computers

MDPI

Article

A Comparative Experimental Design and Performance Analysis of Snort-Based Intrusion Detection System in Practical Computer Networks

Imdadul Karim, Quoc-Tuan Vien *, Tuan Anh Le and Glenford Mapp

School of Science and Technology, Middlesex University, The Burroughs, London NW4 4BT, UK; Imu.Karim@gmail.com (I.K.); T.Le@mdx.ac.uk (T.A.L.); G.Mapp@mdx.ac.uk (G.M.)

* Correspondence: Q.Vien@mdx.ac.uk; Tel.: +44-208-411-4016

Academic Editor: Yevgeniya Kovalchuk

Received: 27 November 2016; Accepted: 4 February 2017; Published: 7 February 2017

Abstract: As one of the most reliable technologies, network intrusion detection system (NIDS) allows the monitoring of incoming and outgoing traffic to identify unauthorised usage and mishandling of attackers in computer network systems. To this extent, this paper investigates the experimental performance of Snort-based NIDS (S-NIDS) in a practical network with the latest technology in various network scenarios including high data speed and/or heavy traffic and/or large packet size. An effective testbed is designed based on Snort using different muti-core processors, e.g., i5 and i7, with different operating systems, e.g., Windows 7, Windows Server and Linux. Furthermore, considering an enterprise network consisting of multiple virtual local area networks (VLANs), a centralised parallel S-NIDS (CPS-NIDS) is proposed with the support of a centralised database server to deal with high data speed and heavy traffic. Experimental evaluation is carried out for each network configuration to evaluate the performance of the S-NIDS in different network scenarios as well as validating the effectiveness of the proposed CPS-NIDS. In particular, by analysing packet analysis efficiency, an improved performance of up to 10% is shown to be achieved with Linux over other operating systems, while up to 8% of improved performance can be achieved with i7 over i5 processors.

Keywords: network security; intrusion detection system; Snort; parallel processing; network traffic monitoring; experimental performance evaluation

1. Introduction

Social media is playing a significant role in the most of the people's daily life where people are connected together via various online social networking services, e.g., Facebook, Twitter, LinkedIn, etc. With sophisticated communication technology advancement, every user is able to access the complex network using either laptops, desktops, smart phones or even simple gadgets. Organisations and industries are adopting the latest networking/computer systems and technologies to fulfill their business, social and commercial interests. However, they also bring huge security challenges to deal with security breaches, loss of private and confidential data or service disruption, etc. in order to provide a perfect security mechanism for their clients or customers [1–4].

There is a vast number of existing security systems that are not well planned or designed with unnoticeable errors. This could result in many misbehaviours from stealing pasta recipes to governmental classified documents or information stored in highly advanced servers. In order to obtain this confidential information, intruders could entail eavesdropping during an online purchase or gaining access keys for more sensitive information. Furthermore, malicious users can take this opportunity to create their own applications, which could be used to recruit computers known

as "zombies" [5]. These zombies can be controlled remotely to organise a revolutionary attack on a large scale, e.g., Distributed Denial of Service (DDoS) attack [6–9]. Computers and servers affected by DDoS attacks utilise all of the excessive unplanned resources to the attackers, and thus run out of resources and stop providing resources to legitimate operations. DDoS attacks have serious consequences amongst all of the other internet based attacks, e.g., TCP, UDP, ICMP and HTTP flood attacks. The DDoS attacks have targeted not only social networking sites and business enterprises [6,7] but also government websites, forcing them to close down their websites and causing the loss of millions of dollars.

Recently, "Snort" has emerged as an effective open-source solution for more secure network computer systems [10,11]. Snort is a network intrusion prevention and detection system based on a rule-based language combining signature, protocol and anomaly inspection methods. Snort can be configured as either of the following: (i) a Packet Sniffer: Snort captures incoming and outgoing packets from the network and displays the packet details on the console; (ii) a Packet Logger: Snort captures and logs captured packet details in a text file; (iii) a Honeypot Monitor: Snort is able to deceive the hostile party; and (iv) a Network Intrusion Detection System (NIDS): Snort analyses the incoming and outgoing traffic against a set of rules to detect any intrusion in the network. Snort-based NIDS (S-NIDS) has been employed to protect the networks from various possible intrusions and attacks [10,12]. Despite its efficiency in enhancing the network security, the S-NIDS has been shown to perform well when the internet speed is slow with low load [12,13]. However, with advanced technologies, the data rate keeps increasing and the network load becomes heavier in order to provide multiple services with multiple functionalities. This accordingly causes a considerable number of packet drops in the current S-NIDS, through which the attackers may exploit to attack the network. Therefore, a more advanced and sustainable S-NIDS with up-to-date technologies is indispensable to deal with not only high-speed but also heavy-traffic networks. In fact, the industries are currently facing challenges with building a secure system that would ensure stringent confidentiality, integrity and reliability requirements in the current state of internet security and dramatically improved technology. To the best of the authors' knowledge, this work is the first attempt to address this practical issue of deploying S-NIDS in high-speed and heavy-traffic networks with large packet size based on the latest powerful multi-core processors (MCPs) and operating systems (OSs).

To this extent, in this paper, we investigate the experimental performance of S-NIDS in a practical network with the latest technology. In order to provide better performance, local and centralised databases are employed to increase packet processing rates and reduce the number of packet drops for maximising detection probability. The main contributions of this work can be summarised as follows:

- An effective testbed is designed based on Snort using different MCPs, e.g., i3, i5 and i7, with different OSs, e.g., Windows 7, Windows Server and Linux. The test-bed is shown to be effective in analysing the performance of S-NIDS in a variety of practical network scenarios. Specifically, in this work, we consider a network with high data speed and/or heavy traffic and/or large packet size. Here, the high-speed traffic refers to the scenario that a fixed number of packets are transmitted in a short interval, while the heavy traffic means that a large number of packets are generated in total.
- Experimental evaluation is carried out for each network configuration to analyse the performance of the S-NIDS in terms of the number of received packets, the number of analysed packets and the number of dropped packets. In particular, we introduce a new metric, namely packet analysis efficiency (PAE), to facilitate a fair comparison between different network configurations in different testbeds. Through the PAE analysis, we not only derive the optimal combination of the MCPs and OSs with respect to various practical network conditions but also demonstrate the system that would produce less false detection probability while maintaining a high detection rate. Furthermore, the factor responsible for the limited performance of the S-NIDS is determined for each network scenario.

- Considering an enterprise network consisting of multiple virtual local area networks (VLANs), a centralised parallel S-NIDS (CPS-NIDS) is proposed with the support of a centralised database server (CDS). Each subnet employs a S-NIDS based on its specific OS, which can log and detect all the incoming and outgoing packets for that subnet. In the proposed system, the CDS is introduced to deal with a huge number of log files and storage. It is shown that the proposed CPS-NIDS receives a higher number of packets for analysis with a significantly reduced number of packet drops, and thus results in a much higher PAE compared to other counterparts. The novelty of the proposed CPS-NIDS lies in the fact that the incoming traffic can be split into different VLAN groups with respect to different configurations, i.e., MCPs and OSs, to deal with different practical network scenarios.

  By employing the CDS, the entire log file of the CPS-NIDS can be accessed and validated in a centralised manner for the whole enterprise network. This allows us to not only keep track of all attack behaviours to develop and enhance security policies, but also to share the knowledge of new attack behaviours between VLANs, which accordingly helps eliminate the same attacks in the future via a learning process.

The rest of this paper is organised as follows: Section 2 summarises related works on S-NIDS as a motivation for our work. Section 3 discusses the configuration of the S-NIDS and network design parameters in various network deployment scenarios. Section 4 presents the proposed CPS-NIDS. Experimental results are presented in Section 5 to validate the concepts. Finally, Section 6 draws the main conclusions from this paper.

2. Related Work

NIDS has been deployed in organisations as a critical security solution to detect large-scale malicious attacks on application, transport and network layers, along with multiple unexpected service based applications [14–17]. While employing the NIDS can detect and monitor network based threats without violating network policies, all packets in the entire network may not be completely analysed, and thus may cause a considerable delay in high-speed and heavy-load network environments [2,10,18–20]. Therefore, it is vital that the NIDS should be able to deal with a large amount of traffic in a short time frame where all variations of network deployment, e.g., network speed, bandwidth, hardware resources, memory and processing power, need to be taken into account, and several nodes can be used in parallel and concurrently.

An open-source "Snort"-based NIDS, i.e., S-NIDS, has recently emerged as a solution to protect the networks from various possible intrusions and attacks [10,12,18,21,22]. Snort uses a number of pattern matching algorithms such as Aho–Corasick, modified Wu–Manber and low memory key-word trie (lowmem) [23]. However, the most important and well-known fact is that the S-NIDS drops packets significantly when dealing with either a large amount of traffic, high speed or large packet size [12–14,18].

In [19], the performance of S-NIDS was experimentally analysed considering a variety of processors, e.g., Intel Pentiums and Celerons, and OSs, e.g., Windows XP and Vista, for a network operating at 100 Megabits/second (Mbps). Specific technologies and methodologies were investigated in [24–26] to detect network attacks in real time by manipulating the S-NIDS for high-speed and heavy-traffic networks. A hybrid based NIDS combining network and host based IDS was evaluated in [12] using Linux OS against heavy and light incoming traffic with a network speed of 100 Mbps. Furthermore, a parallel technique for improving the NIDS performance was proposed in [27] with different parallelism levels and was then evaluated in [26] for TCP packets in both heavy and light incoming traffic. It is shown that parallelism is able to eliminate a considerable number of packet losses by using higher levels of granularity [28]. Nevertheless, commercial NIDSs are still vulnerable when the new attack signature is unknown. Specifically, it was shown in [29,30] that full protection could not be provided by using only a signature. Another crucial issue is the increase of polymorphic worms that can change their characteristics, behaviours or patterns automatically,

leading to significant threats in current NIDSs [30]. Therefore, it is still possible to bypass those security parameters by an intelligent attacker.

Despite the amount of research done to date, none of the works investigated the performance of the S-NIDS in a practical high-speed and heavy-traffic network with large packet sizes using various MCPs and OPs. To this extent, this paper makes further use of the parallelism technique and proposes a new CPS-NIDS that can both provide a significantly improved performance and enhance the monitoring and assessment process in the whole network towards developing a learning mechanism to cope with intelligent attackers.

3. Network Design of a Testbed for S-NIDS Evaluation

In this section, we first present the architecture of a typical S-NIDS and the functionality of its major components. A typical testbed to evaluate the performance of the S-NIDS is then introduced.

3.1. S-NIDS Architecture

Figure 1 illustrates the architecture of a typical S-NIDS consisting of the following major components: packet decoder, pre-processor, detection engine, logging and alerting system, and output modules [19]. When a packet stream passes through the network, S-NIDS listens and captures the packets. The packet decoder receives packets from multiple network interfaces, such as point-to-point protocol, Ethernet, and Serial Link Internet Protocol, and then organises all of these packets for pre-processing and detection engines. The detection engine is the most important part of Snort, which utilises different times for different lengths of packets, and thus is time critical.

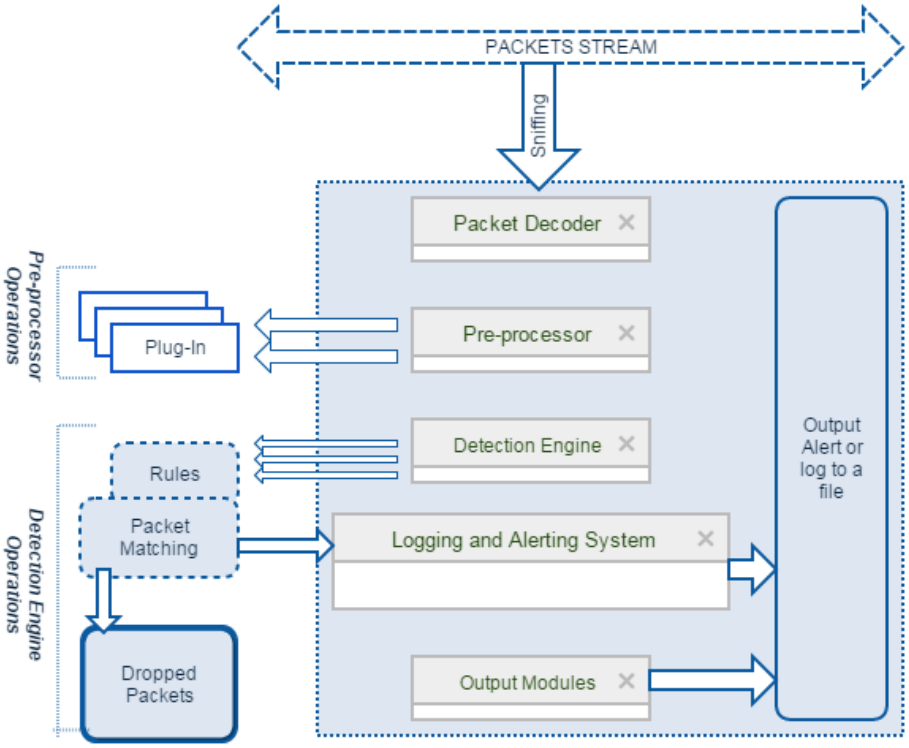

Figure 1. S-NIDS architecture [19].

The mechanism to detect intrusion in data packets is based on Snort rules. (Rules in the S-NIDS have a special structure or syntax to generate alert, e.g., the rule dealing with a file sharing service. In addition, note that it looks for the get string rather than looking into the port number.) The Snort rule is able to read the chain of internal data structures, which have to be matched against all packets.

If a data packet does not match any rule, it will be dropped; otherwise, an appropriate action will be taken [19]. In addition, there is a logging and alerting system to log malicious activities and also to generate an alert. These activities are stored either in a log file, a database locally, or a database server, which can be controlled and managed by the output modules.

As Snort runs in real time, it can drop some packets depending on the data speed and network traffic. The main objective for S-NIDS is therefore to analyse and detect any intrusion coming with incoming and outgoing packet streams with a minimal number of packets dropped, guaranteeing that a maximal number of packets can be analysed for a more efficient S-NIDS. It is well known that the performance of every running application depends on its processing power and memory size. Here, the S-NIDS performance depends mainly on Network Interface Card (NIC), OSs and Random Access Memory (RAM) speeds. To provide powerful processing capabilities, a multi-core system has been further exploited using MCPs to effectively handle multiple programs in a central processing unit (CPU) cycle. Accordingly, in this work, taking into account the practical issues of applying the latest technologies to S-NIDS (note that Snort 3.0 (https://www.snort.org/snort3) supports multiple packet processing threads), we investigate the performance of S-NIDS in various practical network scenarios deploying the latest MCPs along with various OSs.

3.2. Network Deployment Design Parameters and Specifications

Considering the deployment of the S-NIDS, it is expected that all types of attacks coming towards the network could be detected. However, such attack detection may not always be perfect due to the complexity of the practical network. One of the important duties of the S-NIDS is to deal with traffic flow as the attackers can exploit the failure in flow control to pass through the attack signature. Therefore, a high performance system must be introduced to handle the traffic in real time. Although a considerable amount of research works (details can be referred to in Section 2 with references therein) have been carried out to evaluate performance of the S-NIDS, most of the experiments consider a moderate traffic condition with non-realistic traffic flow and under limited conditions.

In this research, we introduce some realistic off-the-shelf hardware specifications for performance evaluation and network design of the S-NIDS. Specifically, we first build a simple testbed to evaluate the performance of the signature-based S-NIDS (Signature-based S-NIDS is chosen in the experiment due to its popularity and the fact that its code or algorithm can be manipulated according to organisation needs [31]) using individual OSs including Windows 7, Windows Server and Linux (Fedora) on i5/i7 machines in different network configurations including high-speed data flow and/or heavy traffic and large packets. By implementing different hardware configuration and test platforms, the results in terms of packet handling capacity, i.e., the number of packets received, packets analysed and packets dropped, will be collected. A list of hardware in all testbeds is summarised in Table 1 and a typical testbed for evaluating the performance of the S-NIDS is illustrated in Figure 2, where 1 Gbps of traffic is generated using a typical machine with 1 Gbps NIC as an attacker. (Due to limited hardware resources, we only generated traffic of 1 Gbps. However, this work could be readily extended to a network having higher traffic.) Different OSs and MCPs with a buffer size of 128 MB are sequentially employed to evaluate the effectiveness of each configuration.

Table 1. Hardware description for all testbeds.

Machine Type	Hardware Description
Network traffic generator/Attacking machines: . Windows 7 (64 bit) . Linux (Fedora 3.5.3-1.fc17.i686)	Dell Optiplex 7010, Intel Core (TM) i5/i7 CPU, 16GB RAM, 1Gbps Full Duplex NIC (intel)
S-NIDS machines: . Windows 7 (64 bit) . Windows Server 2012 (64 bit) . Linux (Fedora 3.5.3-1.fc17.i686)	
Switch	Cisco Catalyst 3650 Series, Gbps Switch with 24x1 Gbps ports
Router	Cisco 1941 Series

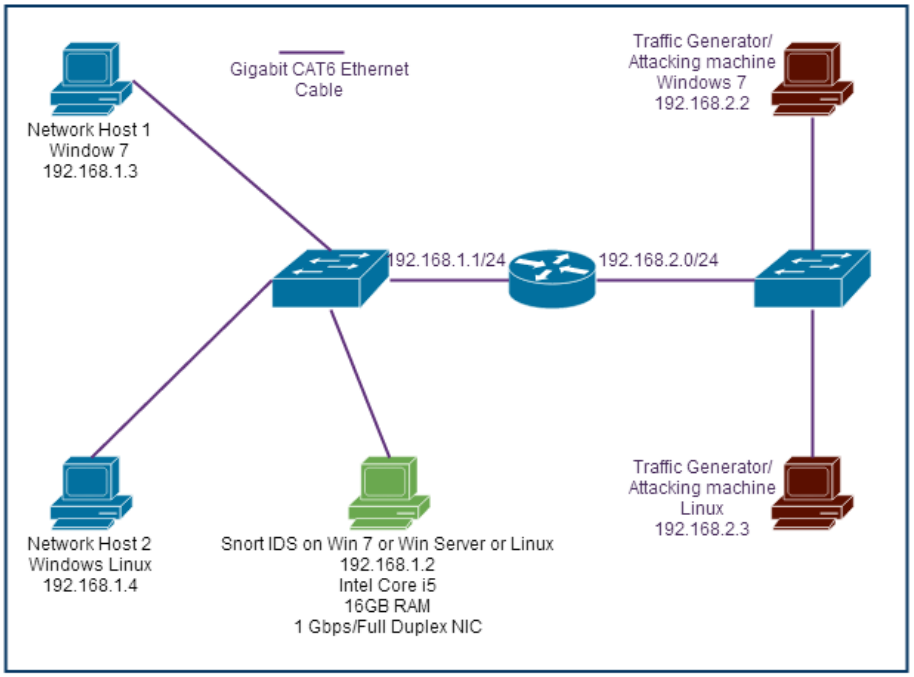

Figure 2. A typical testbed for S-NIDS evaluation.

Relating to various practical network scenarios, in this paper, let us consider the following four testbeds based on the setup in Figure 2:

(i) Testbed 1 *(Heavy traffic)*: 50,000 UDP, TCP and ICMP packets, each of which has a packet size of 128 bytes, are generated at 50 ms intervals.
(ii) Testbed 2 *(High data speed)*: 10,000 UDP, TCP and ICMP packets, each of which has a packet size of 128 bytes, are pushed into the network at 1 ms intervals.
(iii) Testbed 3 *(Large packet size)*: 10,000 UDP, TCP and ICMP packets, each of which has a packet size of 3072 bytes, are generated at 1 ms intervals.
(iv) Testbed 4 *(Heavy traffic and high data speed)*: A combination of the above testbeds 1 and 2, i.e., 50,000 UDP, TCP and ICMP packets, each of which has a packet size of 128 bytes, are generated and pushed into the network at 1 ms intervals.

The network specifications for the Testbeds 1–4 are summarised in Table 2.

Table 2. Network specification for Testbeds 1–4.

Machine	IP Address	Functionality
PC1	192.168.1.2/24	S-NIDS on Windows 7/Windows Server/Linux
PC2	192.168.2.2/24	Traffic generator/Attacking machine on Windows 7
PC3	192.168.2.3/24	Traffic generator/Attacking machine on Linux
PC4	192.168.1.3/24	Receiving host
PC5	192.168.1.4/24	Receiving host
Switch 1	N/A	Connecting PC1, PC4 and PC5
Switch 2	N/A	Connecting PC2 and PC3
Router	ge 0/0: 192.168.1.1 ge 0/1: 192.168.2.1	Connecting internal and external networks

4. Proposed Centralised Parallel Snort NIDS (CPS-NIDS)

In this section, let us consider an experiment in an enterprise network consisting of multiple VLANs. In order to cope with a huge volume of packets being analysed and to help decrease the packets dropped, we propose a centralised parallel S-NIDS (CPS-NIDS) in which each subnet has its own S-NIDS based on a specific OS to detect and log all incoming and outgoing packets specifically for that subnet. The main idea behind implementing CPS-NIDS is to split the incoming traffic into different VLAN groups. (Note that the bottleneck could take place at the central point in the CPS-NIDS. However, this is beyond the scope of our work in which we assume that the central receiving point can perfectly deal with all incoming traffic with no bottlenecks.) Therefore, the proposed system can eliminate a large number of incoming packets coming at the same time, which results in a better performance in terms of a reduced number of packets dropped. In addition, a centralised database server is introduced to deal with a huge number of log files and storage. (In order to configure the database, XAMPP (https://www.apachefriends.org), a completely free and easy to install Apache distribution containing MySQL, PHP and Perl can be used.) The database server can receive real-time log files from different VLANs that would help a Network Analyst have the entire log file of the whole network rather than having to physically access every VLAN to collect the log files, which is time-consuming. By employing the storage and analysis in a centralised manner, we will be able to keep track and detect all attack behaviours. These are helpful in amending or proposing appropriate policies that can be adaptively applied to each VLAN.

With the proposed CPS-NIDS, a large number of incoming packets coming at the same time can be considerably eliminated, which helps reduce the number of packets dropped. Along with a centralised database server, real-time log files from different VLANs can be easily accessed, which allows the detection of attack behaviours. However, it is worth noting that the bottleneck could take place at the central receiving point with a certain amount of packets lost and would cause a reduction in the number of packets received to be analysed.

Figure 3 illustrates the testbed, namely testbed 5, in order to validate the performance of CPS-NIDS with the network specifications listed in Table 3. In this network deployment, when a large amount of traffic is coming through the Router/Firewall, the Firewall/Router's access control list (ACL) can stop some of the traffic as unauthorised or known attackers. After the ACL reduces some of the traffic, the router sends the traffic to the targeted network. Note that our work is different from those in [18,19], where we now take a further step to investigate the performance of the CPS-NIDS in a practical high-speed and heavy-traffic network with large packet sizes using various MCPs and OPs.

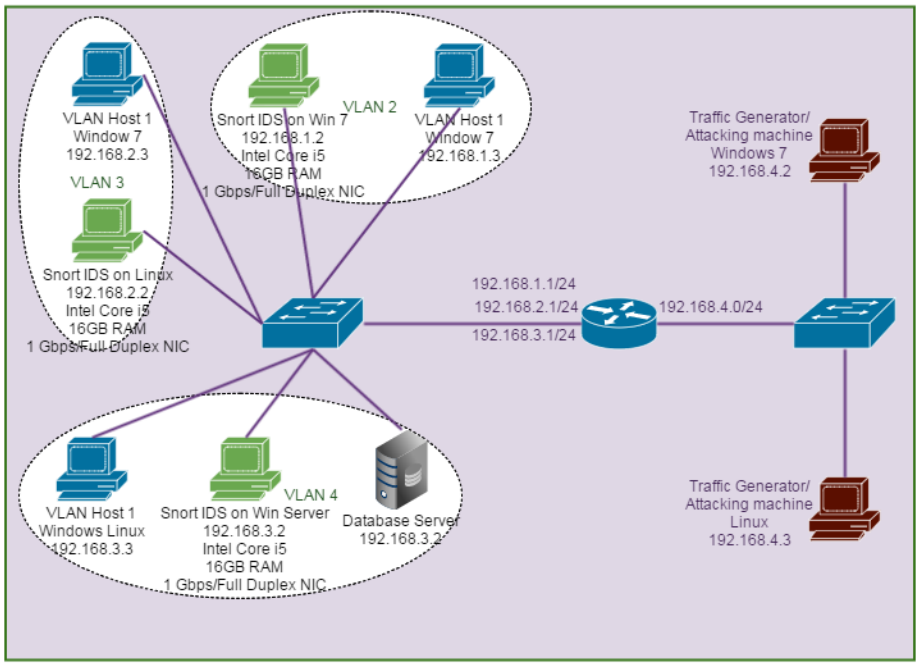

Figure 3. Proposed testbed for Centralised Parallel S-NIDS evaluation.

Table 3. Network specification for Testbed 5.

Machine	IP Address	Functionality
PC1	192.168.1.2/24	S-NIDS on Windows 7
PC2	192.168.3.2/24	S-NIDS on Windows Server 2012 & Database server (same machine)
PC3	192.168.2.2/24	S-NIDS on Linux
PC4	192.168.4.2/24	Traffic generator/Attacking machine on Windows 7
PC5	192.168.4.3/24	Traffic generator/Attacking machine on Linux
PC6	192.168.1.3/24	Receiving host
PC7	192.168.2.3/24	Receiving host
PC8	192.168.3.3/24	Receiving host
Switch 1	N/A	Connecting VLAN 2, VLAN 3 and VLAN 4
Switch 2	N/A	Connecting PC4 and PC5
Router	ge 0/0: 192.168.1.1, 192.168.2.1, 192.168.3.1 ge 0/1: 192.168.4.1	Connecting internal and external networks

5. Experimental Results and Evaluation

In this experiment, three types of packets are considered, including TCP, UDP and ICMP packets. The percentage number of the packets received, the packets analysed and the packets dropped are calculated by taking the average of the three packet types.

5.1. Performance Metrics

Besides the analysis of the number of packets received, packets analysed and packets dropped in S-NIDS, let us introduce a new metric, namely packet analysis efficiency (PAE), which is defined as the ratio of the number of packets analysed and the number of packets received. This new metric helps in providing a fair comparison between various network configurations with respect to various

testbeds. (Note that the packets sent but not received for analysis can be found from the PAE given the number of packets received and the total number of packets sent (i.e., 50,000 packets for heavy traffic models and 10,000 packets for other models).)

Let $N_{X,Y}^{(R)}$, $N_{X,Y}^{(A)}$ and $N_{X,Y}^{(D)}$, $X \in \{\text{Windows 7, Windows Server 2012, Linux}\}$, $Y \in \{\text{i7, i5}\}$, denote the number of packets received, the number of packet analysed and the number of packets dropped, respectively, in the S-NIDS using OS X and MCP Y. In addition, letting $\eta_{X,Y}$ denote the PAE of the S-NIDS, we have

$$\eta_{X,Y} = \frac{N_{X,Y}^{(A)}}{N_{X,Y}^{(R)}}. \tag{1}$$

In this work, we aim at finding an optimal network configuration to minimise the number of packets dropped, i.e.,

$$\min_{X,Y} \frac{N_{X,Y}^{(D)}}{N_{X,Y}^{(R)}}. \tag{2}$$

Since $N_{X,Y}^{(R)} = N_{X,Y}^{(A)} + N_{X,Y}^{(D)}$, we can rewrite Equation (2) as

$$\min_{X,Y} \left(1 - \frac{N_{X,Y}^{(A)}}{N_{X,Y}^{(R)}}\right). \tag{3}$$

Therefore, the optimisation problem in Equation (2) is equivalent to maximising the PAE. That is,

$$\max_{X,Y} \eta_{X,Y}. \tag{4}$$

5.2. Impact of OSs on SNIDS Performance

Table 4 shows the experimental results of five testbeds under investigation with respect to different OSs, including Windows 7, Windows Server 2012 and Linux. The network parameters and specifications are set as stated in Sections 3 and 4. The average number of the packets received, the packets analysed and the packets dropped in various testbeds are plotted in Figures 4–6, respectively. It can be observed that the proposed CPS-NIDS in testbed 5 achieves the highest performance. In addition, the Linux OS is shown to be able to receive and analyse the highest number of packets with the lowest number of packets dropped in all testbeds compared to Windows 7 and Windows Server.

Table 4. Experimental results of all testbeds w.r.t. various operating systems.

Testbed	OS	Packets Received	Packets Analysed	Packets Dropped
Testbed 1 (Heavy Traffic)	Windows 7	91.67%	67.00%	24.67%
	Windows Server	87.67%	63.33%	24.33%
	Linux	93.67%	75.00%	18.67%
Testbed 2 (High Speed)	Windows 7	86.00%	54.00%	32.00%
	Windows Server	88.00%	55.67%	32.33%
	Linux	90.33%	62.00%	28.33%
Testbed 3 (Large Packets)	Windows 7	80.67%	44.33%	36.33%
	Windows Server	80.67%	46.00%	34.67%
	Linux	85.67%	54.33%	31.33%
Testbed 4 (Heavy Traffic & High Speed)	Windows 7	73.00%	36.67%	36.33%
	Windows Server	73.33%	38.33%	35.00%
	Linux	74.00%	40.00%	34.00%
Testbed 5 (Centralised Parallel S-NIDS)	Windows 7	94.33%	81.67%	12.67%
	Windows Server	94.00%	83.00%	11.00%
	Linux	98.00%	89.33%	8.67%

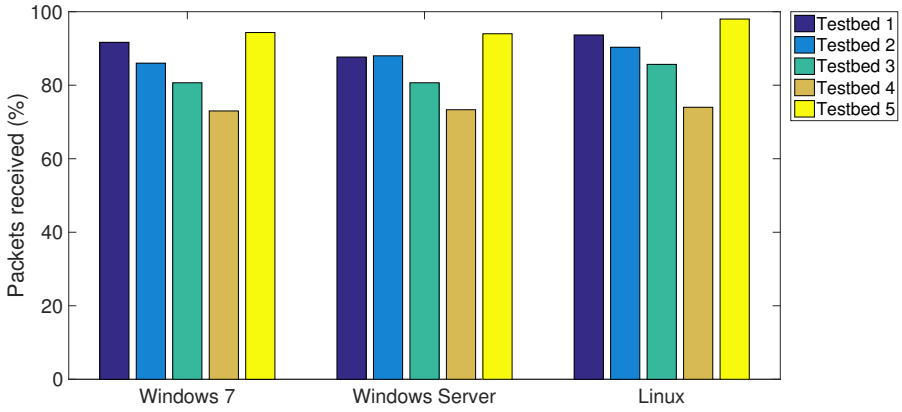

Figure 4. Comparative results of the packets received using different operating systems w.r.t. different testbeds.

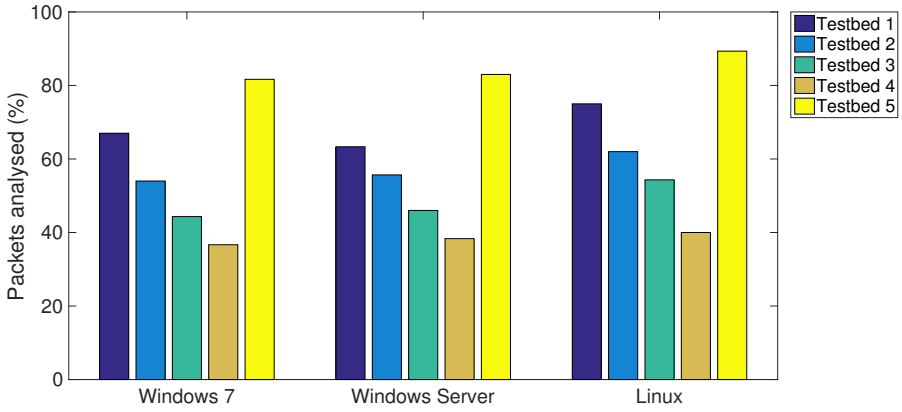

Figure 5. Comparative results of the packets analysed using different OSs w.r.t. different testbeds.

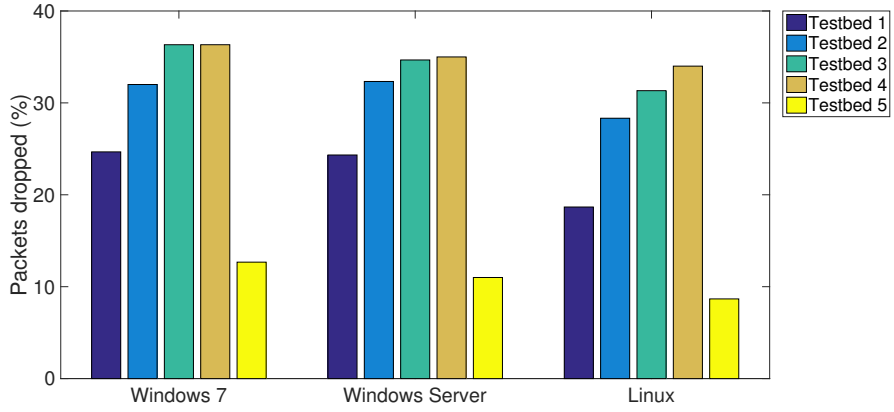

Figure 6. Comparative results of the packets dropped using different OSs w.r.t. different testbeds.

For a fair comparison, Figure 7 plots the PAE of all five testbeds with respect to various OSs. An improved performance of up to 10% can be achieved with Linux OS, while the Windows 7 is

shown to have the lowest performance. In particular, testbed 5 is shown to achieve a highest PAE over all other testbeds, which accordingly verifies the effectiveness of the proposed CPS-NIDS in intrusion detection in multiple VLANs with a low false positive, less processing and, most importantly, a very low number of packets dropped.

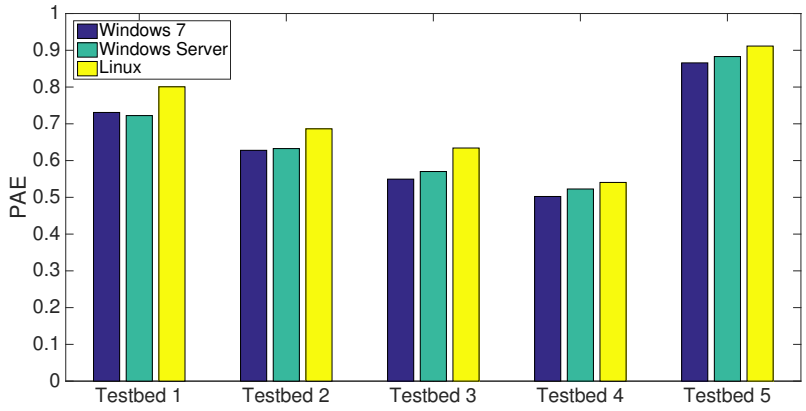

Figure 7. Packet analysis efficiency of S-NIDS in different OSs w.r.t. different testbeds.

5.3. Impact of MCPs on S-NIDS Performance

Investigating the impact of MCPs on the performance of S-NIDS in various testbeds, Table 5 summarises the experimental results of five testbeds using either i5 or i7 MCPs. Similarly, the network parameters and specifications of the testbeds are set as in Sections 3 and 4. Figures 8–10 sequentially plot the average number of the packets received, the packets analysed and the packets dropped in five testbeds with different MCPs. It can be seen that the proposed CPS-NIDS in testbed 5 achieves the highest performance with any types of MCPs. In addition, in these figures, a higher S-NIDS performance is achieved with the i7 compared to that of the i5, which can be easily verified due to the higher efficiency of the i7 processor with the latest technology.

Table 5. Experimental results of all testbeds w.r.t. various multi-core processors.

Testbed	MCP	Packets Received	Packets Analysed	Packets Dropped
Testbed 1	i7	94.33%	76.33%	18.00%
(Heavy Traffic)	i5	91.67%	67.00%	24.67%
Testbed 2	i7	87.33%	59.67%	27.67%
(High Speed)	i5	86.00%	54.00%	32.00%
Testbed 3	i7	85.33%	50.33%	35.00%
(Large Packets)	i5	80.67%	44.33%	36.33%
Testbed 4	i7	75.33%	43.00%	32.33%
(Heavy Traffic+High Speed)	i5	73.00%	36.67%	36.33%
Testbed 5	i7	98.67%	91.00%	7.67%
(CPS-NIDS)	i5	94.33%	81.67%	12.67%

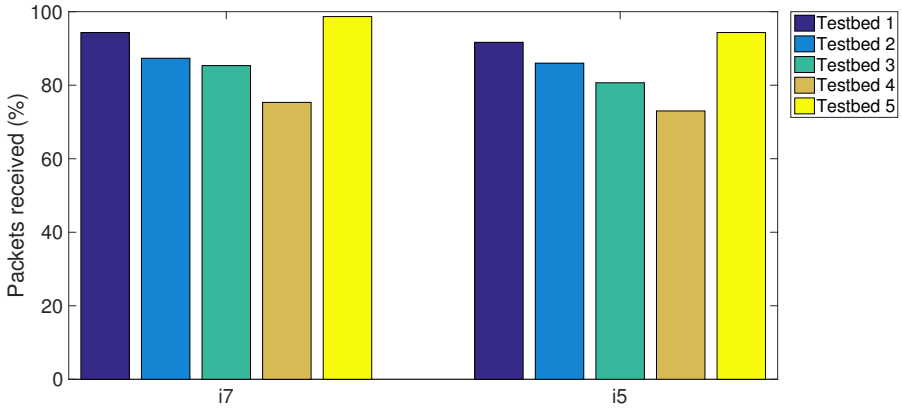

Figure 8. Comparative results of the packets received using different multi-core processors w.r.t. different testbeds.

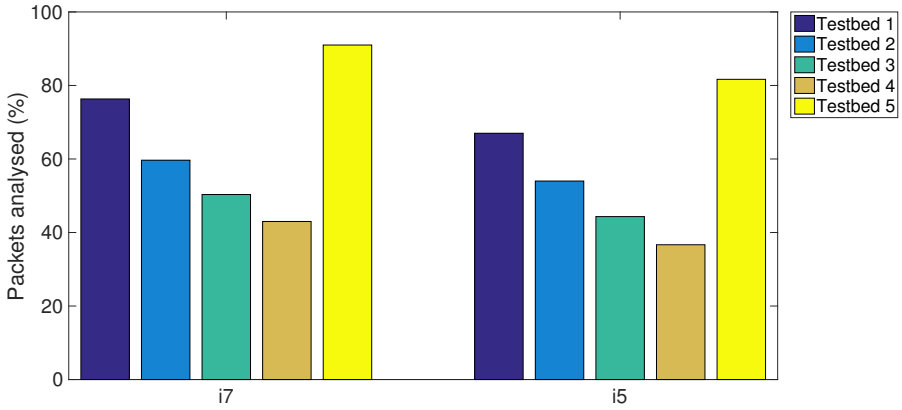

Figure 9. Comparative results of the packets analysed using different MCPs w.r.t. different testbeds.

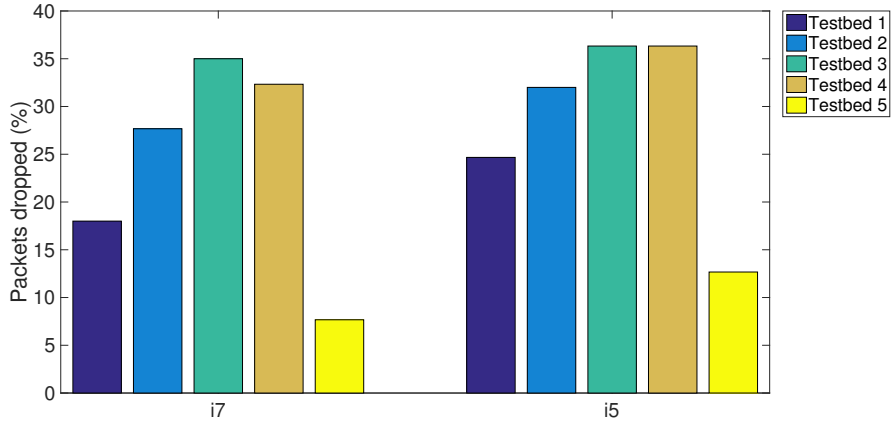

Figure 10. Comparative results of the packets dropped using different MCPs w.r.t. different testbeds.

Considering the fairness in comparison of various configurations, Figure 11 plots the PAE of all five testbeds with respect to two MCPs (i.e., i5 and i7). An improved performance of up to 8% is

shown to be achieved with the i7 processor in all testbeds. Again, testbed 5 is shown to achieve the highest PAE over all other testbeds, which reflects the effectiveness of the proposed CPS-NIDS in any network configuration.

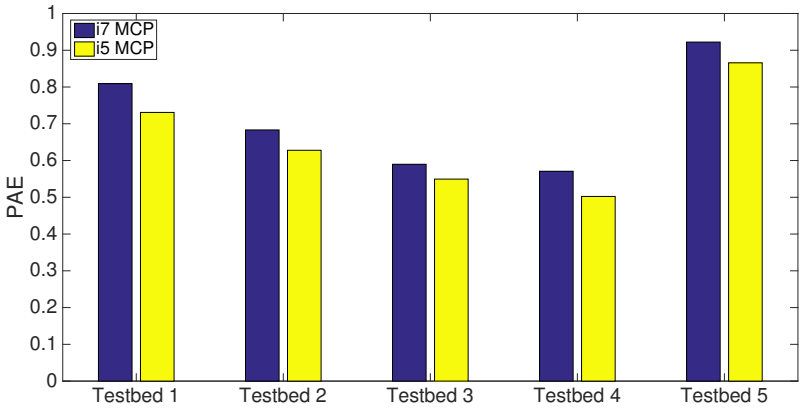

Figure 11. PAE of S-NIDS using different MCPs w.r.t. different testbeds.

6. Conclusions

In this paper, we have proposed a comparative experimental design to evaluate the performance of an open-source S-NIDS taking into account practical computer networks deploying the latest technology. Various testbeds have been built to reflect the practical network scenarios including high data speed and/or heavy traffic and large packet size. Furthermore, we have proposed CPS-NIDS to deal with high data speed and heavy traffic by employing, in parallel, S-NIDS in multiple VLANs. The proposed system not only enhances the performance in terms of a reduced number of packets dropped but also facilitates the network security management to access and keep records of all attack behaviours. By analysing the PAE, it has been shown that an improved performance of up to 10% can be achieved with Linux OS over other OSs, while an i7 MCP obtains up to 8% of improved performance over an i5 MCP. For future work, we will investigate the performance of the proposed CPS-NIDS, taking into account the effects of the bottleneck with packets lost at the central server, as well as the optimisation of the traffic splitting at the central server for VLAN groups with respect to various configurations of MCPs and OSs.

Author Contributions: Imdadul Karim and Quoc-Tuan Vien conceived and designed the experiments; Imdadul Karim performed the experiments, collected and analyzed data; Quoc-Tuan Vien designed the mathematical model, proposed the method, and summarized the data; Tuan Anh Le and Glenford Mapp contributed tools and wrote part of the paper; Imdadul Karim and Quoc-Tuan Vien wrote major part of the paper. All authors have read and approved the final manuscript.

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

References

1. Deka, R.K.; Kalita, K.P.; Bhattacharya, D.; Kalita, J.K. Network defense: Approaches, methods and techniques. *J. Netw. Comput. Appl.* **2015**, *57*, 71–84.
2. Alserhani, F.; Akhlaq, M.; Awan, I.; Mellor, J.; Cullen, A.; Mirchandani, P. Evaluating Intrusion Detection Systems in High Speed Networks. In Proceedings of the 5th International Conference on Information Assurance and Security, IAS'09, Xi'an, China, 18–20 August 2009; Volume 2, pp. 454–459.
3. Elliott, C. Botnets: To what extent are they a threat to information security? *Inf. Secur. Tech. Rep.* **2010**, *15*, 79–103.
4. Trestian, R.; Shah, P.; Nguyen, H.X.; Vien, Q.-T.; Gemikonakli, O.; Barn, B. Towards connecting people, locations and real-world events in a cellular network. *Telemat. Inform.* **2017**, *34*, 244–271.
5. Golbeck, J. *Analyzing the Social Web*; Morgan Kaufmann: Boston, MA, USA, 2013.

6. Mansfield-Devine, S. Anti-social networking: Exploiting the trusting environment of Web 2.0. *Netw. Secur.* **2008**, *2008*, 4–7.
7. Srivastava, A.; Gupta, B.; Tyagi, A.; Sharma, A.; Mishra, A. A Recent Survey on DDoS Attacks and Defense Mechanisms. In *Advances in Parallel Distributed Computing*; Nagamalai, D., Renault, E., Dhanuskodi, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 203, pp. 570–580.
8. Matthews, T. *Incapsula Survey: What DDoS Attacks Really Cost Businesses*; Incapsula, Inc.: Redwood Shores, CA, USA, 2014.
9. Pescatore, J. *DDoS Attacks Advancing and Enduring: A SANS Survey*; Corero Network Security, Inc.: Marlborough, MA, USA, 2014.
10. Beale, F.J.; Team, M.o.t.S.; Baker, A.R.; Esler, J. *Snort Intrusion Detection and Prevention Toolkit*; Syngress: Rockland, MA, USA, 2007.
11. Liao, H.J.; Lin, C.H.R.; Lin, Y.C.; Tung, K.Y. Intrusion detection system: A comprehensive review. *J. Netw. Comput. Appl.* **2013**, *36*, 16–24.
12. Salah, K.; Kahtani, A. Performance Evaluation Comparison of Snort NIDS Under Linux and Windows Server. *J. Netw. Comput. Appl.* **2010**, *33*, 6–15.
13. Lin, P.C.; Lee, J.H. Re-examining the Performance Bottleneck in a NIDS with Detailed Profiling. *J. Netw. Comput. Appl.* **2013**, *36*, 768–780.
14. Guillen, E.; Padilla, D.; Colorado, Y. Weaknesses and strengths analysis over network-based intrusion detection and prevention systems. In Proceedings of the IEEE Latin-American Conference on Communications, LATINCOM'09, Medellin, Colombia, 10–11 September 2009; pp. 1–5.
15. Schaelicke, L.; Slabach, T.; Moore, B.; Freeland, C. Characterizing the Performance of Network Intrusion Detection Sensors. In *Recent Advances in Intrusion Detection*; Vigna, G., Kruegel, C., Jonsson, E., Eds.; Springer: Berlin/Heidelberg, Germany, 2003; Volume 2820, pp. 155–172.
16. Hoque, N.; Bhuyan, M.H.; Baishya, R.; Bhattacharyya, D.; Kalita, J. Network attacks: Taxonomy, tools and systems. *J. Netw. Comput. Appl.* **2014**, *40*, 307–324.
17. Inayat, Z.; Gani, A.; Anuar, N.B.; Khan, M.K.; Anwar, S. Intrusion response systems: Foundations, design, and challenges. *J. Netw. Comput. Appl.* **2016**, *62*, 53–74.
18. Bulajoul, W.; James, A.; Pannu, M. Improving Network Intrusion Detection System Performance Through Quality of Service Configuration and Parallel Technology. *J. Comput. Syst. Sci.* **2015**, *81*, 981–999.
19. Bulajoul, W.; James, A.; Pannu, M. Network Intrusion Detection Systems in High-Speed Traffic in Computer Networks. In Proceedings of the IEEE 10th International Conference on e-Business Engineering, ICEBE 2013, Coventry, UK, 11–13 September 2013; pp. 168–175.
20. Hall, M.; Wiley, K. Capacity Verification for High Speed Network Intrusion Detection Systems. In *Recent Advances in Intrusion Detection*; Wespi, A., Vigna, G., Deri, L., Eds.; Springer: Berlin/Heidelberg, Germany, 2002; Volume 2516, pp. 239–251.
21. Meng, Y.; Kwok, L.F. Adaptive blacklist-based packet filter with a statistic-based approach in network intrusion detection. *J. Netw. Comput. Appl.* **2014**, *39*, 83–92.
22. Kim, I.; Oh, D.; Yoon, M.; Yi, K.; Ro, W. A distributed signature detection method for detecting intrusions in sensor systems. *Sensors* **2013**, *13*, 3998–4016.
23. Aho, A.V.; Corasick, M.J. Efficient String Matching: An Aid to Bibliographic Search. *ACM Commun.* **1975**, *18*, 333–340.
24. Jamshed, M.A.; Lee, J.; Moon, S.; Yun, I.; Kim, D.; Lee, S.; Yi, Y.; Park, K. Kargus: A Highly-Scalable Software-Based Intrusion Detection System. In Proceedings of the 2012 ACM Conference on Computer and Communications Security, CCS 2012, Raleigh, NC, USA, 16–18 October 2012; pp. 317–328.
25. Jiang, W.; Song, H.; Dai, Y. Real-time intrusion detection for high-speed networks. *Comput. Secur.* **2005**, *24*, 287–294.
26. Shiri, F.; Shanmugam, B.; Idris, N. A parallel technique for improving the performance of signature-based network intrusion detection system. In Proceedings of the IEEE 3rd International Conference on Communication Software and Networks, ICCSN 2011, Xi'an, China, 27–29 May 2011; pp. 692–696.
27. Wheeler, P.; Fulp, E. A Taxonomy of Parallel Techniques for Intrusion Detection. In Proceedings of the 45th Annual Southeast Regional Conference, ACM-SE 2007, Winston-Salem, NC, USA, 23–24 March 2007; pp. 278–282.

28. Vasiliadis, G.; Polychronakis, M.; Ioannidis, S. MIDeA: A Multi-Parallel Intrusion Detection Architecture. In Proceedings of the 18th ACM Conference on Computer and Communications Security, CCS 2011, Chicago, IL, USA, 17–21 October 2011; pp. 297–308.
29. Varghese, G.; Fingerhut, J.A.; Bonomi, F. Detecting Evasion Attacks at High Speeds Without Reassembly. In Proceedings of the 2006 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications, SIGCOMM 2006, Pisa, Italy, 11–15 September 2006; pp. 327–338.
30. Lee, K.; Kim, Y.; Hong, S.; Kim, J. PolyI-D: Polymorphic Worm Detection Based on Instruction Distribution. In *Information Security Applications*; Lee, J., Yi, O., Yung, M., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; Volume 4298, pp. 45–59.
31. Wang, Y.; Xiang, Y.; Zhou, W.; Yu, S. Generating regular expression signatures for network traffic classification in trusted network management. *J. Netw. Comput. Appl.* **2012**, *35*, 992–1000.