Detecting Zero-day Polymorphic Worms Using Honeywall

Mohssen Mohammed[1], Mohamed Abdalla Nour[1], Mohamed Elhoseny[1,*]

[1]College of Computing and Informatics, University of Sharjah, UAE
Emails: m_zin44@hotmail.com; mnour@sharjah.ac.ae; melhoseny@sharjah.ac.ae

Abstract

A polymorphic worm is a kind of worm that can change its payload in every infection attempt, so it can evade the Intrusion Detection Systems (IDSs) and perform illegal activities that lead to high losses. These worms can mutate as they spread across the network, causing most of the existing IDSs to carry out the polymorphic worm's detection with high levels of both false positives and false negatives. In this paper, we propose a double-honeynet system that can detect polymorphic worm instances automatically. The Double-honeynet system is a hybrid system with both Network-based and Host-based mechanisms. This allows us to collect polymorphic worm instances at the network-level and host-level, which reduces the false positives and false negatives dramatically. The experimental deployment of a Double-honeynet network over a seven-day period successfully collected instances of various polymorphic worms, including 3511 Allaple, 3228 Conficker, 2817 Blaster, and 2452 Sasser worms. By utilizing, the Honeywall's Walleye interface; we were able to analyze the data and simulate the detection of these worms by generating new signatures, which were not previously recorded, demonstrating the system's capability to detect zero-day polymorphic threats. Analysis of Blaster worm instances revealed significant similarities in their payloads due to exe headers, indicating the necessity of preprocessing to remove these headers before signature generation, although the generation of signatures is beyond the scope of this study.

Keywords: Polymorphic Worms; Zero-day Threats; Cybersecurity; Network Security; Intrusion Detection Systems (IDS); Honeynet; Honeywall; Malware Detection; Walleye Interface; Worm Signature Generation

1. Introduction

A computer worm is a kind of malicious program that self-replicates automatically within a computer network. Worms are serious threats to computers connected to the Internet and their proper functioning. These malicious programs can spread by exploiting low-level software defects [1]. They can use their victims for illegitimate activities. Such activities include corrupting data, sending unsolicited electronic mail messages, generating traffic for distributed Denial of Service (DoS) attacks, or stealing information [2]. Today, the speed at which the worm propagates poses a serious security threat to the Internet. A polymorphic worm is a kind of worm that can change its payload in every infection attempt, so it can evade the Intrusion Detection Systems (IDSs), and damage data. It can be used to launch denial of service attacks, cause information theft, and other illegal activities that lead, for example, to high financial losses. Issues regarding the worm attacks and methods of defense on Android devices have been discussed in detail in [3].

Polymorphic worms are on the rise and this calls for cybersecurity countermeasures. Traditional signature-based intrusion detection systems cannot keep up with the ever-changing threats. Consequently, there is an increasing urgency to come up with detection mechanisms that can change their strategy as often as required by polymorphic worms. Varieties of methods have been put forth by researchers to counter this problem including anomaly detection, machine learning, honeypot-based approaches among others [4]. However, even after these attempts, zero-day polymorphic worm detection remains a challenge because they generate new variants, which cannot be detected through regular means [5].

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

These days, the use of honeynets is to find and analyze destructive acts on network systems has become a very auspicious method. Honey nets are groups of deliberately unsecure computers that are meant to lure and capture attackers to gather useful information regarding their strategies and activities [6]. It is possible for the researchers to deploy honeypots in different network segments strategically in order to obtain knowledge on how polymorphic worms are propagated and their characteristics which may help in coming up with better countermeasures. Moreover, the inclusion of host-based monitoring within the design of the honeynet improves detection capabilities by providing detailed information about individual hosts' behavior [7]. Our proposed double-honeynet system will bridge this gap and provide an encompassing methodology for recognizing as well as mitigating zero-day polymorphic worm attacks by combining both host-based intrusion detection mechanism and network-based one.

In this context, the Honeywall—a specialized security device combining the functionalities of a honeynet and a firewall—emerges as a potent tool in the battle against zero-day polymorphic worms. By directing suspicious traffic through the Honeywall, it not only captures and analyzes the traffic but also allows for real-time monitoring and interaction with potential threats. This dual approach ensures a robust defense mechanism that can adapt to the evolving nature of polymorphic worms. The detailed analysis provided by Honeywall deployments can significantly contribute to understanding the attack vectors and developing proactive defense strategies. Consequently, the integration of Honeywall in a double-honeynet system presents a promising frontier in the detection and mitigation of zero-day polymorphic worm attacks, enhancing the overall resilience of network security frameworks.

Roadmap: Section 2 is Related Works, which gives an overview of work related to the current research. Section 3 discusses a Double-Honeynet Architecture. In Section 4, we discuss the Double-honeynet System Configurations in detail. Experimental results and performance evaluation are reported in Section 5. Then, in Section 6 we present the challenges and future research directions. Finally, section 7 is the conclusion, which summarizes and concludes the research paper.

2. Related Work

Researchers have been studying how to detect polymorphic worms for several years. This section sorts out previous research work under three categories: Publications by Application, Publications by Year and Publications by Type. By doing this, we intend to highlight the extent of knowledge in this area and point out areas where our study is relevant.

2.1. Publications by Application

The field of cybersecurity is faced with a significant challenge in detecting Zero-day Polymorphic worms. This is because these worms are evolving and mutating very fast making them difficult to be detected by the traditional method that involves using signature-based detection. Various methods have been developed over the years to help in combating this type of worm, which elicits elusive threats. Table 1 represents some related research work sorted by detection techniques of zero-day polymorphic worms.

The number of publications on zero-day polymorphic worms fluctuated in the last ten years with random increments and decrements. This inconsistency resulted from various factors like changing focus of studies, advancements in detection technologies as well as changes in threat landscape per se. Notably, periods marked by high attention might be accompanied by outbreaks or improvements in techniques for detecting that can prompt researchers to explore new ways of doing things or make some adjustments to what they already have. On the other hand, the publication rates may fall when there is relative peace within cyberspace or other cybersecurity threats are getting precedence over research. Furthermore, the continuing evolving nature of zero-day polymorphic worms creates difficulties for keeping a long-lasting interest and impetus for research since their perpetrators regularly alter their approaches to counteract identification. Therefore, it is important to understand exactly why these spikes occur because it can help us get better insight into dynamic forces behind scientific directions and develop effective strategies for combating emerging cyber risks.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

Table 1: Publications by Application

Reference	Published Year	Detection Technique	Advantages	Disadvantages
[4]	2014	Hardware-assisted Detection	Offloads detection tasks to specialized hardware.	Cost and deployment complexity.
[5]	2018	Game Theory	Models worm propagation dynamics for detection strategies.	Limited real-world application due to complexity.
[8]	2013	Signature-based Detection	Effectively detects known polymorphic worms.	Ineffective against zero-day polymorphic worms.
[9]	2005	Anomaly-based Detection	Can detect previously unseen polymorphic worms.	High false positive rates.
[10]	2006	Behavioral Detection	Can identify deviations from normal network behavior.	Limited to specific network environments.
[11]	2012	Dynamic Analysis	Analyzes worm behavior in real-time.	High computational overhead.
[12]	2022	Machine Learning	Can adapt to new polymorphic worm variations.	Requires labeled training data.
[13]	2019	Heuristic-based Detection	Effective against previously unseen variations.	May produce false negatives.
[14]	2014	Hybrid Detection	Combines multiple detection techniques for improved accuracy.	Complexity in integration and management.
[15]	2007	Code Emulation	Provides insights into worm payload behavior.	Limited effectiveness against highly obfuscated worms.
[16]	2007	Network Traffic Analysis	Analyzes traffic patterns to identify suspicious activity.	May miss stealthy worm variants.
[17]	2014	Cloud-based Detection	Scalable and adaptable to evolving threats.	Dependence on network connectivity and cloud infrastructure.
[18]	2021	Deep Learning	Capable of learning complex patterns in worm behavior.	Requires substantial computational resources and training data.
[19]	2019	Software-defined Networking	Allows for dynamic network adaptation to worm threats.	Requires specialized network infrastructure.
[20]	2023	Blockchain-based Detection	Provides decentralized and tamper-resistant detection.	Scalability and performance issues.

2.2. Publications by Type

The publication landscape of the last ten years concerning zero-day polymorphic worms has been varied to reflect the interdisciplinary nature of cybersecurity research. These academic journals analyze studies with deeper and theoretical frameworks as well as empirical findings describing ways through which polymorphic worm threats can be detected and mitigated against. Additionally, by disseminating ideas about innovative approaches, emerging

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

trends and practical applications in the field, conference proceedings have offered researchers a platform where they meet to share ideas thereby allowing for exchange of thoughts among participants. Besides, books and review papers contain useful information on real-life cases, pragmatic challenges, or industry perspectives on fighting back polymorphic worm's attacks. This paper demonstrates how different types of publications around zero-day polymorphic worms have resulted in various audiences involved in this kind of research which includes fostering collaborations between academic researchers and practitioners then further leading to modifications in cyber security practices as well as technologies. Figure 1 shows the percentage of each publication type in this research field for the last 10 years.

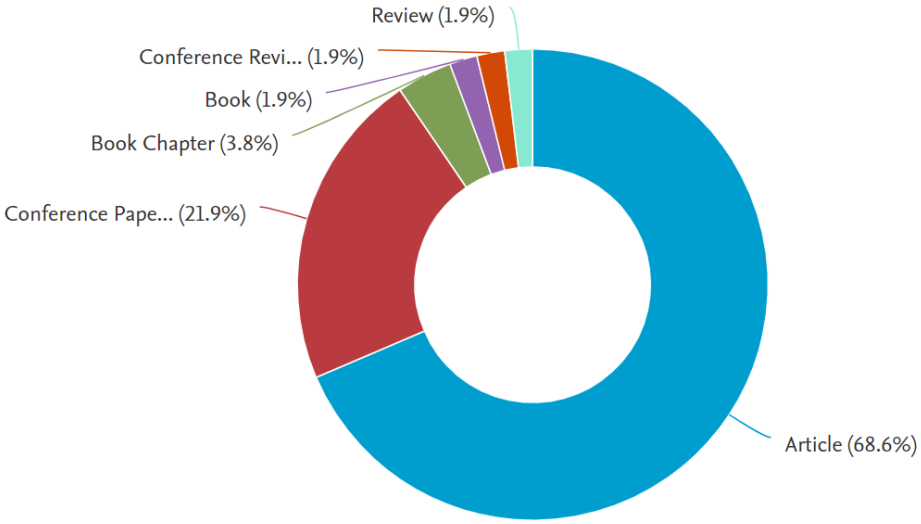

Figure 1. Publications by Type

3. Double-Honeynet Architecture

The purpose of Double-honeynet system is to detect unknown (i.e., previously unreported) worms automatically. A key contribution of this system is the ability of distinguishing worm activities from normal activities without any involvement of experts in the field.

Figure 2 shows the main components of the Double-honeynet system. Firstly, the incoming traffic goes through the Local Router, which samples the unwanted inbound connections and redirects the samples' connections to Honeynet 1. As the redirected packets pass through the Local Router, Packet Capture (PCAP) library is used to capture the packets and then to analyze their payloads to contribute to the signature generation process.

The Local Router is configured with publicly accessible addresses, which represent wanted services. Connections made to other addresses are considered unwanted and redirected to Honeynet 1 through the Internal Router. Once Honeynet 1 is compromised, the worm will attempt to make outbound connections to attack another network. The Internal Router is implemented to separate the Double-honeynet from the Local Area Network (LAN). This Router intercepts all outbound connections from Honeynet 1 and redirects those to Honeynet 2, which does the same task forming a loop. The looping mechanism allows us to capture different instances of the polymorphic worm as it mutates on each loop-iteration. We stop the loop after a considerable amount of time in order to collect polymorphic worms.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

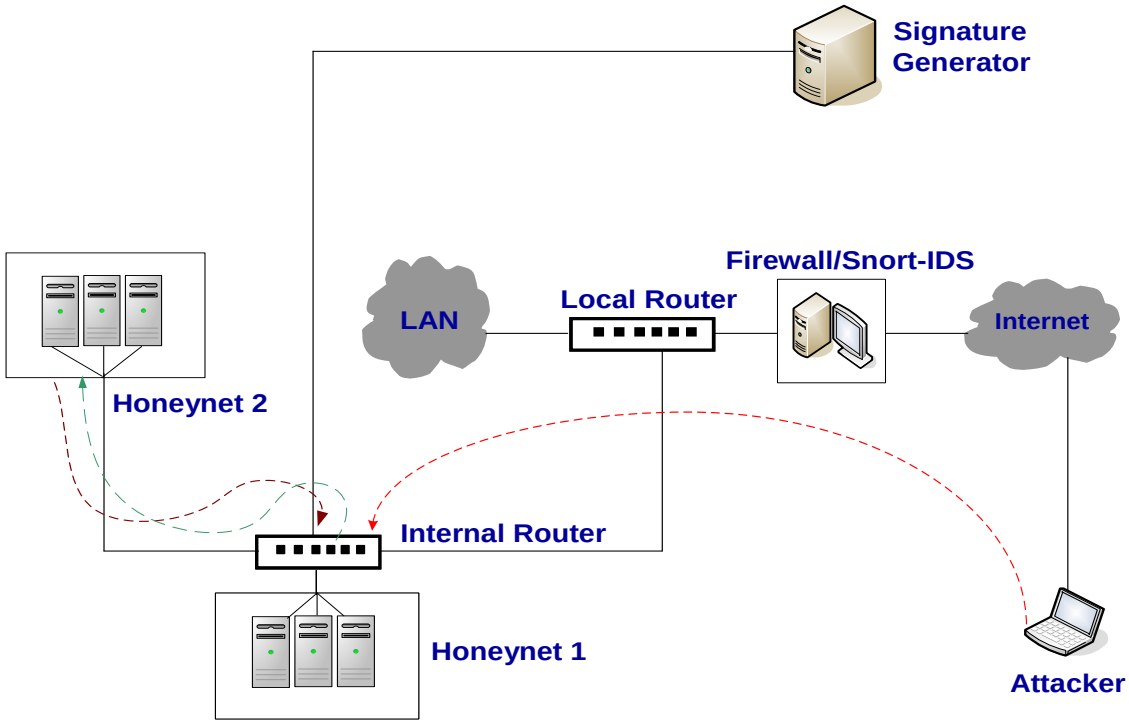

Figure 2. Double-honeynet System

Only those packets that make outbound connections are considered as polymorphic worms, and hence the Double-honeynet system forwards only the packets that make outbound connections. This policy is in place because benign users do not try to make outbound connections if they are faced with non-existing addresses. In fact, our system collects other malicious activities, which do not intend to propagate themselves but to attack targeted machines only. Such malicious attack is out of our work scope.

When Honeynet 1 and Honeynet 2 collect enough instances of worm payloads, they are forwarded to the Signature Generator component, which generates signatures automatically using specific algorithms.

For example, as shown in Figure 2, if the Local Router suspects Packet 1 (P1), Packet 2 (P2), and Packet 3 (P3) to be malicious, it redirects them to the Honeynet 1 through the Internal Router. Among these three packets, P1 and P2 make outbound connections and Internal Router redirects these outbound connections to Honeynet 2. In Honeynet 2, P1 and P2 change their payloads and become $P1'$ and $P2'$ respectively (i.e., $P1'$ and $P2'$ are the instances of P1 and P2). Therefore, in this case, $P1'$ and $P2'$ make outbound connections and the Internal Router redirects these connections to Honeynet 1. In Honeynet 1, $P1'$ and $P2'$ change their payloads and become $P1''$ and $P2''$ respectively (i.e., $P1''$ and $P2''$ are also other instances of P1 and P2).

Now, P1 and P2 are found malicious because of the outbound connections. Therefore, Honeynet 1 forwards P1, $P1''$, P2, $P2''$ to the Signature Generator for signature generation process. Similarly, Honeynet 2 forwards $P1'$ and $P2'$ to the Signature Generator for signature generation process.

In this scenario, P3 does not make any outbound connection when it gets to Honeynet 1. Therefore, P3 is not considered malicious.

3.1. Software

The software tools used in the Double-honeynet system are introduced below.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

3.1.1. Honeywall Roo CDROM

The Honeywall Roo CDROM version 1.4 is downloaded from the Honeynet Project and Research Alliance. It provides data capture, control and analysis capabilities [21], [22]. Most importantly, it monitors all traffic that go in and out of the Honeynet. Honeywall Roo CDROM runs Snort-inline, an Intrusion Prevention System based on the Intrusion Detection System Snort. Snort-inline either drops unwanted packets or modifies them to make them harmless. It records information of all the activities in the Honeynet using Sebek. It runs the Sebek server, while the Sebek clients run on the Honeypots. The clients then send all captured information to the server. For management and data analysis, it uses the Walleye Web interface. Walleye also works as a maintenance interface, but there is a command line tool and a dialog menu that can also be used to configure and maintain the Honeywall.

3.1.2. Sebek

Sebek is a data capture tool which mainly records keystrokes, but also all other types of sys_read data [23]. It records and copies all activity on the machine including changes to files, network communications, etc. The main method it uses is to capture network traffic and reassemble the TCP flow. This is in the case of unencrypted data. Encrypted data are another problem, because Sebek can only reassemble it in its encrypted form. Instead of breaking the encryption, Sebek circumvents it by getting the data from the Operating System's kernel. Sebek has a client-server architecture. On the client side, it resides entirely in the Operating System kernel. Whenever a system call is made, Sebek hijacks it by redirecting it to its own *read ()* call. This way Sebek can capture the data prior to encryption and after decryption.

After capturing the data, the client sends it to the server, which saves it in a database or simply logs the records. The server is normally on the Honeywall machine in the case of a Honeynet, and it collects data from all the Honeypots and puts it all together for analysis.

To prevent detection by intruders, Sebek employs some obfuscation methods. On the client, it is completely hidden from the user and therefore, from an intruder on the system as well. This is however, not enough because the data that are captured have to be sent to the Server, thereby exposing itself. Sebek uses a covert channel to communicate with the server. It generates packets to be sent inside Sebek without using the TCP/IP stack and the packets are sent directly to the driver bypassing the raw socket interface. The packets are then invisible to the user and Sebek modifies the kernel to prevent the user from blocking transition of the packets. Figure 3 shows Sebek Deployment.

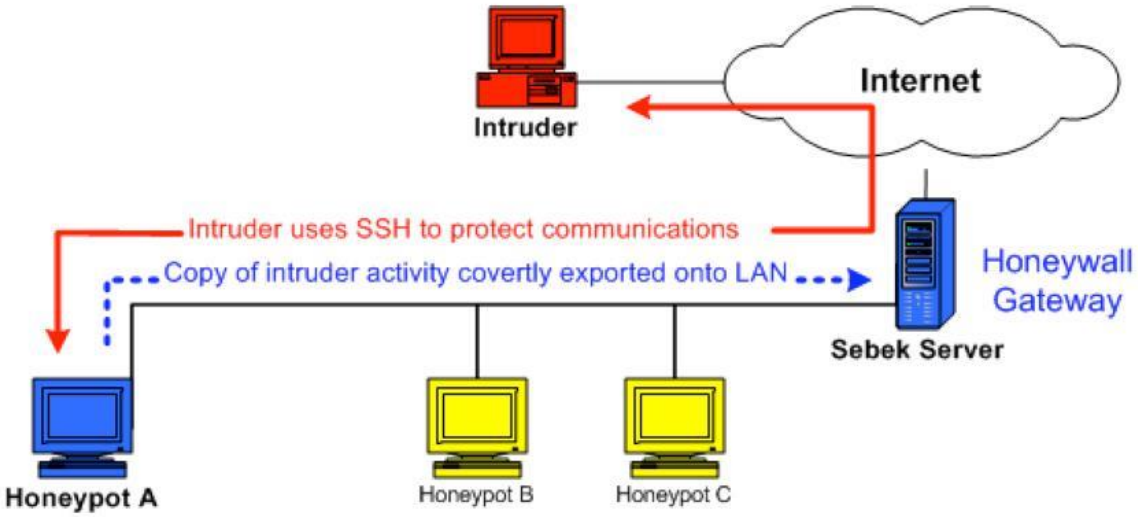

Figure 3. Sebek Deployment [Source: Reference [23], http://old.honeynet.org/papers/sebek.pdf]

In the case of multiple clients, there is a risk of the clients seeing each other's packets. Sebek configures its own raw socket interface on the clients to ignore all incoming Sebek packets. Only the server can receive Sebek packets. Due to its comprehensive log capabilities, it can be used as a tool for forensics data collection. It has a Web interface that can perform data analysis.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

3.1.3. Snort-inline

Snort_inline is a modified version of Snort. It is "*An Intrusion Prevention System (IPS) that uses existing Intrusion Detection System (IDS) signatures to make decisions on packets that traverse snort_inline*". The decisions are usually drop, reject, modify, or allow [24].

4. Double-honeynet System Configurations

In this section, we discuss the Double-honeynet system architecture, and configuration using VMware.

4.1. Implementation of Double-honeynet Architecture

Figure 4 shows the architecture of the Double-honeynet system, implemented using VMware Workstation version 7 on PC Intel Pentium 4, 3.19-GHZ CPU, 8GB RAM, and the PC running on Microsoft Windows. The operating system of that personal computer is referred as host operating system in Figure 4. The host machine was connected to our home router and it accessed the Internet through it.

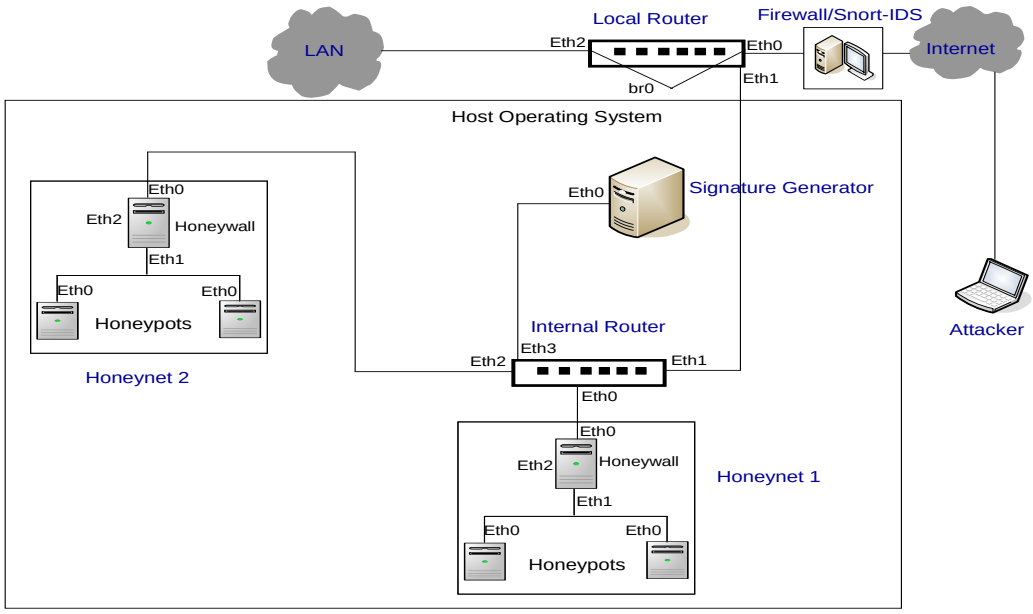

Figure 4. Double-honeynet Architecture

We used virtual machine to deploy the Double-honeynet system due to the lack of resources and to keep the establishment low-cost. One personal computer (PC) was used and VMware Workstation was installed on it. The VMware Workstation is a software package that gives its users the opportunity to create virtual machines that constitute virtual networks interconnected with each other. Thus, we created the Double-honeynet system as a virtual network seen from the outside world as an independent network. Attackers could locate the Honeypot and attack it. The Honeypot was transparently connected to the Internet through the Honeywall, which in turn intercepted all outbound and inbound traffic. Therefore, malicious traffic targeting the Honeypot (inbound) or malicious traffic generated by the compromised Honeypot (outbound) were available to us from the Honeywall for further analysis and investigation. As we mentioned before, Honeynet 1 and Honeynet 2 were configured to deliver unlimited outbound connections. The *Internal Router* was used to protect our local network by redirecting all outbound connections from Honeynet 1 to Honeynet 2 and vice versa.

4.2. Double-honeynet Configurations

Our Double-honeynet system contains six components, which are Local Router, Internal Router, LAN, Honeynet 1, Honeynet 2, and Signature Generator. The subnet mask for each subnet (whether Local Router, Internal Router, LAN, Honeynet 1, Honeynet 2, and Signature Generator) is consequently 255.255.255.0. The following subsequent sections discuss the configurations of each component.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

4.2.1. Local Router Configuration

As we mentioned before, Local Router's function is to pass unwanted traffic to the Honeynet 1 through the Internal Router. For example, if the IP address space of our LAN is 212.0.50.0/24, with one public Web server, the server's IP address is 212.0.50.19. If an attacker outside the network lunches a worm attack against 212.0.50.0/24, the worm scans the IP address space of victims. It is highly probable that an unused IP address, e.g., 212.0.50.10 will be attempted before 212.0.50.19. Therefore, Local Router will redirect the packet to Honeynet 1 through Internal Router. After the worm compromises the Honeynet 1, the worm will try to make an outbound connection to harm another network. We configured the Internal Router to protect the LAN from worms' outbound connections. The Internal Router intercepts all outbound connections from Honeynet 1 and redirects them to Honeynet 2, which performs the same task being done by the Honeynet 1 forming loop connections. Below are the details of the Local Router machine properties and IPtables configuration.

The machine operates on Ubuntu Linux 9.10 and is equipped with three network cards: Eth0, Eth1, and Eth2. Eth0 and Eth2 function as bridged LAN ports, while Eth1 is dedicated to connecting the Local Router with Honeynet 1 via the Internal Router. The IP address assigned to Eth1 is 192.168.50.20.

Before setting IPtables, IP forwarding was enabled on the Local Router by editing the /etc/sysctl .conf file to include the following line: Net.ipv4.ip_frowrd =1.

The settings of the Network Address Translator (NAT) in the kernel using IPtables are as follows:

- Do not translate packets going to the real public server:

```
# iptables -t nat -A PREROUTING -m physdev --physdev-in eth0 -d 212.0.50.19 -j RETURN
```

- Translate all other packets going to the public LAN to Internal Router:

```
# iptables -t nat -A PREROUTING -m physdev --physdev-in eth0 -d 212.0.50.0/24 -j DNAT --to 192.168.50.22
```

4.2.2. Internal Router Configuration

Again, as mentioned before, the Internal Router's function is to protect the LAN from worms' outbound connections and to redirect the outbound connections from Honeynet 1 to Honeynet 2 and vice versa. Let us investigate more about the Internal Router machine properties and IPtables configuration in the following texts.

The machine operates on Ubuntu Linux 9.10 and is equipped with four network cards, identified as Eth0, Eth1, Eth2, and Eth3. Each network card serves a distinct function: Eth0 connects the internal router to the Honeynet1-clients, Eth1 links the internal router with the local router, Eth2 connects the internal router to the Honeynet 2-clients, and Eth3 links the internal router to the signature generator.

The IP addresses assigned to these network cards are as follows: Eth0: 192.168.51.20, Eth1: 192.168.50.22, Eth2: 192.168.58.20 and Eth3 192.168.55.20.

Before we set the Iptables rules, we enable the IP Forwarding in Internal Router by editing /etc/sysctl .conf file as follows: Net.ipv4.ip_frowrd =1

The settings of the Network Address Translator (NAT) in the kernel using IPtables are as follows:

- Translate packets coming in from eth1 to the Honeynet 1

```
# iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 192.168.51.22
```

- From Honeynet 1 don't translate packets to the signature generator

```
# iptables -t nat -A PREROUTING -i eth0 -s 192.168.51.22 -d 192.168.55.22 -j RETURN
```

- From Honeynet 1 translate all other packet to Honeynet 2

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 192.168.58.22
```

- From Honeynet 2 don't translate packets to the Signature generator

```
# iptables -t nat -A PREROUTING -i eth0 -s 192.168.58.22 -d 192.168.55.22 -j RETURN
```

- From Honeynet 2 translate all other packet to Honeynet 1

```
# iptables -t nat -A PREROUTING -i eth0 -j DNAT --to 192.168.51.22
```


DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

4.2.3. LAN Configuration

As we mentioned before, we have one public Web server in our LAN with this IP address: 212.0.50.19. Below are the details of the public Web server machine properties.

The machine operates on the Ubuntu Linux 9.10 operating system. It is equipped with a single network card, designated as Eth0. The IP address assigned to this network card (Eth0) is 212.0.50.19.

4.2.4. Honeynet 1

As shown in Figure 4, Honeynet 1 contains Honeywall and two Honeypots. The main function of the Honeynet 1 is to capture polymorphic worms' instances. Below are the details of the Honeywall machine properties and configuration.

The machine is equipped with three network cards: Eth0, Eth1, and Eth2. Eth0 is responsible for connecting Honeynet 1 with Honeynet 2 via the Internal Router, while Eth1 connects Honeynet 1 with its clients, the Honeypots. Eth2 serves as the Management interface. The IP addresses assigned to these interfaces are as follows: Eth0 uses 192.168.51.22, Eth1 uses 192.168.52.20, and Eth2 uses 192.168.40.7.

For Honeywall configurations, the Honeynet public IP addresses, which represent the external IPs for the Honeypots and are essentially the IPs of the attackers, are 192.168.52.22 and 192.168.52.23. The Honeynet network is identified using CIDR (Classless Inter-Domain Routing) notation as 192.168.52.0/24, with a broadcast address of 192.168.52.255.

The management interface, used for remote management via SSH and the Walleye Web interface, is assigned to Eth2. Its IP address is 192.168.40.7, with a network mask of 255.255.255.0. The default gateway for this interface is 198.168.40.1, and the DNS server IP is 192.168.40.2. The SSHD listens on port 22, and the allowed TCP ports for the management interface are 22 and 443. Access to the management interface is restricted to IP addresses within the 192.168.40.0/24 range.

Regarding firewall restrictions, the Double-Honeynet is configured to permit unlimited outbound connections. Sebek, a data capture tool used on Honeypots, has two components: a client that captures attackers' activities and sends the data covertly to the server, and a server that collects this data. The server typically runs on the Honeywall gateway. The destination IP address for Sebek packets is 192.168.52.20, and the destination UDP port is 1101.

The honeypot machines have specific properties and configurations. Honeypot 1 operates on Microsoft Windows and uses a single network card, Eth0, which is assigned the IP address 192.168.52.22. Honeypot 2 runs on Ubuntu Linux 9.10 and utilizes one network card, Eth0, with the IP address 192.168.52.23.

4.2.5. Honeynet 2 Configuration

Honeynet 2 contains Honeywall and two Honeypots. Honeynet 2 function is to capture polymorphic worm's instances. Below are the details of the Honeywall machine properties and configuration.

The machine properties encompass various network configurations and specifications for effective operation. Firstly, concerning network connectivity, the system boasts three network cards: Eth0, Eth1, and Eth2, each serving distinct functions within the setup. Eth0 facilitates the connection between Honeynet 2 components through the Internal Router; Eth1 links Honeynet 2 with its clients (Honeypots), while Eth2 serves as the designated Management interface.

Assigning IP addresses to these network interfaces further delineates their roles within the network. Eth0 is assigned 192.168.58.22, Eth1 holds 192.168.59.20, and Eth2 is designated as 192.168.40.8.

Additionally, the configuration of the Honeywall, a pivotal component, is detailed for effective management and security. This includes specifying the public IP addresses for the honeypots (192.168.59.22 and 192.168.59.23), defining the Honeynet Network's CIDR notation (192.168.59.0/24), and establishing the broadcast address (192.168.59.255).

A critical aspect of the Honeywall setup involves the Management Interface, essential for remote management. Eth2 is dedicated to this purpose, with its IP address set to 192.168.40.8. The interface's network mask is configured as 255.255.255.0, with a default gateway at 198.168.40.1. Furthermore, DNS server IP for honeywall gateway is 192.168.40.2. Moreover, SSH and Walleye Web interfaces are accessible through this interface, with

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

SSHD listening on port 22 and TCP ports 22 443 permitted for access. Access to the management interface is restricted to the IP range 192.168.40.0/24.

Furthermore, Firewall Restrictions are configured to enable unlimited outbound connections, as per the specified settings. Sebek Variables are also configured, specifying the destination IP address (192.68.59.20) and UDP port (1101) for Sebek Packets.

Lastly, the configuration details of the individual Honeypots are provided. Honeypot 1 runs on Microsoft Windows, with its sole network card (Eth0) assigned the IP address 192.168.59.22. Conversely, Honeypot 2 operates on Ubuntu Linux 9.10, with similar network setup and IP address assignment (192.168.59.23). These specifications collectively form the foundation of an intricately configured and interconnected honeypot network, primed for monitoring and analysis of malicious activities.

4.2.6. Signature Generator Configuration

The function of the signature generator is to generate signatures for polymorphic worm's samples. This machine runs on Ubuntu Linux 9.10 as its operating system. It possesses a single network card, Eth0, and its assigned IP address is 192.168.55.22.

5. Experimental Results

After the successful implementation of Double-honeynet network, we connected the Double-honeynet network to the Internet for seven days to collect attacks. Of course, we checked it periodically to make sure that the network is working properly, to see if the Honeywall serves its purposes, and to examine the data collected. During the seven days, our Double-honeynet system has collected different polymorphic worm's instances and other attacks data. We used Walleye, the Honeywall's Web graphical user interface, to analyze the data.

5.1. Numerical Results

To simulate the detection of unknown polymorphic worms, we removed the current signature patterns for the Allaple, Conficker, Balster, and Sasser polymorphic worms from all the Snort instances so that our system can generate new signatures as if the worm is unknown (i.e.,not previously recorded). Once our system has run with the activated polymorphic worm, a new signature would be generated that can be exported to Snort or Bro IDS. Table 2 shows some samples of polymorphic worms that were captured by the Double honeynet system.

Table 2: Polymorphic Worm Instances

Polymorphic Worm	**Number of instances**
Allaple worm	3511
Conficker worm	3228
Blaster worm	2817
Sasser worm	2452

Below we show some of the payloads of Blaster worm instances as an example for the collected worms. We have converted the blaster exe files to decimal. Fig 5,6,7,8 show the payloads for 4 different instances of Blaster worm.

Figure 5 represents the decimal representation of the payload for the first instance of the Blaster worm, displayed in a 16-column format. The initial rows contain similar values indicative of the executable (exe) header, common across instances due to its role in file metadata. Subsequent rows, after the header, show the actual worm payload data, which varies due to the polymorphic nature of the worm. This variability in the payload data is crucial for signature generation, as it helps distinguish the worm's unique behaviors. To create an accurate signature, the exe header must be removed, focusing solely on the unique payload to ensure the signature does not merely replicate the header information.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

```
77 90 144 0 3 0 0 0 4 0 0 0 255 255 0 0
184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 248 0 0 0
14 31 186 14 0 180 9 205 33 184 1 76 205 33 84 104
105 115 32 112 114 111 103 114 97 109 32 99 97 110 110 111
116 32 98 101 32 114 117 110 32 105 110 32 68 79 83 32
109 111 100 101 46 13 13 10 36 0 0 0 0 0 0 0
152 84 62 146 220 53 80 193 220 53 80 193 220 53 80 193
130 23 91 193 223 53 80 193 167 41 92 193 213 53 80 193
95 41 94 193 198 53 80 193 234 19 90 193 125 53 80 193
239 23 117 193 222 53 80 193 6 22 76 193 208 53 80 193
38 22 73 193 207 53 80 193 220 53 81 193 231 52 80 193
234 19 91 193 143 53 80 193 35 21 84 193 221 53 80 193
82 105 99 104 220 53 80 193 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 80 69 0 0 76 1 4 0
172 88 119 68 0 0 0 0 0 0 0 0 224 0 14 33
11 1 6 0 0 48 10 0 0 144 2 0 0 0 0 0
137 26 0 0 0 16 0 0 0 64 10 0 0 0 0 16
0 16 0 0 0 16 0 0 4 0 0 0 0 0 0 0
4 0 0 0 0 0 0 0 0 16 13 0 0 16 0 0
0 0 0 0 2 0 0 0 0 0 0 16 0 0 16 0 0
0 0 16 0 0 16 0 0 0 0 0 0 0 16 0 0 0
228 190 10 0 127 0 0 0 76 172 10 0 180 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 144 12 0 232 127 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
20 144 0 0 104 3 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
```

Figure 5. First instance of Blaster worm

Figure 6 shows the decimal representation of the payload for the second instance of the Blaster worm, organized into 16 columns. Similar to the first instance, the initial rows display the executable (exe) header, characterized by repetitive values such as '77 90 144', which are standard across different instances due to their role in file metadata. Following these initial rows, the unique worm payload data begins, displaying the polymorphic nature of the worm with varying values that differ from the first instance. This variability in the payload, after removing the exe header, is essential for generating accurate signatures, as it highlights the worm's distinct behaviors and mutations, enabling effective detection and mitigation of the Blaster worm.

```
77 90 144 0 3 0 0 0 4 0 0 0 255 255 0 0
184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 128 0 0 0
14 31 186 14 0 180 9 205 33 184 1 76 205 33 84 104
105 115 32 112 114 111 103 114 97 109 32 99 97 110 110 111
116 32 98 101 32 114 117 110 32 105 110 32 68 79 83 32
109 111 100 101 46 13 13 10 36 0 0 0 0 0 0 0
80 69 0 0 76 1 3 0 42 124 55 63 0 0 0 0
0 0 0 0 224 0 15 1 11 1 2 55 0 32 0 0
0 16 0 0 0 80 0 0 240 113 0 0 0 96 0 0
0 128 0 0 0 0 64 0 0 16 0 0 0 2 0 0
1 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0
0 144 0 0 0 16 0 0 0 0 0 0 2 0 0 0
0 0 16 0 0 16 0 0 0 0 16 0 0 16 0 0
0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0
0 128 0 0 72 1 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 99 111 100 101 0 0 0 0
0 80 0 0 0 16 0 0 0 0 0 0 0 2 0 0
0 0 0 0 0 0 0 0 0 0 0 0 128 0 0 224
116 101 120 116 0 0 0 0 0 0 32 0 0 0 96 0 0
0 20 0 0 0 2 0 0 0 0 0 0 0 0 0 0
0 0 0 0 64 0 0 224 114 115 114 99 0 0 0 0
0 16 0 0 0 128 0 0 0 2 0 0 0 22 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 64 0 0 192
49 46 50 52 0 85 80 88 33 12 9 2 9 65 251 242
189 156 246 111 79 48 84 0 0 213 17 0 0 32 44 0
```

Figure 6. Second instance of Blaster worm

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

Figure 7 represents the payload data of the third instance of the Blaster worm, displayed in a decimal format. Each row contains a sequence of numbers, with 16 columns per row, showing the raw byte values of the Blaster worm executable file. The initial rows, displaying repetitive patterns, correspond to the executable header common to all instances, which needs to be removed for accurate signature generation. The subsequent rows exhibit varying byte patterns indicative of the worm's actual payload. This differentiation helps in distinguishing the polymorphic characteristics of the worm, essential for developing new detection signatures. The presented data highlights the importance of filtering out common headers to focus on the unique aspects of each worm instance for effective analysis and signature creation.

```
77 90 144 0 3 0 0 0 4 0 0 0 255 255 0 0
184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 240 0 0 0
14 31 186 14 0 180 9 205 33 184 1 76 205 33 84 104
105 115 32 112 114 111 103 114 97 109 32 99 97 110 110 111
116 32 98 101 32 114 117 110 32 105 110 32 68 79 83 32
109 111 100 101 46 13 13 10 36 0 0 0 0 0 0 0
170 31 32 159 238 126 78 204 238 126 78 204 238 126 78 204
201 184 51 204 242 126 78 204 121 186 48 204 230 126 78 204
45 113 17 204 254 126 78 204 45 113 19 204 247 126 78 204
238 126 79 204 111 127 78 204 201 184 35 204 95 126 78 204
201 184 32 204 93 126 78 204 201 184 52 204 239 126 78 204
201 184 54 204 239 126 78 204 82 105 99 104 238 126 78 204
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
80 69 0 0 76 1 4 0 135 137 238 70 0 0 0 0
0 0 0 0 224 0 2 33 11 1 8 0 0 96 17 0
0 48 2 0 0 0 0 0 7 67 0 0 0 16 0 0
0 128 16 0 0 0 0 16 0 16 0 0 0 16 0 0
4 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0
0 96 19 0 0 16 0 0 0 0 0 0 2 0 0 0
0 0 16 0 0 16 0 0 0 0 16 0 0 16 0 0
0 0 0 0 16 0 0 0 68 95 17 0 127 0 0 0
12 72 17 0 220 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 128 18 0 4 211 0 0 212 246 0 0 28 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 76 88 3 0 64 0 0 0
0 0 0 0 0 0 0 0 180 241 0 0 140 4 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 46 116 101 120 116 0 0 0
195 79 17 0 0 16 0 0 0 80 17 0 0 16 0 0
0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 96
46 100 97 116 97 0 0 0 148 12 1 0 0 96 17 0
```

Figure 7. Third instance of Blaster worm

Figure 8 shows the payload data of the fourth instance of the Blaster worm, represented in decimal format. Similar to the previous instances, the initial rows exhibit repetitive numerical patterns, which correspond to the executable header shared by all instances of the worm. These headers must be removed to avoid generating incorrect signatures. The subsequent rows present more varied and unique byte sequences, reflecting the polymorphic nature of the Blaster worm. This variability in the payload is crucial for identifying the distinctive characteristics of each worm instance. Analyzing these differences helps in creating accurate and effective detection signatures by focusing on the unique segments of the worm's code, which differ from the common header.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

```
77 90 144 0 3 0 0 0 4 0 0 0 255 255 0 0
184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 128 0 0 0
14 31 186 14 0 180 9 205 33 184 1 76 205 33 84 104
105 115 32 112 114 111 103 114 97 109 32 99 97 110 110 111
116 32 98 101 32 114 117 110 32 105 110 32 68 79 83 32
109 111 100 101 46 13 13 10 36 0 0 0 0 0 0 0
80 69 0 0 76 1 4 0 42 124 55 63 0 0 0 0
0 0 0 0 224 0 15 1 11 1 2 55 0 22 0 0
0 18 0 0 0 2 0 0 203 17 0 0 0 16 0 0
0 48 0 0 0 0 64 0 0 16 0 0 0 2 0 0
1 0 0 0 0 0 0 0 4 0 0 0 0 0 0 0
0 96 0 0 0 16 0 0 0 0 0 0 2 0 0 0
0 0 16 0 0 16 0 0 0 0 16 0 0 16 0 0
0 0 0 0 16 0 0 0 0 0 0 0 0 0 0 0
0 80 0 0 192 6 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 46 116 101 120 116 0 0 0
88 20 0 0 0 16 0 0 88 20 0 0 0 4 0 0
0 0 0 0 0 0 0 0 0 0 0 0 32 0 0 96
46 98 115 115 0 0 0 0 60 1 0 0 0 48 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 128 0 0 192 46 100 97 116 97 0 0 0
140 8 0 0 0 64 0 0 140 8 0 0 0 26 0 0
0 0 0 0 0 0 0 0 0 0 0 0 64 0 0 192
46 105 100 97 116 97 0 0 192 6 0 0 0 80 0 0
192 6 0 0 0 36 0 0 0 0 0 0 0 0 0 0
```

Figure 8. Fourth instance of Blaster worm

5.2. Payloads Analysis

Each figure above contains 16 columns, each figure from row 1 until row 9 contains the same decimal numbers; this similarity is result of the exe header for each instance. Therefore, in the first step we must remove the exe header from each instance before starting any process to generate a signature for the blaster worm. If we do not remove the exe header, then the signature of the blaster worm will be the exe header, which is not right. We should mention that the objective of this paper is to collect polymorphic worm instances, but generating signatures of the collected samples are out of the scope of this paper.

6. Challenges and Future Research Directions

This section consists of the faced challenges on Detecting Zero-day Polymorphic Worms Using Honeywall and the future research directions beyond this topic.

6.1. Challenges

In the realm of Detecting Zero-day Polymorphic Worms Using Honeywall, several challenges have emerged as indicated in this research paper and this subsection will discuss some of them.

Dynamic Nature of Polymorphic Worms: The dynamic nature of polymorphic worms makes them difficult to detect using traditional signature-based methods. As they change every time, they are not detected by intrusion detection systems (IDS) that use signatures thus posing great danger to network security.

High False Positives and Negatives: Current IDSs struggle with the detection of polymorphic worms leading to high rates of false negatives and false positives alike. This undermines the reliability of threat identification and response mechanisms thereby making it hard for security experts to distinguish between real threats and normal network activities.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

Zero-Day Attacks: Furthermore, zero-day polymorphic worms increase the complexity by developing new variations that cannot be detected by usual approaches. In addition, these attacks exploit unknown vulnerabilities, which expose organizations until patches, or detection systems are developed.

Evolution of Attack Strategies: The perpetrators of worms constantly change their tactics in order to avoid being caught through scanners. These include using obfuscation techniques such as hiding payloads, which can complicate an efficient response system for emerging threats in terms of identification through security networks.

Resource Intensive Detection Mechanisms: Several advanced discovery methods including machine learning and dynamic analysis are computationally expensive and require many labeled training data for them to be effective. This challenge is faced by organizations who lack resources or knowledge for implementing and maintaining complex security infrastructure.

Complexity in Integration and Management: Hybrid detection approaches often involve complicated integration and management processes, which integrate different types of detection techniques to increase their accuracy. Security teams are supposed to figure out how different detection mechanisms work together as well as interpret their outputs, especially within highly dynamic networks.

Privacy and Compliance Concerns: The deployment of honeynets and other deception-based detection mechanisms may raise privacy and compliance concerns, particularly related to the collection and analysis of potentially sensitive network data. There is a delicate balance between effective threat detection on the one hand, regulatory requirements, privacy considerations among others that makes implementation of these technologies by organizations challenging.

Scalability and Performance Issues: Although cloud-based and software-defined, networking (SDN) approaches provide flexibility and adaptability in the face of changing threats; they may result in a performance overhead and scaling problems. Thus, ensuring cybersecurity practitioners must take into account how well detection mechanisms can scale without affecting performance to address this issue.

A comprehensive approach that utilizes advanced detection methods, strong threat intelligence, and proactive security measures is necessary to identify, mitigate, and retaliate against polymorphic worms in a constantly evolving cybersecurity environment.

6.2. Future Research Directions

The Double-honeynet system proposed for detecting zero-day polymorphic worms presents multiple opportunities for future research and development on the cybersecurity. Here are some possible directions:

- Enhanced Signature Generation Algorithms: Although the Double-honeynet system is successful in capturing instances of polymorphic worms, signature generation algorithms can still be improved. Future work could concentrate on developing more accurate and efficient algorithms, which could generate signatures for polymorphic worms in real time thereby increasing its responsiveness to potential threats.
- Dynamic Adaptation Mechanisms: Examining dynamic adaptation mechanisms within the Double-honeynet system may lead to improved detection capabilities. Through incorporation of techniques like machine learning or artificial intelligence, the system could adapt its detection strategies based on evolving worm behaviors; thus, enhancing detection capabilities for zero-day polymorphic worms.
- Behavioral Analyzing Techniques: Complementing signature-based detection through behavioral analyzing techniques can be a more holistic way of detecting polymorphic worms. Possible future research can investigate combining anomaly detection, heuristics, and behavioral analyses to detect abnormalities in networks that may indicate activities associated with polymorphic worms.
- Scalability and Deployment Considerations: Scalability and deployment considerations become of increasing concern as cyber security threats evolve. Future research should optimize Double-honeynet systems for scalability, such that it can handle large-scale networks and high traffic volumes effectively. Additionally, the study could explore seamless deployment approaches for various network environments to enhance the practicality of this system.
- Integration with Threat Intelligence Platforms: Tying up the Double-honeynet system with threat intelligence platforms would provide contextual information useful for threat analysis and response. Future research might examine mechanisms for incorporating threat intelligence feeds into the collected data from external sources enhancing the detection capabilities of this system.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

- Evaluation in Real-World Environments: The experimental results that are outlined here show the effectiveness of Double-honeynet system in controlled environments, but future research should be done to evaluate its performance in real world environments. These actions would help reveal useful ideas concerning the practical efficiency, user friendliness and scalability of this technology in live situations.
- Collaborative Defense Mechanisms: Investigation collaborative defense mechanisms will improve resilience of double honeynet system against polymorphic worm attacks. It is also possible for an upcoming inquiry to identify ways, which enable sharing of threat intelligence and coordination of responses across interconnecting Honeynets. At the same time, think about how such a future study might lead to collective defense measures against emerging threats.

Addressing these future research directions would transform Double-honeynet system into a more robust and efficient mechanism for detecting zero-day polymorphic worms thus enhancing cyber resilience on networked environment.

7. Conclusion

In this paper, we proposed a new approach for detecting zero-day polymorphic worms using a Double-honeynet system. This system can detect and capture polymorphic worm instances that were not previously seen using unique characteristics of honeynets with signature-based detection mechanisms. Having experimented widely and evaluated using different real-world worm samples, we have shown that high detection rates are attained by the Double-honeynet system at low false positive rates. Our findings show that this system effectively detects being infected by polymorphic worms by its real-time generation of accurate signatures where it captures behavioral patterns. Addressing the difficulties in zero-day polymorphic worm detection is what makes this research work's major contribution, which is notorious for its ability to hide away from ordinary ways taken to expose them. Combining honeynet technology with signature-based detection, our method provides a proactive defense mechanism against emerging threats thereby enhancing network security and resilience. Moreover, our study shows the significance of frequent invention and adaptation in cybersecurity investigations. It is vital that proactive detection mechanisms are developed which could identify and ameliorate imminent threats in real-time as cyberspace evolves with more advanced cyber threats. Besides, those future studies accounted for in this paper build on the above to advance the double-honeynet system further and find new ways to improve cybersecurity defense strategies. Addressing these research foci will enable us to strengthen zero-day polymorphic worms' defenses and other emergent cyber threats that contribute towards a safer cyberspace. In conclusion, it can be said that Double honeynet system may be considered as a promising measure aiming at improving network security against zero day polymorphic worms. We believe that further research and development will make notable contributions to advancing cyber security defense mechanisms in this area.

Funding: "This research received no external funding"

Conflicts of Interest: "The authors declare no conflict of interest."

References

[1] Mohammed, M.M.Z.E., Chan, H.A., Ventura, N., Hashim, M., and Amin, I., "A modified Knuth-Morris-Pratt Algorithm for Zero-day Polymorphic Worms Detection", In Proceedings of the 2009 International Conference on Security & Management (SAM 2009), July 13-16, 2009, Las Vegas Nevada, USA, 2 Volumes, CSREA Press, 2009, pp. 652-657.

[2] Mohammed, , M.M.Z.E. and Chan, H.A., "Honeycyber: Automated Signature Generation for Zero-day Polymorphic Worms", Proceedings of the IEEE Military Communications Conference (MILCOM), San Diego, USA, 17-19 Nov. 2008, pp. 1-6.

[3] Mohssen M.Z.E.M, Chan, H.A., Ventura, N., Hashim, M., and Amin, I., "Accurate Signature Generation for Polymorphic Worms Using Principal Component Analysis", Proceedings of IEEE Globecom 2010 Workshop on Web and Pervasive Security (WPS 2010), Miami, Florida, USA,6-10 December 2010, pp. 1555-1560.

[4] R. Kaur and M. Singh, 'A survey on zero-day polymorphic worm detection techniques', IEEE Communications Surveys and Tutorials, vol. 16, no. 3, pp. 1520–1549, 2014.

[5] S. M. A. Sulieman and Y. A. Fadlalla, 'Detecting Zero-day Polymorphic Worm: A Review', in 2018 21st Saudi Computer Society National Computer Conference (NCC), 2018, pp. 1–7.

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024

[6] D. V. Silva and G. D. R. Rafael, 'A review of the current state of Honeynet architectures and tools', International Journal of Security and Networks, vol. 12, no. 4, pp. 255–272, 2017.

[7] N. Kailasanathan, S. Somayaji, S. Fuladi, F. Benedetto, S. Ulaganathan, and G. Yenduri, 'Enhancing Security of Host-Based Intrusion Detection Systems for the Internet of Things', IEEE Access, vol. PP, pp. 1–1, 01 2024.

[8] M. M. Z. E. Mohammed, H. A. Chan, N. Ventura, and A.-S. K. Pathan, 'An automated signature generation method for zero-day polymorphic worms based on multilayer perceptron model', 2013, pp. 450–455.

[9] J. Newsome, B. Karp, and D. Song, 'Polygraph: automatically generating signatures for polymorphic worms', in 2005 IEEE Symposium on Security and Privacy (S&P'05), 2005, pp. 226–241.

[10] M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, 'A multifaceted approach to understanding the botnet phenomenon', in Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement, Rio de Janeriro, Brazil, 2006, pp. 41–52.

[11] U. K. Tupakula and V. Varadharajan, 'Dynamic state-based security architecture for detecting security attacks in virtual machines', Computer Journal, vol. 55, no. 4, pp. 397–409, 2012.

[12] F. Alhaidari et al., 'ZeVigilante: Detecting Zero-Day Malware Using Machine Learning and Sandboxing Analysis Techniques', Computational Intelligence and Neuroscience, vol. 2022, p. 1615528, May 2022.

[13] G A Priyanka , Ashwin Kumar H R , Deepika S , Neha Bindinganavalle, Manjunath G S, 'Detecting Zero Day Malware', INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH & TECHNOLOGY (IJERT), vol. 08, issue. 05, May 2019.

[14] R. Kaur and M. Singh, 'Efficient hybrid technique for detecting zero-day polymorphic worms', 2014, pp. 95–100.

[15] M. Polychronakis, K. G. Anagnostakis, and E. P. Markatos, 'Network-level polymorphic shellcode detection using emulation', Journal in Computer Virology, vol. 2, no. 4, pp. 257–274, Feb. 2007.

[16] S. G. Cheetancheri, 'Collaborative defense against zero-day and polymorphic worms: detection, response and an evaluation framework', University of California at Davis, USA, 2007.

[17] A. A. Shahin, 'Polymorphic Worms Collection in Cloud Computing', arXiv [cs.DC]. 2014.

[18] S. M. Sohi, J.-P. Seifert, and F. Ganji, 'RNNIDS: Enhancing network intrusion detection systems through deep learning', Computers and Security, vol. 102, 2021.

[19] H. Al-Rushdan, M. Shurman, S. H. Alnabelsi, and Q. Althebyan, 'Zero-Day Attack Detection and Prevention in Software-Defined Networks', in 2019 International Arab Conference on Information Technology (ACIT), 2019, pp. 278–282.

[20] D. Denysiuk, O. Geidarova, M. Kapustian, S. Lysenko, and A. Sachenko, 'Blockchain-based Deep Learning Algorithm for Detecting Malware', in International Workshop on Intelligent Information Technologies & Systems of Information Security, 2023.

[21] Know your enemy Honeywall cdrom roo. Available at: https://projects.honeynet.org/honeywall/ (last accessed: August 18, 2012)

[22] The Honeynet Project. Roo CDROM User's Manual, Available at: http://old.honeynet.org/tools/cdrom/roo/manual/index.html (last accessed: August 18, 2012)

[23] Know your enemy Sebek, a kernel based data capture tool. Available at: http://old.honeynet.org/papers/sebek.pdf (last accessed: August 18, 2012)

[24] Snort – The de facto Standard for Intrusion Detection/Prevention. Available at: http://www.snort.org (last accessed: August 18, 2012)

DOI: https://doi.org/10.54216/JCIM.150104
Received: April 01, 2024 Revised: June 05, 2024 Accepted: July 17, 2024