An Overview about the Polymorphic Worms Signatures

Raja A. Moftah
Faculty of Information Technology
Benghazi University, Libya
rajaMoftah@hotmail.com

Abdelsalam M. Maatuk
Faculty of Information Technology
Benghazi University, Libya
abdelsalam.maatuk@uob.edu.ly

Peter Plasmann
Faculty of Computing, Engineering and Science
University of South Wales, UK
peter.plassmann@southwales.ac.uk

Shadi Aljawarneh
Software Engineering Department
Jordan University of Science and Technology
Irbid, Jordan

ABSTRACT
With the proliferation of the internet among casual users as well as businesses, the range and frequency of security threats have increased dramatically. One of these threats is a particular type of malware known as a polymorphic worm. This is a program that can mutate its appearance with each infection and spreads through the network via semantics-preserving code manipulation methods or self-encryption techniques. This paper describes the characteristics of polymorphic worms and then explains the most common forms of pattern based detection, such as Autograph.

General Terms
Measurement, Design, Reliability, Security, Verification.

Keywords
Polymorphic Worms, Security, Pattern, IDS, Vulnerability.

1. INTRODUCTION
A worm propagates causing an online security threat by spreading its pattern each time to other machines on the network. By exploiting system vulnerability, each instance of the worm requires an invariant byte string in its payload, which is important as these sequences are distinctive to different types of worm and can be identified by signature [1]. Various security experts have proposed IDS systems, which are used to preserve networks against worms, such as Bro and Snort databases deployed at the edge of the network and the internet [2]. An important function of IDSs is to examine traffic in order to compare it against the signatures accumulated in their databases. When a novel worm is detected, analysis of the worm code is typically done manually by security experts in order to generate a signature. The signature is subsequently distributed, allowing each IDS to update their databases with the new signature. This sometimes occurs days after the worm is released, which is dangerously slow if the worm threat is extremely fast replicating and able to spread through an entire network in seconds [3].

ICEMIS '15, September 24-26, 2015, Istanbul, Turkey

DOI: http://dx.doi.org/10.1145/2832987.2833031

Additionally, the problem arises from the use of these databases is that they are inefficient in detecting more recent types of worms, such as polymorphic worms, which have the ability to alter their appearance and hence have many differences in the signatures for the same worm [4].

The speed of worms generally outbreak in zero day and the polymorphic worm are approximately the same; they can mutate at every copy, in addition to keeping the original algorithm unchanged (invariant bytes). Within each exploit, the worms begin to modify the bytes by deleting the portions of some pieces of code, inserting or modifying some byte sequences thereby avoiding detection through a simple signature matching techniques. However, the parts of the code that remain unchanged can be used to characterise the signature of a polychromic worm [1]. For this purpose, security experts have developed a number of automatic and faster methods to derive more accurate and efficient signatures for worms. Generating a signature automatically can ultimately be read by firewalls or Intrusion Prevention Systems to quickly contain the worm spread. These automatic methods can be used to extract good quality signatures, which preserve all invariant bytes and restriction distances which make identification and preventing worm easier [6].

This paper describes the characteristics of polymorphic worms and then explains the most common forms of pattern based detection, such as Autograph.

2. POLYMORPHIC WORMS
Noh et al. [7] stated that most of the internet worms cause damage to networks through consumption of bandwidth that threatens the security of internet infrastructures and the information about the platform. This threat has become increasingly likely, with the development of advanced worms such as polymorphic worms that can change their program code without human interaction by replicating themselves, enabling them to exploit operating systems and software vulnerabilities in order to contaminate a system [8]. Bayoglu et al. [8] also argued that this type of worm harms the internet by exploiting network infrastructure to transmit copies of itself to other computers. This mechanism of self-propagation helps the worm to spread to many networks very quickly. Each copy keeps the novel algorithm intact during the mutation process, enabling the evasion of detection by a straightforward signature matching technique based IDS. This means that the worm changes its prototype each time, sending this copy to infect other systems, although there are some fractions of its code remaining unchanged. Xiao et al. [9, 10, 11] explained that the propagation of a worm includes three stages:

- Target finding: each copy decides on the next victim by IP address.
- Worm transferring: after finding a target, the worm sends itself to the victim device.
- Infection stage: when the worm's code has transferred to victim machine, the code will be executed.

Kim et al. [5] described the extensive costs of Internet worm epidemics on the network. Through rapid spreading from the victim host to other target hosts and interruption to the computers, network services such as Code Red worm epidemic were estimated to have cost 2.6 billion dollars [5].

2.1 Polymorphism Techniques

According to Tang et al. [4] polymorphism techniques have been exploited to generate worm flows. Polymorphic engines have published shellcode generators with various techniques include ADMmutate, Clet and TAPiON [4]. These techniques have been used to write shellcode of polymorphic that contains Garbage, register shuffling, equivalent code substitution and encoding [encryption/decryption] to evade worm detection. However, the polymorphic worms require invariant bytes and restricted distances to exploit the vulnerability of servers.

The polymorphic mechanism leads to confusion of worm detection approaches by concealing the worm's payload through the use of encoding techniques to write polymorphic shellcodes. If the worm mutates, its payload will generate a different form from its copy, but still have the same function. Polymorphic worms commonly include four parts: Decryption Routine, Decryption Key, Encrypted Worm Code along with Exploit Code [8, 10]. A polymorphic worm exploits an initial vulnerability and then decrypts the encrypted worm code utilizing the decryption routine along with the decrypted key [8]. Various keys of encryption and decryption are applied to encrypt the worm code for each worm sample. Decryption Key along with Encrypted Worm Code models vary for each worm sample, while the Decryption Routine and Exploit code models stay unchanged [8]. Thus, obfuscation mechanisms are used by each worm sample to formulate different Decryption Routine models, which creates an Exploit Code in the unchanged part of the polymorphic worm code that is a source of high false positives if individually utilized to detect the worm. In addition, Bayoglu et al. [8] argued that encryption does not include the full code of the worm as that would make the code inoperative. Each worm therefore has a part of the code that exists for the purposes of exploiting prospective victims. The unencrypted part is used afterwards to branch the implementation cycle to the decryption routine along with the initial code.

2.2 Polymorphic Body

Tang et al. [4] explain that the polymorphic worm sample (infection flow) contains a string sequence. These strings include invariant bytes and wildcard bytes. Invariant bytes contain fixed values and should be present in each worm sample in order to ensure that the infection is successful. Wild card bytes change their values for each diverse worm sample. For instance, a polymorphic Code Red II worm has a sequence of seven invariant contents: "GET",".ida?","XX","%u","%u780","=", along with "HTTP/1.0 r n". So, security professional people attempt to extract the invariant contents of polymorphic worm as a signature. Invariant bytes in a worm flow create a number of invariant contents that are essential to successful worm infection. In other words, the invariant content,"%u 7801" is 4 bytes after "%u", which illustrates the number of characters between two substrings. These distance restrictions are important for the exploitation of a vulnerable server [3]. According to Tang et al. [4], this causes a number of difficulties, as some invariant components cannot be extracted in polymorphic worms. Previous approaches were able to simply generate a single invariant component [5], whereas polygraph [3] is able to extract most invariant components expect for one-byte invariant component such as "=" in the Code Red II worm. Most approaches also do not take into account the all distance restriction in the Code Red II. Despite this, each one-byte invariant components and distance restriction are crucial in worm detection.

2.3 Worm Detection Signature

A worm detection signature is an approach used to find activities of polymorphic worms. The key idea of an extracted signature is to discover match invariant substrings or sequence similarities in all different aspects of a payload. Bayoglu et al. [8] claimed that these methods dealing with polymorphic worms can be further classified into content based detection and behaviour based detection. Content based polymorphic worm detection systems use the worm content to generate information to facilitate matches with the worm. Behaviour based approaches are concerned with the behaviour of the worm in the flow of the network along with system activities. Nevertheless, Xiao et al. [9] stated that worm detection techniques should be classified into two schemas: signature-based and anomaly-based. Currently, automatic signature generation techniques can be associated between two detection schemes.

2.3.1 Signature–Based Worm Detection

Signature–Based Worm Detection is a typical approach used in IDSs, which works by representing the behaviour and prototypes of the worms and examining for a match. If a match is detected, the detection of the prevention systems IDSs will be raised. This model includes the Network signature type that contains regular expressions, which is intended to match each infection within the network stream of a worm. There are also other types that aim to track a worm in a file system such as File signatures, or to expose the behaviour of the worm in the target host, such as Log signature applications [9]. This study will focus on Network signature types, divided into exploit signatures and vulnerability signatures. Most IDSs and anti-virus vendors provide two types of signatures and offer information on both exploits and vulnerabilities used for worms [9].

2.3.2 Anomaly-Based Worm Detection:

Anomaly-Based Worm Detection schemes construct models of program behaviour or normal networks, raising an alarm when the program or behaviour of a host departs from these models [9].Various means can be used to perform this, by checking the payload of each packet to ensure that each packet satisfies the normal model to detect the payload from polymorphic worms, based on real and dynamic execution of network data [9].

2.3.3 Automatic Signature Generation

Signature-based detection systems are precise, efficient and simple to progress and deploy [9]. On the other hand, they are not able to discover unknown worms unless novel signatures are available. Anomaly-based detection systems are capable of detecting unknown worms. However, they suffer from high levels of false alarms (false positives) while modelling normal behaviour

is extremely complex. Automatic signature generations have been developed to associate between advantages of signature-based detections and anomaly-based detections. Computer security experts can use anomaly-based detection to find unknown worms, along with automatic signature generation systems to produce accurate signatures for detection. Worm containment will then reacts quickly after the worm detection and reduce the harm which can be caused. Currently, automatic signature generation has become an important issue and numbers of techniques have been projected. These systems are classified into two subtypes Host-based and Network-based [9].

2.3.3.1 Host-based Signature Generation (HSG)

HSG systems run to defend the hosts, utilising the host information to discover the attempts of the infection and extract signatures from these attempts. Usually, HSG systems generate precise signatures rapidly, but they also have several negative effects on the defending host recital and configuration such as requirements that the host recompiles the kernel [9].

2.3.3.2 Network-based Signature Generation (NSG)

NSG systems only analyze the suspicious network flow along with output content-based signatures [9]. Compared with HSG systems, NSG systems are more sensitive at early stages of worm propagation, because they able to capture worm samples earlier from the network router and gateway level. Early networks-based signature generation systems contain Autograph, which was capable of generating solely single-string signatures. Polygraph and Hamsa generate token-based approaches, in which a token is a byte sequence that occurs in a significant number of suspicious traffics. These tokens have a high exposure of suspicious flows and low false positive reactions to the normal flow pool. Recently, Simplified Regular Expression was developed to apply multiple sequence alignment to generate signatures [9].

2.3.4 Exploit-Bashed Signature Generation Schemes

As mentioned above, there are various approaches for automatic signature generation that is able to output exploit-based signatures: Network-based and Host-based [4]. These types can be further classified with Exploit based signatures into:

2.3.5 Network-based and Exploit-based Schemes

This has exploited by early network-based signature generation approaches, containing Honeycomb, Autograph, PAYL and Earlybird which only deal with a single infection. Meanwhile, Polygraph and Hamsa approaches choose a set of token that has a high coverage of suspicious flow and provides a high level of false positive results. Recently, generation SER signature has been shown to be more accurate and does not depend on well classified worm flow pools [4].

2.3.6 Host-based and Exploit-based Schemes

There are a number of approaches including TaintCheck that apply dynamic taint examination to detect anomaly instruction implementation and output three-byte signatures, which are used to overwrite a jump target. DACOD assumes a similar technique and outputs a set of token as signatures to detect intrusions. DIRA is a compiler that is able to transform random programs so it can detect control hijacking attacks in which malicious packets are recognized as the signature [4]. However, this study does not require the identification of several unrevealed information sources, such as TaintCheck and DACOD, which require distinguishing the vulnerable programs or having the source code as DIRA [4]. They cannot automatically create the signature of a worm also require knowledge of certain hidden information because all methods are based on Host-based schemes that seek to collect more information at a host as binary/sources code of vulnerable programs [12]. This study seeks to discover the approaches such as autograph and aims to generate more accurate signatures to identify and filter out polymorphic worms.

3. PATTERN-BASED DECTION

3.1 Autograph

Autograph is one of an early pattern which was constructed to automatically generate signatures for novel Internet worms. This approach is used to generate signatures that demonstrate high true positive rate (high sensitivity) and low false positive rate (high specificity), using content based filtering. Kim et al. [5] restrict their analysis to worms that propagate over TCP transport. The signature is a tuple, including: IP protocol number, destination port number, along with byte sequence. The content is based on filtering and considers the payload in the network stream, when it matches the byte sequences in the signature by utilising the same IP protocol destined for the destination port number, and is then classified as a worm.

3.2 Architecture of Autograph

In general, for an overview of this system, all traffic crossing the Demilitarized Zone (DMZ) inputs to an Autograph monitor and outputs a list of the worm signatures. This system comprises of two phases: suspicious flow selection and signature generation. The suspicious flow selection is used to classify the network stream as a suspicious flow pool (malicious) along with non-suspicious flow pool (innocuous). The signature generation is found in the veracity of how worms propagate to exploit the software vulnerability and to execute; thus all instances consist of one or an additional common byte sequence. The worm spreads in a bulky number, so the amount of common content block is high. The suspicious flow is used to generate a signature by dividing it into smaller content blocks, and the number of suspicious flows in which every block's content arises is counted. This content block is ranked according as its prevalence, with a higher prevalence achieving a higher count. The commonly occurring content block is utilised as the signature [5].

The authors explain that this system relies on heuristics to investigate network traffic and to derive a packet classifier which is usually used as a port scanner discovery mechanism to classify incoming traffic as either malicious or innocuous [5]. Scanning discovery is executed through monitoring inbound unsuccessful TCP connections. Every external host which has accomplished an unsuccessful connection and tries additional than x internal IP addresses is measured to be a scanner. Therefore, the autograph generates signatures for worms that transmit by randomly scanning IP addresses only. The discovered mechanism is inapplicable if port scanners utilise spoof platform address. In order to deal with this issue, the authors plan in the future to apply various anomaly detection mechanisms [5].

Autograph carries out TCP flow reassembly for inbound payloads which are malicious. In addition, all malicious flow is accumulated with regard to its destination port. The process of signature generation is initiated when the malicious flow pool consists of more than a threshold number of flows for a particular destination port. While for the signature generation, autograph computes the frequency with which non-overlapping payload

substrings occur across all malicious payloads, along with noting the frequently occurring substring for the candidate's signature. To achieve this, each flow's payload is separated into variable-length content blocks by COntent-based Payload Partitioning (COPP), along with the number of malicious payloads, as every content block that occurs is counted. Kim et al. [5] indicate this count as the content's prevalence. Also, COPP is initiated in the file scheme domain, and calculates a series of Rabin fingerprints and a sliding window of the flow's payload to make the content blocks. The content blocks emerging only in flows originating from single platform IP addresses are removed. Moreover, Kim et al. [5] identifies that these content blocks sometimes occurred due to misconfigured systems which are not malicious. Also, the byte strings that are identified as achieving a high false positive could be blacklisted through a local administrator. The content blocks remaining are utilised for generating a signature. Autographing repetitively the process for the majority of prevalent content block is chosen as signature and then all flows found in this content block are eliminated. This process is repetitive, with remaining flows continuing until the fraction of the entire flows in the pool has been enclosed. Autograph will report the set of chosen signatures in Bro's signature feature at the end of the process.

3.3 Evaluation of Autograph System

In a general evaluation, the quality of the signature generated by this system depends upon the size of the content block. When the size of the content block has been made small, it is in most cases autograph signature generating. Then, the suspicious flow after that passes through the generator signatures, which automatically generates various classes of signatures. The authors provide more detail about the evolution of the generation of signatures in their paper through Autograph, and they also describe their prototype implementation. However, the client can easily guess from the performed experiments that a prototype exists. Initially, they have explored the effect of the content size on the quality of signature generation. In their experiment, Autograph is supplied with packets traces from DMZ from two different research labs, each of the research's 29 IP address include the complete packet payload. For computing Autograph's true positive rate, the identical trace was experimented with initially by applying Bro with well-known signatures- the scanning-based HTTP worms Code-Red, Code-RedII, Nimda, and this was followed by applying Autograph's signatures.

For computing the rate of the false positive, a sanitised trace was performed by taking out the entire flows from the traces that were formerly known through Bro as worms. Subsequently, Bro and Autograph's signatures were run in the sanitised trace. This testing shows that the presentation of generated signatures differs for varies parameters, which is the fraction flows detected through the generated signatures, along with the content block size. The author states that the optimal parameters established may not be related to other traces. Also, they assert that the Autograph system is able to generate extremely short signatures or worms, along with imperfect polymorphism (set 56-byte sequence). Even so, they point out that the short signature may cause high false positive rates. Autograph's distributed signature discovery system is estimated to determine how quickly Autograph discovers and generates a signature for a newly released worm.

The results show that there exists a range of processes whereby the system produces signatures which have not sourced any false positive, in an appropriate fashion. Finally, to accelerate the accumulation of the Internet worm payloads, Autograph is developed along with a technique to share suspicious resource addresses among whole monitors. This technique is a so-called tattler protocol, and is in addition to RTPC protocol which is utilised for controlling multimedia conferencing sessions; furthermore, it has been revealed to scale to thousands of senders. The tattler protocol is applied to allow the monitors to announce the IP address and destination port of scans received, and every announcement includes around one to 100 port scanner reports. Simulations demonstrate that the peak bandwidth consumed by tattler through a Code-RedI epidemic is just 15Kbps, but that does not include the background port scanning [5].

4. CONCLUSONS

This paper provides a survey on the polymeric worms and the characteristics of polymorphic worms and then the most common forms of pattern based detection have been explained, such as Autograph. As a part of future work, we will develop a framework architecture to describe the polymorphic signature.

5. REFERENCES

[1] Saudi, MM. Tamil, EM. Nor, SAM. Idris, MYI. Seman, K.. 2008. EDOWA Worm Classification. In *Proceedings of the World Congress on Engineering 2008, vol. 1.*

[2] Unterleitner M. 2012. *Computer Immune System for Intrusion and Virus Detection: Adaptive Detection Mechanisms and their Implementation*. AV Akademikerverlag.

[3] Newsome, J. Song, D. 2005. Dynamic taint analysis for automatic detection, analysis, and signature generation of exploits on commodity software. In *the 12th annual network and distributed system security symposium*.

[4] Tang, Y., Xiao, B., Lu, X. 2009. Using a bioinformatics approach to generate accurate exploit-based signatures for polymorphic worms. *Comput. Secur*, vol. 28(8), pp. 827-842. http://dx.doi.org/10.1016/j.cose.2009.06.003

[5] Kim, H.A. 2010. *Privacy-Preserving Distributed, Automated Signature-Based Detection of New Internet Worms*. Ph.D. Thesis, Carnegie Mellon University, Pittsburgh.

[6] Wang, J., Hamadeh, I., Kesidis, G. And Miller, D. 2006. Polymorphic worm detection and defense: system design, experimental methodology, and data resources. In *Proceedings of the 2006 SIGCOMM*. ACM, New York, NY, USA, pp. 169-176. DOI=http://doi.acm.org/10.1145/1162666.1162676

[7] Noh, H., Kim, J., Yeun, C.Y. and Kim, K. 2008. New Polymorphic Worm Detection based on Instruction Distribution and Signature. In *the 2008 Symposium on Cryptography and Information Security*. Miyazaki, Japan.

[8] Bayoglu, B. and Sogukpinar, I. 2008. Polymorphic Worm Detection Using Token–Pair Signatures. In *Proceedings of the 4th int. workshop (SecPerU '08)*, ACM, NY, USA, pp. 7-12. DOI=http://doi.acm.org/10.1145/1387329.1387331

[9] Xiao, B., Tang, Y., Luo, J. And Wei, G. 2009. Concept, characteristics and defending mechanism of worms. In *IEICE Transactions on Info. and Syst*. vol. E92-D(5), pp. 799–809.

[10] Aljawarneh, S. 2011. A web engineering security methodology for e-learning systems. In *Network Security*, *Elsevier*. vol. 2011(3), pp. 12-15. doi:10.1016/S1353-4858(11)70026-5

[11] Aljawarneh, S. 2011. Cloud Security Engineering: Avoiding Security Threats the Right Way. In *IJCAC*, vol. 1(2), pp. 64-70. doi:10.4018/ijcac.2011040105.

[12] Li, Z., Sanghi, M., Chen, Y., Kao, MY. And Chavez, B. 2006. Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. In *Security and Privacy, 2006 IEEE Symposium on.*