An Efficient Detection and Prevention Approach of Unknown Malicious Attack: A Novel Honeypot Approach

Aatif Sarfaraz, Atul Jha, Avijit Mondal, and Radha Tamal Goswami

Abstract In this modern era, security has gotten to be the foremost broadly concerned in each domain as recently approaching malware postures a danger to the systems. So our fundamental concern is to identify and anticipate a malware assault on the system. As the polymorphic worm postures, an enormous challenge to identify as they have more than one occurrence and exceptionally expansive endeavors is required to detain every occurrence and to generate signatures. This work proposes malicious attack detection and prevention using honeypot technology. We have proposed a double-honeynet framework, which can distinguish and avoid modern worms. We apply system call analysis to detect malware which mostly focuses on the polymorphic and metamorphic virus rather than utilizing a signature-based approach.

Keywords Polymorphic worm · Honeypot · Double honeypot · Sticky honeypot · System call analysis

1 Introduction

Nowadays, malicious attacks became the biggest threat to network security. Malicious attacks include viruses, worms and trojans. In each of these attacks, some codes are written called malicious code or malware. Malware could be a sort of computer program outlined to require over or harm a computer's working framework without the client's information or endorsement. Worms are being the major

A. Sarfaraz (✉) · A. Jha
Department of Computer Science, Techno International Batanagar, Kolkata, India

A. Mondal
Makaut, Kolkata, India
e-mail: avijit.mondal@tib.edu.in

R. T. Goswami
Techno International New Town, Kolkata, West Bengal, India
e-mail: rtgoswami@tict.edu.in

K. Khanna et al. (eds.), *Cyber Security and Digital Forensics*,
Lecture Notes on Data Engineering and Communications Technologies 73,
https://doi.org/10.1007/978-981-16-3961-6_2

threat among all malicious attacks, because of their self-replicating nature [1]. It is capable of copying itself and sending it over the network to the computers on the network without any human interaction. Unlike viruses, worms did not need to attach themselves to any program in order to propagate. As there is a particular behavior, blocking software program is there which coordinates with the working framework and monitors the program behavior in genuine time for malicious activities. With the evolution of network security tools and techniques, malware is getting to be more intelligent each day, and polymorphic malware is the most recent participants in this calamitous amusement of overcoming the adversary. Polymorphic worm being one of the latest and effective among all [2]. A polymorphic worm changes its appearance with every occurrence [3] such that each occurrence of which receives a totally distinctive personality from its parent, which makes it difficult to gets detected.

Network intrusion detection system (NIDS) must be established in order to protect the network from malware, and it acts as an alarm that notifies the network admin every time malware is detected. IDSs basically are of two types: host-based and network-based. Host-based IDSs inspect data held on discrete computers in the network which serves as a host, whereas network-based IDSs inspect data interchanged between computers.

Security specialists physically create the IDS signature by studying the network traces after the modern worm has been discharged. Tragically, this work takes a parcel of time. Analysts have, as of late, given consideration to computerizing the era of marks for IDS to coordinate worm activity.

Our research is based on the honeypot technique which provides a solution for the above-mentioned problem. A honeypot may be a trap that mimics an authentic network resources, a self-contained secure and monitored area [4, 5]. Its essential objective is to bait and detect malicious attacks and intrusions. It can be used for surveillance and early warning; it can also benefit security researchers to understand emerging threats. Honeypots can be categorized as high and low interaction honeypot [6]. A high interaction honeypot like honeynet works as a real operating system, whereas a low interaction honeypot like honeyd imitates one or more than one real system.

Our research proposes a double-honeynet architecture which gathers all possible instances of the polymorphic worm and sends them for signature generation. In our research, we have used system call analysis approach instead of signature-based approach for the detection of the polymorphic worm inside the signature generator. Our system makes it possible to gather all the instances of the polymorphic worm and then forward those instances to the signature generator which then generates the signature.

This paper is divided into the following segments. Segments 1.1 and 1.2 give a brief idea about worms and polymorphic worms. Segment 2 reviews the associated work for the automatic signature generation framework. Segment 3 presents our proposed framework to describe the problem encountered by the current automatic signature generator. The signature generator algorithm for polymorphic and metamorphic virus using system call analysis will be discussed in Segment 4. The

algorithm for our proposed architecture is discussed in Segment 5. Segment 6 concludes the paper, and in Segment 7 we will discuss some future work that we intend to perform in the future.

1.1 Worms

"A worm is a self-reproducing program that can be created with the capabilities to perform any kind of task," for example, deletion of files or sends documents via e-mail. There can be an adverse effect on network traffic due to the worm self-replicating property:

A worm:

(a) may install a pathway also called a backdoor for unprotected access by the attacker in the infected system.
(b) vulnerability introduces the worm into the system.
(c) may infect one system and by the property of self-replication, it can spread all over the system network.

1.2 Polymorphic Viruses

Polymorphic infections assume numerous forms by scrambling code in an unexpected way with every infection. This term caught in within the mid-1990s with devices that risen to produce thousands of modern slight variations of code based on transformed routines to subvert-signature innovation utilized by an anti-virus computer program.

A polymorphic infection incorporates an encrypted infection body and a decryption routine, to begin with controlling the computer and later decodes the infection body [7].

However, a polymorphic infection moreover includes a mutation engine that produces randomized decryption routines that alter each time an infection infects an unused program. The mutation engine and virus body were both encrypted in a polymorphic infection.

As soon as the users run an infected program, following activities will take place:

- The framework is beneath the control of the decryption routine that can decode both the infection body and the mutation engine [8].
- Next, the control has been shifted from the system to the virus through the decryption routine which finds a new program to infect.
- Next the infection replicates itself along with the mutation engine in RAM.

- At this point, mutation engine gets invoked by the virus which then arbitrarily produces a new decryption routine that can decrypt the virus. However, it marks a very little or no likeness to any earlier decryption routine
- At this point, the current copy of the virus body and the mutation engine gets encrypted by the virus [9, 10].
- Ultimately, the virus adds the current encryption routine, together with the recently encrypted virus body and mutation engine, onto the latest program.

As a consequence of the above-mentioned process, the virus body is encrypted as well as virus decryption routine differs from malware to malware. With no predetermined signature to scan for and no predetermined decryption routine to malware, it cannot look the same.

2 Literature Review

Levin et al. [9] in his paper outlined the technique of how the honeypot pulls out the characteristics of worm exploits, which can be examined for signature generation. The drawback of his paper is that the signature for the worm exploit is detected manually, and this process of manual signature generation for the worm takes a lot of time which gives worms more time to propagate and infect other hosts in the network. Our paper uses double-honeynet architecture which overcomes the above-mentioned problem by consequently producing signatures for worms. Our framework reduces the time and effort required for signature generation.

Honeycomb was one of the first proposed frameworks for signature generation designed by Kreibich and Crowcroft [11]. Honeycomb can be implemented as a honeyd plugin, and it produces signatures from the detected activities at honeypot. The honeycomb system is based on the LCS algorithm, mainly focuses on the "longest-shared-byte" sequence between different sets of packets. To match almost all kinds of worm instances, a honeycomb generates a signature that comprises a single or contiguous substring of worm payload. In this paper, the major drawback is the signature which is generated by the honeycomb fails to capture all the instances of the polymorphic worm with low false negative and low false positive. Our paper uses double-honeynet architecture which is based on system call analysis that overcomes the above-mentioned drawback by capturing almost all instances of polymorphic worms. The system call analysis is used in our framework over the signature-based approach as it focuses mostly on polymorphic and metamorphic worms which help in capturing almost all kinds of polymorphic worm instances.

Goswami et al. [12] in their paper described how to automatically generate the signature for a polymorphic worm using double-honeynet framework which is built on the PCA, which uses the most noteworthy information shared between each and every polymorphic worm instances as the signature. The drawback of this framework is that it may not distinguish all the occasions of the polymorphic worm.

Singh et al. [13] in their paper outlined a system called the Earlybird system for the signature generation to identify worm. This architecture calculates packet content prevalence at a single observation point like a network demilitarized zone. The Earlybird system differentiates benign repetitions from outbound content, by counting the number of definite starting and endpoints related to the string that repeats frequently in the payload. Similar to honeycomb, Earlybird also generates a signature which comprises a single-contiguous substring of a payload to coordinate most of the worm occasions. In this paper, the major drawback is the signature which is generated by the Earlybird system fails to catch all the occurrences of the worm with low false negative and low false positive. This paper uses a system call analysis-based double honeynet framework which focuses more on polymorphic and metamorphic worms that overcomes the above-mentioned paper's drawbacks.

3 System Architecture

A polymorphic worm could be a sort of worm which can alter its appearance each time it propagates, so a polymorphic worm has numerous instances. After infecting every host, a new occurrence of polymorphic worm is produced for propagation. A single honeynet is capable of detecting only one instance of it. Therefore, we need a double-honeynet framework which will catch all the conceivable polymorphic worm instances. The proposed double-honeynet model establishes a loop that accumulates all polymorphic worm occurrences as shown in Fig. 1. For all of the above-mentioned reasons, we propose a double honeynet architecture for the detection of zero-day polymorphic worm automatically.

The process begins when the incoming traffic approaches the gate translator and passes through it. The gate translator then separates the suspicious incoming traffic and swerve them to honeynet 1. "The gate translator consists of publically attainable addresses, which represent wanted amenities." Traffics linked to any additional addresses, other than the publicly attainable addresses, were considered suspicious and are swerve to the honeynet 1 by gate translator.

In the next step, the unwanted suspicious traffic which arrived at honeynet 1 will strive to make an outbound connection. An internal translator that is implemented in the router and associated with each honeypot separates the honeypot from the entire network. All the outbound connections that are made from honeynet 1 will be obstructed by the internal translator and are swerve to honeynet 2. Honeynet 2 does the same forming a loop.

When adequate occurrences of worm payloads are accumulated by honeynet 1 and honeynet 2, then they are passed through a sticky honeypot which moderates down the worm proliferation rate so that it will be easy for the signature generator to generate signatures. Afterward, the signature generator generates signatures automatically by system call static analysis that will be discussed in the next section. All the malicious traffic will go through the system call analysis, and a corresponding signature will be generated. The generated signatures are forwarded

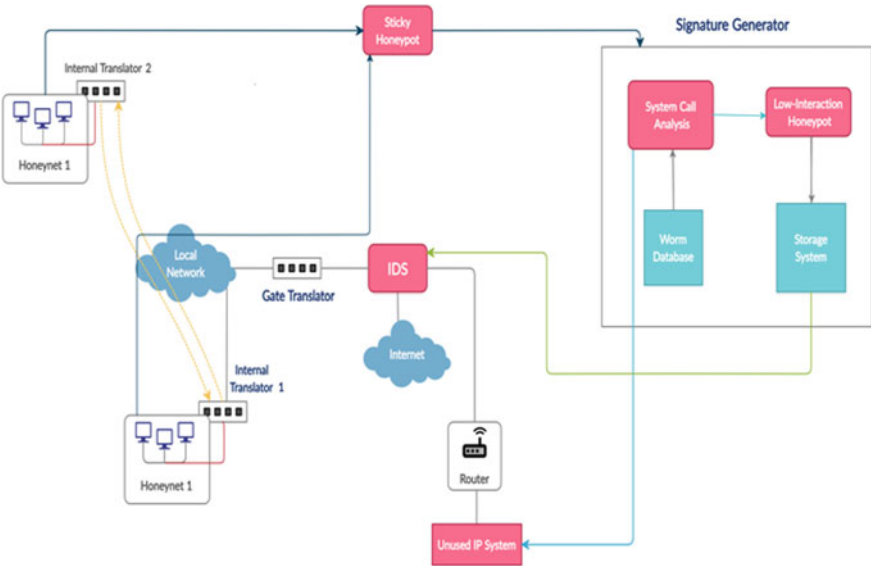

Fig. 1 System architecture

to the centralized storage system via a low interaction honeypot. Then the IDS can get all the information so that by analyzing it can protect against any attack. If the system call analysis is incapable of detecting any signature, then the payload will be automatically transferred to the unused IP system.

4 System Call Analysis

It is utilized to distinguish malware, generally focuses on the polymorphic and metamorphic infections. This strategy works based on the presumption that all malware variations share a common core signature—a combination of many highlights of the programming code [14]. In this technique, two fundamental steps were included: To start with, the portable executable (PE) decompressed and passed through a parser, this parser makes a list of Windows APIs calling grouping. Minute, this API grouping will be compared against the worm database, and a similarity measure was utilized to conclude the analyzed file [15]. In the event, if the similarity is more noteworthy than a certain limit, then at that point detection is triggered.

5 Proposed Algorithm

Step 1: The gate translator swerves the traffic coming from the Internet toward honeynet 1.
Step 2: Internal translator that is implemented in the router disconnects honeynet 1 from the entire network.
Step 3: All the outbound connection from honeynet 1 was obstructed by the internal translator and swerves them to honeynet 2. Honeynet 2 does the same forming a loop.
Step 4: When an adequate amount of worm instances are acquired by looping through honeynet 1 and honeynet 2, then they are swerved to the sticky honeypot to slow down the worm propagation rate.
Step 5: Sticky honeypot transfers the worm instances to the signature generator to generate their signatures.
Step 6: Signature generator consists of system call analysis, worm database, low interaction honeypot, and centralized storage system.
Step 7: In system call analysis, portable executable file is decompressed and passed through a parser which creates a list of windows API calling sequences.
Step 8: These API sequences will be compared against the worm database, if the similarity is greater than a certain threshold, then detection is triggered.
Step 9: All these worm signatures which are detected by system call analysis will be transferred to a centralized storage system through low interaction honeypot by which IDS can get all the payload information.
Step 10: In case, the system call analysis is incapable of detecting any signature, then that payload will be automatically redirected to the Internet through an unused IP system.

6 Conclusion

We have proposed an algorithm to generate the signature of newly emerged polymorphic worms automatically. In this paper, we have proposed a new detection technique “double honeynet” for the detection of newly emerged polymorphic or metamorphic worms. The Framework also includes a sticky honeypot which slows down the worm propagation rate. The system is based on the system call analysis that compares windows API sequences with the worm database; if the similarities are greater than a certain threshold, the detection is triggered.

7 Future Work

As honeypots are moderately a modern innovation and have a great scope for future applications, we plan to execute this proposed framework on an experimental test bed.

In the future, we intend to propose a new architecture which will be based on sandboxing. Using a sandboxing for modern malware discovery gives another layer of assurance against modern security threats—zero-day polymorphic worm. And whatever happens within the sandbox remains within the sandbox—avoiding system failures and keeping computer program vulnerabilities from spreading.

"A sandbox in a cybersecurity domain is a secluded environment on a network that imitates an end user working environment. Sandboxes are used to execute malicious code to analyze the results without having any adverse effect on the rest of the network. Sandbox environment gives a proactive layer for network security defense against threats."

The key objective of our research is to lower the false alarm rates and to produce high-quality signatures for the polymorphic worm.

References

1. Spitzner, L.: Honeypots: Tracking Hackers. Addison Wesley Pearson Education, Boston (2002)
2. Tang, Y., Chen, S.: An automated signature-based approach against polymorphic ınternet worms. IEEE Trans Parallel Distrib Syst **18**(7), 879–892 (2007)
3. Cavallaro, L., Lanzi, A., Mayer, L., Monga, M.: LISABETH: automated content-based signature generator for zero-day polymorphic worms. In: Proceedings of the Fourth İnternational Workshop on Software Engineering for Secure Systems, Leipzig, Germany, May 2008, Lissa (2008)
4. Mohhammed, M.M.Z.E., Anthony Chan, H., Ventura, N.: Honeycyber: automated signature generation for zero-day polymorphic worms. In: Proceedings of the IEEE Military Communications Conference, MILCOM (2008)
5. Fogla, P., Sharif, M,. Perdisci, R., Kolesnikov, O., Lee, W.: Pol polymorphic blending attacks. In: Proceedings of the 15th Conference on USENIX Security Symposium, Vancouver, B.C., Canada (2006)
6. Li, Z., Sanghi, M., Chen, Y., Kao, M.Y., Chavez, B.: Hamsa: fast signature generation for zero-day polymorphic worms with provable attack resilience. In: Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May (2006)
7. Kim, H.-A., Karp, B.: Autograph: toward automated, distributed worm signature detection. In: Proceedings of 13 USENIX Security Symposium, San Diego, CA, Aug. (2004)
8. Kreibich, C., Crowcroft, J.: Honeycomb–creating intrusion detection signatures using honeypots. In: Workshop on Hot Topics in Networks (Hotnets-II), Cambridge, Massachusetts, Nov. 2003 (2003)
9. Levine, J., La Bella, R., Owen, H., Contis, D., Culver,B.: The use of honeynets to detect exploited systems across large enterprise networks. In: Proceedings of IEEE Workshops on Information Assurance, New York, June 2003, pp. 92–99 (2003)
10. Gusfield, D.: Algorithms on Strings, Trees and Sequences. Cambridge University Press, Cambridge (1997)

11. Krebitch, C., Crowcroft, J.: Honeycomb: creating intrusion detection signatures using Honeypots. ACM SIGCOMM Comput Commun Rev **34**(1), 51–56 (2004)
12. Goswami, R. T., Mondal, A., Mishra, B. K., Mahanti, N. C.: Handbook of Information Security. John Wiley & Sons, Inc., Hoboken, New Jersey
13. Singh, S., Estan, C., Varghese, G., Savage, S.: https://www.lastwatchdog.com/internets-40th-anniversary-timeline-milestones/
14. https://epdf.pub/official-isc2-guide-to-the-sscp-cbk-second-edition.html
15. Salah, E.D., Aslan, H.K., El-Hadidi, M.T.: A detection scheme for the SK Virus. In: Proceedings International Federation for Information Processing (2002)