An Automated Signature Generation Approach for Polymorphic Worms Using Factor Analysis

Mohssen M. Z. E. Mohammed[1], H. Anthony Chan[2], Neco Ventura[2,] Mohsin Hashim[3,] and Izzeldin Amin[3]

[1,2] Department of Electrical Engineering, University of Cape Town, Cape Town, Western Cape , South Africa

[3] Faculty of Mathematical Sciences, University of Khartoum, Khartoum, Khartoum, Sudan

Emails: m_zin44@hotmail.com; h.a.chan@ieee.org; neco@crg.ee.uct.ac.za ;mohsinhashim@yahoo.com; izzeldinamin@yahoo.com

Abstract - *Internet worms pose a major threat to Internet infrastructure security, and their destruction will be truly costly. Therefore, the networks must be protected as much as possible against such attacks. In this paper we propose automatic and accurate system for signature generation for unknown polymorphic worms. We have designed a novel double-honeynet system, which is able to detect new worms that have not been seen before. We apply Factor Analysis to determine the most significant substrings that are shared among polymorphic worm instances and use them as signatures. The system is able to generate accurate signatures for polymorphic worms.*

Keywords: Polymorphic Worms, Honeynet, IDSs.

1 Introduction

The yearly growth of internet worms increasingly threatens the availability and integrity of Internet-based services. Internet worm is a malicious program that spreads automatically among hosts on a network by exploiting various vulnerabilities present on those hosts. A computer worm differs from a computer virus in that a computer worm can run itself. A virus needs a host program to run, and the virus code runs as part of the host program. 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 is the process of monitoring computers or networks for unauthorized entrance, activity, or file modification. IDS can also be used to monitor network traffic, thereby detecting if a system is being targeted by a network attack such as a denial of service attack. There are two basic types of intrusion detection: host-based and network-based. Each has a distinct approach to monitoring and securing data. Host-based IDSs examine data held on individual computers that serve as hosts, while network-based IDSs examine data exchanged between computers. There are two basic techniques used to detect intruders: Anomaly Detection and Misuse Detection (Signature Detection). Anomaly Detection is designed to uncover abnormal patterns of behavior, the IDS establishes a baseline of normal usage patterns, and anything that widely deviates from it gets flagged as a possible intrusion. Misuse detection (signature detection) commonly called Signature Detection, this method uses specifically known patterns of unauthorized behavior to predict and detect subsequent similar attempts. These specific patterns are called signatures [16, 17].

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 discusses the related work regarding automated signature generation systems. Section 3 reviews anatomy of polymorphic worms Section 4 introduces the proposed system architecture to address the problems faced by current automated signature systems. Signature generation m for Polymorphic Worm using Factor Analysis will be discussed in section 5. Section 6 concludes the paper.

2 Related work

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 [3]. 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 Earlybird 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. Earlybird, 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 polymorphic 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 Polymorphic Internet Worms by Yong Tang and Shigang Chen[10] described a system to detect new worms and generate signatures automatically. This system implemented a 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 implemented 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 all polymorphic worm instances. Our system overcomes this disadvantage by using double-honeynet (high-interaction honeypot), which enables us to make unlimited outbound connections between them, so we can capture all polymorphic worm instances.

3 Polymorphic worms

Every worm has a unique bit string which can be used to identify the worm (i.e. all instances of the worm in the network have the same bit string representation). Hence worms can be detected easily using simple signature based techniques (i.e. by comparing the network packets against a database of known signatures). Polymorphic worms, on the other hand, change their representation before spreading i.e. each instance of a polymorphic worm will have a different bit stream representation [4].

3.1 Polymorphic worm techniques

- Encryption

Here, the worm encrypts its body with a random key each time before spreading. A small executable code is then attached to the body of the worm. This executable code is responsible for encrypting the encrypted body of the worm on the victim's machine and then gives control to the worm.

- Code substitution

Here, the instructions in the worm body are substituted with semantically equivalent instructions. Some examples are mentioned below:

1. Multiplication can be achieved by successive addition.

2. Addition can be achieved using xor operator.

3. Register renaming: if you want to transfer a value from register B to A, first move the value to any unused register and then move it to A [4].

3.2 Parts of a polymorphic worm

- Body/Code of the worm

This is the part of the worm which is malicious and does the actual damage.

- Polymorphic Engine (PE)

The polymorphic engine is responsible for changing the representation of the worm either by encryption, code substitution or both.

- Polymorphic Decryptor (PD)

The polymorphic decryptor is responsible for decrypting the worm (if encryption technique is used for polymorphism) on the victim's machine and then give control to the worm [4].

4 Double-honeynet system

4.1 System architecture

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-honeybet system. firstly, the incoming traffic goes through the Gate Translator which samples the unwanted inbound connections and redirects the samples connections to Honeynet 1.

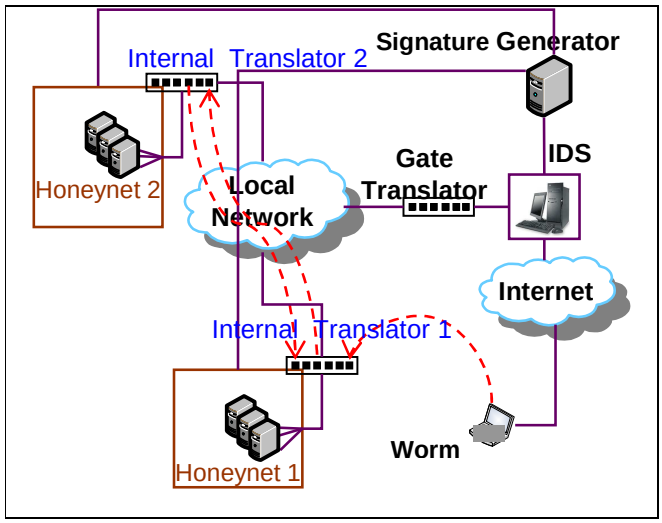

Figure 1.System architecture.

Thee 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.
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 Siganture Generator component which generates signatures automatically using specific algorithms that will be discussed in the next section. Afterwards, the Siganture Generator component updates the IDS database automatically by using a module that converts the signatures into Bro or pseudo-Snort format.
For further details on the double-honeynet architecture the reader is advised to refer to our published works [13,14].

5 Siganture generation algorithms

In this section, we describe signature generation process steps (Substring Exaction Algorithm and Factor Analysis method). The Substrings Extraction algorithm aims to extract substrings from polymorphic worm whereas the Factor Analysis method aims to get the most significant substrings that shared among polymorphic worm instances and to use them as signatures.

5.1 Substrings Extraction

Let's assume we have a polymorphic worm A that has n instances (A_1,..., A_n). Assume further that Ai has length Mi for i=1,…,n. Assume that we select A1 to be the instance from which we extract substrings. Now consider the instance A1 to be the string (a_1 a_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_1 a_2 ... a_x). Then shift one position to the right to extract a new substring, which will be (a_2 a_3... a_{x+1}). Continuing this way the last substring from A_1 will be (a_{m1-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_1 a_2 ... a_{x+1}). The substrings extraction will continue satisfy this condition X< M.

Figures 2 and Table 1 show all substrings extraction possibilities from the string ZYXCBA assuming the minimum length of X is equal to three (3).

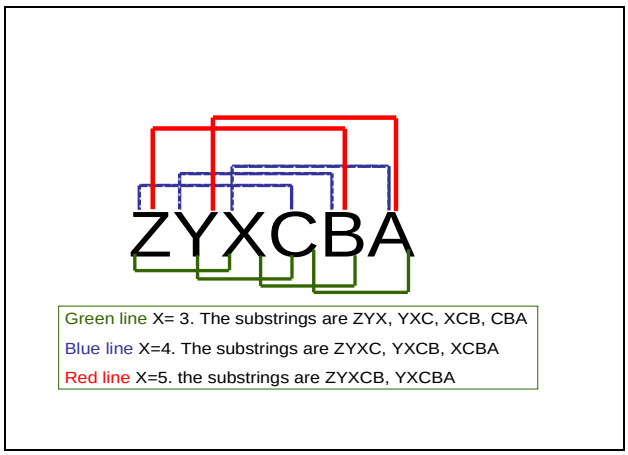

Figure 2. Substrings extraction.

TABLE 1: 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

5.2 Factor Analysis

In this subsection we give a brief introduction to factor analysis and how it is done [15]. Factor Analysis is a multivariate method used for data reduction purposes. The basic idea is to represent a set of variables by a smaller number of variables. In this case they are called factors. These factors can be thought of as underlying constructs that cannot be measured by a single variable. The objective of the use of Factor Analysis in this paper is to reduce the Polymorphic worm payloads dimensions, so the most important factors will appear and use them as signatures.

- **Assumptions**

Factor analysis is designed for interval data, although it can also be used for ordinal data. The variables used in factor analysis should be linearly related to each other. This can be checked by looking at scatterplots of pairs of variables. Obviously the variables must also be at least moderately correlated to each other, otherwise the number of factors will be almost the same as the number of original variables, which means that carrying out a factor analysis would be pointless.

- **The steps in factor analysis**

The factor analysis model can be written algebraically as follows. If you have p variables X_1, X_2, . . . , X_p measured on a sample of n subjects, then variable i can be written as a linear combination of m factors F_1, F_2, . . . , F_m where, as explained above m < p. Thus,

$$X_i = a_{i1} F_1 + a_{i2} F_2 + \ldots + a_{im} F_m + e_i$$

Where the a_is are the factor loadings (or scores) for variable i and e_i is the part of variable X_i that cannot be 'explained' by the factors.

There are three main steps in a factor analysis:

1. **Calculate initial factor loadings**

This can be done in a number of different ways; the two most common methods are described very briefly below:

- *Principal component method*

As the name suggests, this method uses the method used to carry out a principal components analysis. However, the factors obtained will not actually be the principal components (although the loadings for the k^{th} factor will be proportional to the coefficients of the k^{th} principal component).

- *Principal axis factoring*

This is a method which tries to find the lowest number of factors which can account for the variability in the original variables that is associated with these factors (this is in contrast to the principal components method which looks for a set of factors which can account for the total variability in the original variables).

2. **Factor rotation**

Once the initial factor loadings have been calculated, the factors are rotated. This is done to find factors that are easier to interpret. If there are 'clusters' (groups) of variables — i.e. subgroups of variables that are strongly inter-related — then the rotation is done to try to make variables within a subgroup score as highly (positively or negatively) as possible on one particular factor while, at the same time, ensuring that the loadings for these variables on the remaining factors are as low as possible. In other words, the object of the rotation is to try to ensure that all variables have high loadings only on one factor.

3. **Calculation of factor scores**

When calculating the final factor scores (the values of the m factors, F1, F2, . . . , Fm, for each observation), a decision needs to be made as to how many factors to include. This is usually done using one of the following methods:

Choose m such that the factors account for a particular percentage (e.g. 75%) of the total variability in the original variables.

Choose m to be equal to the number of eigenvalues over 1 (if using the correlation matrix). [A different criteria must be used if using the covariance matrix.].

Use the scree plot of the eigenvalues. This will indicate whether there is an obvious cut-off between large and small eigenvalues.

The final factor scores are usually calculated using a regression-based approach.

6 Conclusion

We have proposed automated signature generation for Zero-day polymorphic worms using double-honeynet. We have proposed new detection method "Dou-ble-honeynet" to detect new worms that have not been seen before. The system is based on Factor Analysis that determines the most significant data that are shared among all polymorphic worms instances and use them as signatures. The main objectives of this research are to reduce false alarm rates and generate high quality signatures for polymorphic worms.

7 References

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

[2] D. Gusfield. "Algorithms on Strings, Trees and Sequences". Cambridge University Press: Cambridge, 1997.

[3] 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.

[4] P. Fogla M. Sharif R. Perdisci O. Kolesnikov W. Lee. "Polymorphic Blending Attacks"; Proc. of the 15th conference on USENIX Security Symposium, Vancouver, B.C., Canada, 2006.

[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] 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.

[14] Mohssen M. Z. E. Mohammed and H. Anthony Chan. "Fast Automated Signature Generation for Polymorphic Worms Using Double-Honeynet"; Proc. of the Third International Conference on Broadband Communications, Information Technology & Biomedical Applications, 2008 .

[15] Manly, B.F.J., 'Multivariate Statistical Methods', Third edition 2005, Chapman and Hall.

[16] Snort – The de facto Standard for Intrusion Detection/Prevention. Available: http://www.snort.org, 23 May 2011.

[17] Bro Intrusion Detection System. Available: http://www.bro-ids.org/, 23 May 2011.