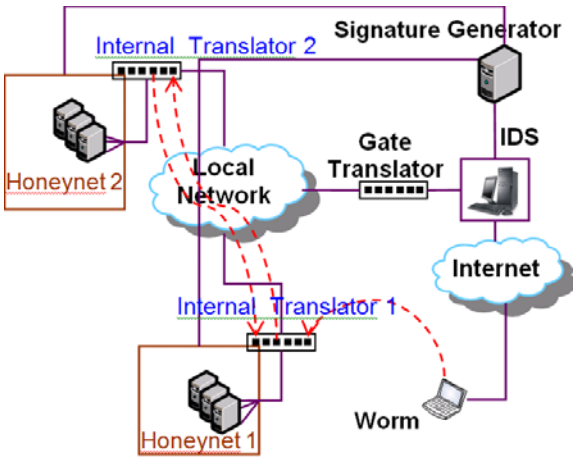

Figure 1.System Architecture

Secondly, once Honeynet 1 is compromised, the worm will attempt to make outbound connections. Each honeynet is associated with an Internal Translator implemented in router that separates the honeynet from the rest of the network. The Internal Translator 1 intercepts all outbound connections from honeynet 1 and redirects them to honeynet 2 which does the same forming a loop.

Only packets that make outbound connections are considered malicious, and hence the Double-honeynet forwards only packets that make outbound connections. This policy is due to the fact that benign users do not try to make outbound connections if they are faced with non-existing addresses.

Lastly, when enough instances of worm payloads are collected by Honeynet 1 and Honeynet 2, they are forwarded to the Signature Generator component which generates signatures automatically using specific algorithms that will be discussed in the next section. Afterwards, the Signature Generator component updates the IDS database automatically by using a module that converts the signatures into Bro or pseudo-Snort format. The above proposed system implemented by using Vmware Server 2. The implementation results are out of the scope of this paper.

For further details on the double-honeynet architecture the reader is advised to refer to our published works [14, 15].

5. Signature Generation Algorithms

In this section, we describe signature generation algorithms processes (Substrings

Exaction and Principal Component Analysis processes). The Substrings Extraction process aims to extract substrings from polymorphic worm whereas the Principal Component Analysis process aims to get the most significant substrings that shared between polymorphic worm instances and to use them as signatures.

Let's assume we have a polymorphic worm A that has n instances (A_1,..., A_n). Assume further that A_i has length M_i for i=1,…,n. Now consider the instance A_1 to be the string (a_1a_2 a_3... a_{M1}). Let X to be the minimum length of the substrings that we are going to extract from A_1. The first substring from A_1 with length X is (a_1a_2... a_X). Then shift one position to the right to extract a new substring, which will be (a_2a_3... a_{X+1}). Continuing this way the last substring from A_1 will be (a_{M_1-X+1} ... a_{M1}). In general if instance A_i has length equal to M and minimum length equal to X, then the Total Substrings Extraction of A_i TSE (A_i) will be obtained by this equation:

$$TSE(A_i) = M-X+1$$

The next step is to increase X by one and start new substrings extraction from the beginning of A_1. The first substring will be (a_1a_2... a_{X+1}). The substrings extraction will continue satisfy this condition X< M.

The procedures mentioned above will be applied for the rest of instances of the polymorphic worm A (i.e. A_2, A_3,.., A_n).

Figures 2 and Table 2 show all substrings extraction possibilities from the string ZYXCBA assuming the minimum length of X is equal to three (3).

Table 2. Substrings Extraction

No. of Subtractions	Length of X	Substrings
S1,1	3	ZYX
S1,2	3	YXC
S1,3	3	XCB
S1,4	3	CBA
S1,5	4	ZYXC
S1,6	4	YXCB
S1,7	4	XCBA
S1,8	5	ZYXCB
S1,9	5	YXCBA

For example, if we have a polymorphic worm with N+1 instances (A0, A2,..., An), we select A0 to be the instance from which we extract substrings. If 9 substrings are extracted from A0 , then assume that each substring will be equal to Si for i= 1 to 9. i.e. A0 =(S1 , S2 ,..., S9). Here we determine the frequency count of each substring Si (A0 substrings) in the each of the rest instances (A1 ,..., An). Then we apply the Principal Component Analysis (PCA) on the frequency count data to reduce the dimension and get the most significant data.

In the next section, we describe the PCA processes in details.

6. Using PCA to determine the most Significant Data

The PCA is often used to reduce the dimension of data for easy exploration and further analysis. It is concerned with explaining the variance-covariance structure of a set of variables through a few new variables which are functions of the original variables [16].

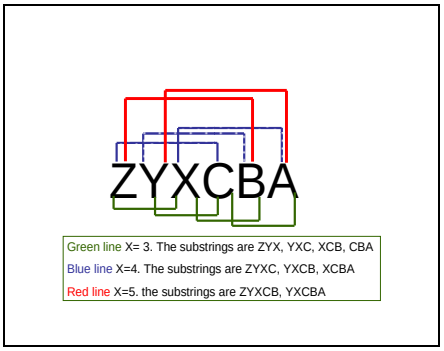

Figure 2. Substrings Extraction

The methodology of employing the PCA to the given problem is outlined below:

Let F_i denotes the vector of frequencies (F_{i1} ,..., F_{iN}) of the substring S_i in the instances (A_1,..., A_n), i= 1,...,L. We construct the frequency matrix F by letting F_i be the i^{th} row of F.

$$F=\begin{pmatrix} f_{11} & \cdots & f_{1N} \\ \vdots & \ddots & \vdots \\ f_{L1} & \cdots & f_{LN} \end{pmatrix}$$

6.1. Normalization of data

The normalization of the data is applied by normalizing the data in each row of the matrix F, yielding a matrix D (L x N).

$$D=\begin{pmatrix} d_{11} & \cdots & d_{1N} \\ \vdots & \ddots & \vdots \\ d_{L1} & \cdots & d_{LN} \end{pmatrix}$$

$$d_{ik} \leftarrow \frac{f_{ik}}{\sum_{j=1}^{N} f_{ij}}$$

6.2 Mean adjusted data

To get the data adjusted around zero mean, we use the formula:

$$g_{ik} \leftarrow d_{ik} - \bar{d}_i \quad \forall i,k$$

Where $\bar{d}_i$ = mean of the i^{th} vector

$$= \frac{1}{N}\sum_{j=1}^{N} d_{ij}$$

The data adjust matrix G is given by:

$$G=\begin{pmatrix} g_{11} & \cdots & g_{1N} \\ \vdots & \ddots & \vdots \\ g_{L1} & \cdots & g_{LN} \end{pmatrix}$$

Evaluation of the covariance matrix: Let g_i denotes the i^{th} row of G, then the covariance between any two vectors g_i and g_j given by:

$$\text{Cov}(g_i, g_j) = C_{ij} = \frac{\sum_{k=1}^{L}(d_{ik} - \bar{d}_i)(d_{jk} - \bar{d}_j)}{N-1}$$

Then the covariance matrix C (N x N) is given by:

$$C=\begin{pmatrix} C_{11} & \cdots & C_{1N} \\ \vdots & \ddots & \vdots \\ f_{N1} & \cdots & f_{NN} \end{pmatrix}$$

Eigenvalue Evaluation: Evaluate the eigenvalues of the matrix C from its characteristic polynomial $|C - \lambda I|$ = 0, and then compute the corresponding eigenvectors.

Principal Component Evaluation: Let λ_{large} large be the largest eigenvalue of the covariance matrix C and V is the Principal Component of the data set.

The projection of data adjust along the Principal Component**:**

Feature Descriptor = V^T x G.

7. Experiment Results

We perform experiments to demonstrate the effectiveness of the proposed signature generation algorithms which are Substrings Exaction and Principal Component Analysis (PCA) in identifying polymorphic worms. 300 instances Blaster worm is used in the experiments. We used Matlab code running on a PC with Intel Pentium 4, 3.19-GHZ CPU and 4.00 GB RAM.

In the experiments, we artificially generate the variants of these worms based on some polymorphism techniques discussed in Section 2. The Figure 3 shows the worm payload before reduction process. The X-axis indicates the number of worm instances that are used to generate signature. The Y-axis indicates frequency count of the extracted substrings. The worm payload dimension might be high before reduction. Hence, the worm signature might be hard to be found in such dimension.

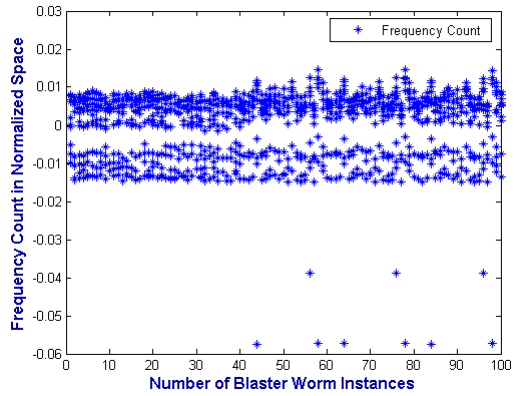

Figure 3. Blaster Worm Substrings before Reduction

The Figure 4 shows the most significant data (worm signature) obtained by the reduction of worm payload using the PCA. It is noticed that the dimension of worm payload is reduced dramatically.

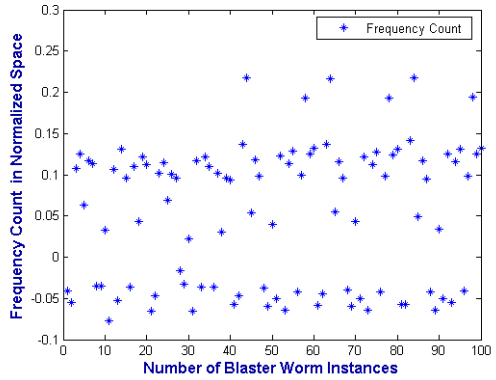

Figure 4. Blaster Worm Substrings after Reduction

The Figure 5 shows the worm detection rate for mixed traffic (worm instances & normal traffic). X-axis indicates the number of mixed traffic, Y-axis indicates the worm detection rate. The detection rate increases as the number of instances increases.

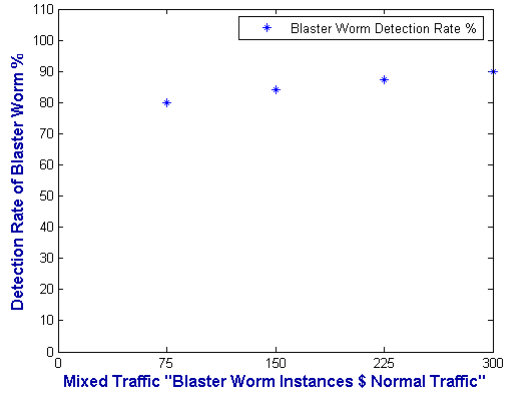

Figure 5. Blaster Worm Detection Rate

Figure 6 shows the False positives and false negatives percentages for the worm. We get zero false positives whereas the false negatives decrease as the number of worm instances increases.

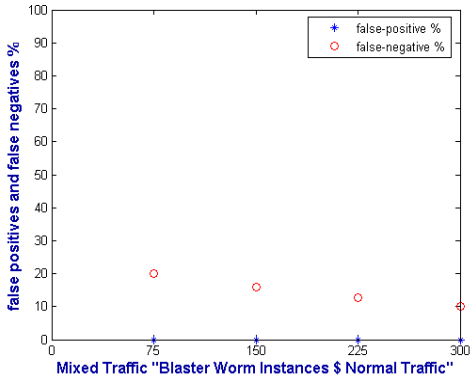

Figure 6. Blaster Worm False Positives and False Negatives

8. Conclusion

We have proposed automated signature generation system for polymorphic worms. We have proposed new detection method “Double-honeynet” to detect new worms that have not been seen before. The system is based on Principal Component Analysis that determines the most significant data that are shared between all polymorphic worms’ instances and use them as signatures. The experimental results show that the PCA has successfully detected polymorphic worms with zero false positives and low false negatives. The main objectives of this research are to reduce false alarm rates and generate high quality signatures for polymorphic worms.

9. References

[1] L. Spitzner, “Honeypots: Tracking Hackers,” Addison Wesley Pearson Education: Boston, 2002.

[2] Hossein Bidgoli, “Handbook of Information Security,” John Wiley & Sons, Inc., Hoboken, New Jersey.

[3] D. Gusfield, “Algorithms on Strings, Trees and Sequences,”, Cambridge University Press: Cambridge, 1997.

[4] J. Levine, R. La Bella, H. Owen, D. Contis ,and B. Culver, "The use of honeynets to detect exploited systems across large enterprise networks," Proc. of 2003 IEEE Workshops on Information Assurance, New York, Jun. 2003, pp. 92- 99.

[5] C. Kreibich and J. Crowcroft, "Honeycomb–creating intrusion detection signatures using honeypots," Workshop on Hot Topics in Networks (Hotnets-II), Cambridge, Massachusetts, Nov. 2003.

[6] H.-A. Kim and B. Karp, "Autograph: Toward automated, distributed worm signature detection," Proc. of 13 USENIX Security Symposium, San Diego, CA, Aug., 2004.

[7] S. Singh, C. Estan, G. Varghese, and S. Savage, "Automated worm fingerprinting," Proc. Of the 6th conference on Symposium on Operating Systems Design and Implementation (OSDI), Dec. 2004.

[8] James Newsome, Brad Karp, and Dawn Song," Polygraph: Automatically generating signatures for polymorphic worms," Proc. of the 2005 IEEE Symposium on Security and Privacy, pp. 226 – 241, May 2005.

[9] V. Yegneswaran, J. Giffin, P. Barford, and S. Jha, "An architecture for generating semantics-aware signatures," Proc. of the 14th conference on USENIX Security Symposium, 2005.

[10] Yong Tang, Shigang Chen," An Automated Signature-Based Approach against Polymorphic Internet Worms," IEEE Transaction on Parallel and Distributed Systems, pp. 879-892 July 2007.

[11] Zhichun Li, Manan Sanghi, Yan Chen, Ming-Yang Kao and Brian Chavez. Hamsa, "Fast Signature Generation for Zero-day Polymorphic Worms with Provable Attack Resilience," Proc. of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2006.

[12] Lorenzo Cavallaro, Andrea Lanzi, Luca Mayer, and Mattia 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, May 2008.

[13] J. Nazario. "Defense and Detection Strategies against Internet Worms ". Artech House Publishers (October 2003).

[14] Mohssen M. Z. E. Mohammed, H. Anthony Chan, Neco Ventura. "Honeycyber: Automated signature generation for zero-day polymorphic worms"; Proc. of the IEEE Military Communications Conference, MILCOM, 2008.

[15] Mohssen M. Z. E. Mohammed, H. Anthony Chan, Neco Ventura, Mohsin Hashim,and Izzeldin Amin, "A modified Knuth-Morris-Pratt Algorithm for Zero-day Polymorphic Worms Detection," Proceedings of The 2009 InternationalConference on Security and Management (SAM'09), Las Vegas, USA, 13-16 July2009.

[16] C. C. Aggarwal and P. S. Yu, " Outliner Detection for High Dimensional Data," Proceedings of the ACM SIGMOD Conference, Santa Barbara, CA, May 21-24, 2001.

[17] Snort – The de facto Standard for Intrusion Detection/Prevention, Available: http://www.snort.org, 14 February 2011.

[18] Bro Intrusion Detection System. Available: http://www.bro-ids.org/, 14 February 2011.

