Received 21 April 2024, accepted 23 April 2024, date of publication 29 April 2024, date of current version 7 May 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3395123

RESEARCH ARTICLE

Advancing Snort IPS to Achieve Line Rate Traffic Processing for Effective Network Security Monitoring

MUHAMMAD SHEERAZ[1], MUHAMMAD HANIF DURAD[1], SHAHZAIB TAHIR[2], (Senior Member, IEEE), HASAN TAHIR[3], (Senior Member, IEEE), SAQIB SAEED[4], AND ABDULLAH M. ALMUHAIDEB[5]

[1]Department of Computer and Information Sciences, Pakistan Institute of Engineering and Applied Sciences, Islamabad 45650, Pakistan
[2]Department of Information Security, College of Signals, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[3]Department of Information Security, School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Islamabad 44000, Pakistan
[4]Saudi Aramco Cybersecurity Chair, Department of Computer Information Systems, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia
[5]Saudi Aramco Cybersecurity Chair, Department of Networks and Communications, College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Dammam 31441, Saudi Arabia

Corresponding author: Abdullah M. Almuhaideb (amalmuhaideb@iau.edu.sa)

This work was supported by the Saudi Aramco Cybersecurity Chair, Imam Abdulrahman Bin Faisal University.

ABSTRACT Intrusion Prevention Systems (IPS), capable of preventing the organizational network from a cyber-attack in addition to detecting it, are widely adopted by organizations to protect their networks from unauthorized access, attacks, and malicious activities. Similarly, Snort an open-source IPS is extensively used for effective network security monitoring and analysis. When functioning as an IPS, Snort can be deployed in inline mode within an organizational network, so that all the organizational network traffic travels through it, hence actively blocking or preventing malicious traffic in real-time. This requires Snort to process the network traffic fast enough to match the network traffic line rate. But the Snort IPS default data acquisition module i.e. advanced packet filtering (AF_PACKET) cannot process network traffic at the line rate that causes packet loss and network services disturbance. This research work discusses the technologies available to make Snort IPS process network traffic at line rate. Packet filtering framework (PF_RING) and data plane development kit (DPDK) are the most effective and widely used software technologies, whereas the Napatech smart network interface card (smartNIC) is a very efficient hardware technology for achieving line rate traffic processing. A throughput comparison shows that PF_RING and DPDK achieve a throughput close to 1G with 100% CPU utilization whereas Napatech smartNIC achieves full 1G throughput with CPU utilization of less than 5%. Furthermore, the integration of Snort IPS with the security information and event management (SIEM) system has been discussed for better attack detection in an organizational network.

INDEX TERMS AF_PACKET, DPDK, intrusion detection system, intrusion prevention system, IXIA BreakingPoint system, Napatech smartNIC, PF_RING, SIEM.

I. INTRODUCTION

The advent of computer networks, especially the Internet has enabled people to connect and share their resources irrespective of long distances. With the development of cloud, IoT, and big data technologies, Internet usage has increased by a great deal. All these Internet-based technologies have augmented the amount of network traffic to a colossal level [1]. Web applications are very common and easy to access for users. People are using the Internet for activities like online shopping, business transactions, news, the stock market, etc. The augmented usage of the Internet by the end users has

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

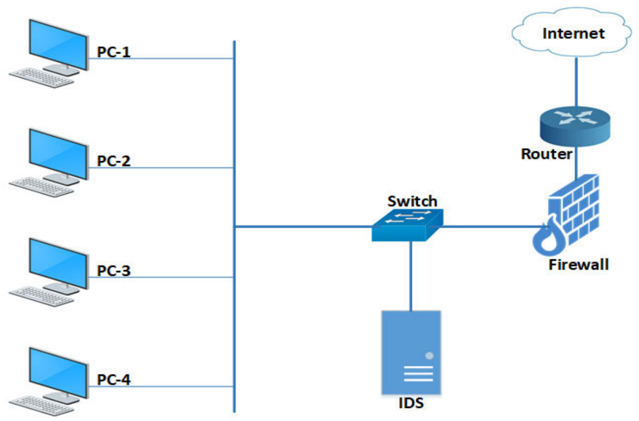


FIGURE 1. An intrusion detection system (IDS).

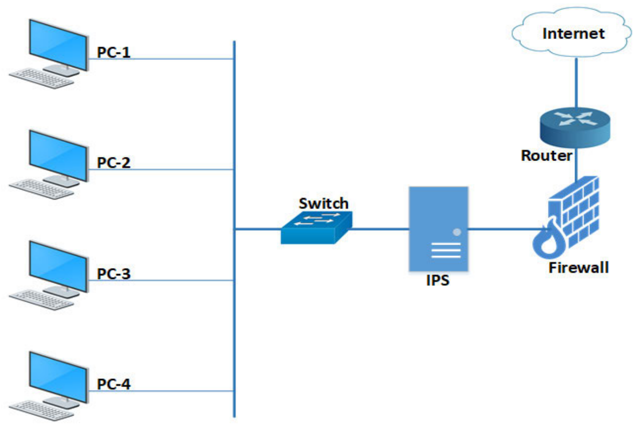


FIGURE 2. An intrusion prevention system (IPS).

provided a broader attack surface to the adversaries. They carefully pick their victims who are both individuals and organizational networks. They exploit the weaknesses and vulnerabilities present in the applications and systems [2].

Various types of cyber-attacks are common and also increasing day by day such as denial of service (DoS) attacks, SQL injection (SQLi) attacks, cross-site scripting (XSS) attacks, ransomware attacks, cyber vandalism, cyber espionage, etc. [2], [3], [4], [5]. Therefore, several security systems have been developed and deployed to safeguard the IT infrastructure against such attacks. There isn't one security system that can guarantee to protect the entire network. Organizations have to use several security systems for the protection of the network and organize them in a layered order. An intrusion detection system (IDS) [6], [7], [8] is one of the security systems that is used to shield the IT infrastructure of organizations. The IDS is called the second layer of protection in a network. The IDS along with a firewall is used to safeguard critical assets of a network [9]. The typical deployment of an IDS in a network is shown in Figure 1.

An IDS system inspects the network traffic at a deeper level than the firewall. An IDS system can be deployed as a host-based IDS (HIDS) or as a network-based IDS (NIDS). When an IDS system is configured in such a way that besides detecting intrusions it can prevent or block specific network traffic then it is called an intrusion prevention system (IPS). The typical deployment of an IPS in a network is shown in Figure 2.

Snort is an open-source IDS/IPS system that is considered a de-facto standard for intrusion detection systems. It is a lightweight and signature-based IDS/IPS system [10]. It is the most widely used IDS/IPS system for the detection and prevention of intrusions. It has been developed based on a single-threaded application model. Its simple, easy, and fast deployment method is one of its conspicuous features. In network security, an IPS system is preferred over an IDS system because an IPS system provides some additional features that an IDS system lacks such as its capability to block malicious network traffic along with its detection [11]. Snort can be configured and deployed both as an IDS as well as an IPS. When Snort is deployed as an IPS in the network, it receives the network traffic on one of its interfaces, inspects the network traffic, and then sends the traffic through the other interface. The Snort IPS system requires at least two physical interfaces to work in IPS mode. It is also called the inline mode of Snort. The inline mode of Snort is very critical because if any sort of problem occurs the whole network communication gets disturbed. In other words, it becomes a single point of failure in the network i.e. if it is unavailable due to any failure it will break the communication of a network with the Internet.

In the inline mode, Snort must be able to handle the network bandwidth. It must support the network traffic line rate for the smooth functioning of the network. This poses a challenge for the Snort IPS system. This is because the Snort IPS default data acquisition (DAQ) module for inline mode is advanced packet filtering (AF_PACKET) but it does not support line rate communication of network traffic. The DAQ module handles the process of receiving and sending the network traffic. However, there also exist some other DAQ modules that include Internet packet queue (IPQ), Internet protocol firewall (IPFW), and netfilter queue (NFQ). The IPQ was an older mechanism for handling iptable packets, and IPFW is used in BSD systems. The NFQ is a new and improved mechanism for handling iptable packets [12]. But none of these DAQ modules guarantee line rate traffic capturing. Therefore, to deploy Snort as an IPS some solution is required to enable Snort IPS to capture and send network traffic in real-time and at line rate.

The following contributions are made through this research:

- A Snort can function as an IDS and an IPS where when being used as an IDS, it focuses on detection and alerting, while as an IPS it takes proactive measures to prevent and block the threats. This research utilizes Snort in the inline mode aimed at enhancing the throughput of the Snort IPS. To achieve this, the presented research explores software-based and hardware-based solutions.
- This research presents a performance comparison of software and hardware packet-capturing mechanisms, along with other parameters important for throughput augmentation.

TABLE 1. Comparison of different throughput enhancement solutions.

S.No.	Research Work	Software Technology	Hardware Technology	Hardware Dependency	Min. Cores Requirement
1	[1]	DPDK, PF_RING, Netmap, Netfiler	×	✓	×
2	[16]	Multiple Snort instances	×	×	16
3	[17]	DPDK, Hyperscan	×	✓	×
4	[18]	DPDK	×	✓	×
5	[19]	Kernel parameters modification	×	×	×
6	[20]	HTTP Flow-based detection	×	×	×
7	Our Work	AF_PACKET, DPDK, PF_RING	Napatech smartNIC	depends *	×

* AF_PACKET and PF_RING don't have HW dependency whereas DPDK and Napatech smartNIC are HW dependent.

- This research sheds light on the significance of integrating Snort IPS with a security information and event management (SIEM) system, exploring various aspects of the integration.

The rest of the paper is organized as follows: Section II is mainly concerned with related work. Section III presents the available software and hardware technologies widely used for the throughput enhancement of Snort IPS. Section IV illustrates the experimental setup used in this research. Section V presents detailed results and discussion. Section VI provides a very useful discussion about integrating Snort IPS with a SIEM system. Section VII presents the conclusion and future work of our research.

II. RELATED WORK

Snort is an open-source IDS/IPS system that has widely been studied in the literature to boost its performance under heavy traffic. For this purpose, several frameworks have been proposed that rely on parallelization [13], [14], [15] as depicted in Table 1. In [16] an efficient intrusion detection system for the high-speed network is proposed by using open-source Snort. Multiple instances of Snort are run on the machine. Multiple network interface cards (NICs) are used. Each NIC captures and forwards network traffic to a corresponding Snort instance. Several Snort instances are being run depending on the number of cores present on the machine. 1 Gbps network speed is achieved by running Snort instances on 16 CPU cores. It is quite obvious that if Snort instances are run on a machine that does not have 16 CPU cores, 1 Gbps network speed cannot be achieved. Hence line rate network traffic cannot be achieved in such a scenario.

In [17] Snort IDS is optimized by using the data plane development kit (DPDK) and Hyperscan. DPDK is an Intel software solution (set of libraries and drivers) for achieving high data rate packet-capturing. Hyperscan is also an Intel software solution for fast pattern matching. Although the optimized Snort IDS shows better line rate packet-capturing and detection rate for malicious traffic under high-speed network traffic conditions, at the same time it also introduces some dependencies. Intel DPDK and Hyperscan technologies do not support every hardware platform. Therefore the proposed Snort IDS solution is not generic and it is hardware-dependent. Also, the proposed Snort IDS system only considers the DPDK solution and does not present a performance comparison against the packet filtering framework (PF_RING) solution. PF_RING is also an efficient software solution for fast packet-capturing.

In [18] traditional Snort IDS system limitations are explored and highlighted. Results show that Snort IDS in inline mode only supports up to 100 Mbps network traffic line rate. In its default configurations, Snort cannot be deployed on a 1 Gbps network in inline mode. Intel DPDK software framework is used to achieve a 1 Gbps network traffic line rate. In this case, as DPDK is used for fast packet-capturing, it is also hardware-dependent and is not supported on every NIC.

In [1] some of the efficient packet-capturing and processing technologies are discussed. Libpcap, Libpcap-mmap, Netmap, Netfilter, DPDK, and PF_RING technologies that are used for packet-capturing are analyzed. The technical characteristics and experimental results of these technologies are compared. It is shown that PF_RING and DPDK show the best results among all these technologies. However, details about the hardware platform used for the generation of results are lacking. The information about CPU and RAM is not given. Similarly, although 1G NIC is used its vendor and model have not been mentioned. This is essential because DPDK does not support every NIC.

A kernel parameters modification approach has been presented in [19]. Linux network application programming interface (NAPI) is a kernel-level packet-capturing mechanism available to be used by user applications. But in its default parameters configuration, this packet-capturing mechanism is well below the line rate and it does not provide an efficient packet-capturing mechanism. However, if some parameters like Budget B value are assigned values other than default the application i.e. Snort performs better. Snort performs best at Budget B value of 14 instead of its default value of 300.

The proposed architecture in [16] requires at least 16 CPU cores for its operation along with multiple network card interfaces to serve those CPU cores. Our research only requires one CPU core and two network card interfaces. The proposed Snort architectures in [17] and [18] are not generic and only limited to the supported hardware for DPDK and Hyperscan, unlike our research which also provides a solution for every hardware platform. In [1] the details about the hardware platform, CPU, RAM, NIC, etc. have not been provided and only software solutions have been explored, unlike our research that provides such details along with exploring the hardware solution also. In [19] only the kernel parameters modification approach has been discussed with no comparison of software and hardware technologies for the packet-capturing mechanism. In our research, we not only discuss various software and hardware packet-capturing technologies along with their comparison but also provide detailed information about the test environment used for the experimentation.

The inability of the Snort to handle line rate network traffic on various platforms has been highlighted in [10]. The technique of flow-based intrusion detection to improve the throughput of the Snort IDS has been proposed in [20] but it only applies to intrusion detection, not intrusion prevention. The performance analysis and evaluation of Snort and some other intrusion detection systems have been presented in [21], [22], and [23]. We also refer readers to [24], [25], and [26] that present a thorough survey of IDS/IPS in terms of the techniques, datasets, and challenges.

III. AVAILABLE SOLUTIONS

If Snort is configured and integrated properly, the aforementioned problems associated with the Snort's inline mode can be solved. These include both software as well as hardware solutions. This section highlights the available solutions:

A. SOFTWARE-BASED SOLUTIONS

Some of the software solutions have been discussed in the above section. Among them, DPDK and PF_RING have demonstrated exceptional performance in achieving line rate network traffic in Snort IPS's inline mode [11], [12]. Although Netmap is another such solution that is used for achieving line rate network traffic in Snort inline mode, it has a very complex deployment procedure and is not widely used for this purpose. Therefore the following software solutions are used most often to achieve line rate network traffic in the inline mode of Snort IPS.

1) AF_PACKET

The AF_PACKET library is the default packet-capturing library in the Snort IPS and it is installed by default while installing the Snort IPS. Therefore no extra effort is required for its installation and integration with the Snort IPS. The AF_PACKET library is also not hardware-dependent. But having said that, AF_PACKET does not provide a high line rate throughput in the inline mode. The AF_PACKET needs two network interfaces for its operation in IPS mode. AF_PACKET creates a bridge between the two network interfaces and copies packets from one interface to the other and vice versa [27]. A few parameters have to be set to run Snort in IPS or inline mode using AF_PACKET [12]. These parameters can be set using the command line or in the configuration file of Snort i.e. snort.conf. We set parameters in the configuration file as follows.

```
config daq: afpacket
config daq_dir: /usr/local/lib/daq
config daq_mode: inline
config daq_var: buffer_size_mb=1024
```

In the first line, we select afpacket as the daq module. Then we specify the directory that contains the installed daq module. The daq mode is inline and the buffer size is 1024 MB. In this case, we can run Snort with the following command.

```
./snort -c /etc/snort/etc/snort.conf -i eth0:eth1
```

With "-i" we specify the network interfaces pair. We do not need to specify the rest of the parameters because we have specified these parameters in the configuration file. If we do not specify these parameters in the configuration file we can also set these parameters through the command line as given below.

```
./snort -c /etc/snort/etc/snort.conf -i eth0:eth1 --
daq afpacket --daq-dir /usr/local/lib/daq -daq-mode
inline --daq-var buffer_size_mb=1024 -Q
```

The working of AF_PACKET is illustrated in Figure 3. When a network packet arrives on the network interface card, it is copied into the kernel space of the operating system. User applications can access this packet by calling the application programming interface (API) function from the user space, which copies this packet into the user space. Now user application has full control of this packet for use. Because of this long journey of the network packet from the network interface card to the user application, the throughput of the Snort IPS is reduced.

2) DATA PLANE DEVELOPMENT KIT (DPDK)

The DPDK is an open-source collection of software libraries and network interface controller polling mode drivers that take away the burden of packet capture from the operating system kernel. The DPDK processes packets efficiently as compared to the operating system kernel TCP stack. The DPDK was initially developed by Intel Corporation but from 2017 onwards it is managed by the Linux Foundation. The DPDK has been developed in C language and is available for Linux, Windows, and FreeBSD operating systems [28]. The DPDK provides its functionality for x86, ARM, and PowerPC hardware architectures [29].

The DPDK packet-capturing library installation and integration with the Snort IPS is slightly complex. It is available in source code form and it has to be compiled, installed, and integrated separately. However, lately, some Linux distributions also provide DPDK in packaged form. The DPDK

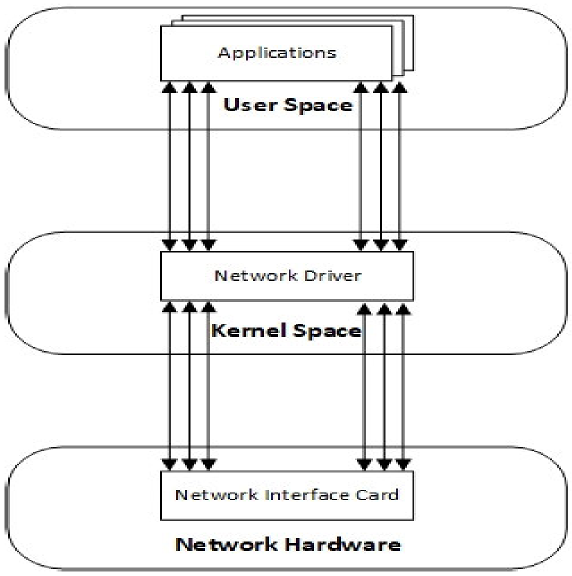


FIGURE 3. AF_PACKET working.

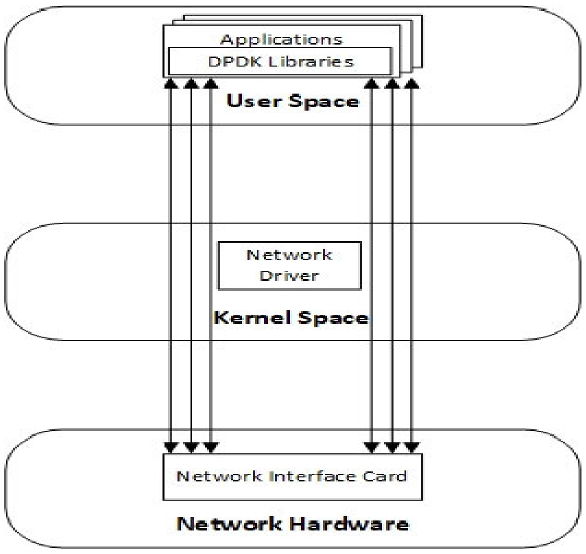


FIGURE 4. DPDK working.

library is hardware-dependent which means that it cannot be run on every network interface card. The list of DPDK-compatible network interface cards is available on the DPDK website [29].

When the DPDK library source code is compiled and installed, a DPDK kernel module is also installed. When the DPDK library is run, this kernel module is loaded to control the packet-capturing mechanism. If we have two network interfaces like eth0 and eth1, when Snort IPS is run in inline mode on these two interfaces, these interfaces no longer become available for the operating system. These interfaces also do not show up with the "ifconfig" command in Linux. This is because these interfaces are run with modified DPDK drivers that are not under the control of a normal kernel module. When we stop running Snort IPS, these interfaces can be reconnected to the standard operating system drivers. After that, these interfaces reappear with the "ifconfig" system command.

The DPDK controls network packet-capturing in user space. It does this at a much higher speed than traditional operating system kernel TCP stack and also saves precious CPU cycles for other tasks. The Snort version 2.9 and onwards at its core does not call the libpcap library interface directly, rather it calls the DAQ module interface which in turn can call DPDK, AF_PACKET, PF_RING, or any other packet-capturing library, adding an abstraction layer [17]. This has eased the process of compiling and integrating a new packet-capturing library with the Snort. Earlier, if the DPDK library was to be integrated with the Snort, along with the DPDK library Snort was also required to be recompiled and installed. Similarly, if any other packet-capturing library was to be integrated with Snort, Snort was also required to be recompiled and installed. However, since Snort version 2.9, the data acquisition module has been separated from Snort. Now if the DPDK library is to be integrated with Snort, only the DPDK library is required to be compiled and installed not the Snort. The working of DPDK is illustrated in Figure 4. The command to run Snort with DPDK is given below.

```
./snort -c /etc/snort/etc/snort.conf --daq dpdk --
daq-dir /usr/local/lib/daq --daq-var dpdk_args=”-c
40” --daq-mode inline -i dpdk0:dpdk1 -Q
```

The DPDK enables an application running in user space to capture packets directly from the NIC driver removing the need to copy packets from the NIC driver to the operating system kernel space and then from kernel to user space (normal functioning of operating system kernel TCP stack). The DPDK requires huge pages for its functioning to allocate a large memory pool and to reduce the number of lookups and page management [30]. On arriving at the network interface card, a network packet is picked up by the modified DPDK driver. The DPDK libraries are present in the user space. These DPDK libraries pick up the network packet directly from the network interface card through the modified network card driver, bypassing the kernel space of the operating system. This saves precious CPU time for other tasks. In this mechanism, the step of copying network packets in the kernel space is eliminated which improves the overall throughput of the Snort IPS.

3) PF_RING

The PF_RING is a set of software libraries and customized drivers for capturing packets at high speed using a zero-copy mechanism, proposed by L. Deri [1]. The PF_RING can convert a commodity PC into an efficient and cheap packet-capturing and manipulation machine. The PF_RING packet-capturing library has to be installed and integrated with the Snort IPS separately. It can be compiled and installed from the source or it can be installed by the rpm package provided for the Linux operating system i.e. CentOS-7 in our case. So PF_RING can be integrated with very little effort and

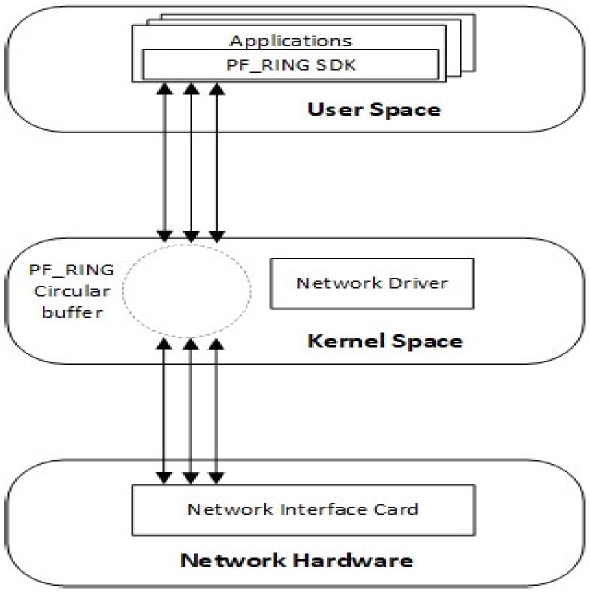


FIGURE 5. PF_RING vanilla mode working.

TABLE 2. Supported driver families.

Driver Family	Supported Cards
1 Gbit (e1000e, igb)	Intel 8254x, 8256x, 82571, 82572, 82573, 82574, 82583, 82575, 82576, 82580, I210, I350
10 Gbit (ixgbe/ixgbevf)	Intel 82599, X520, X540, X55x
10/40 Gbit (i40e)	Intel X710/XL710
10/25/50/100 Gbit (ice)	E810

has low integration complexity. The PF_RING library is also not dependent on hardware which means that it can be run on any network interface card (NIC).

Two modes or versions of PF_RING can be used in user applications according to the requirements [31]. The first mode of operation of PF_RING is Vanilla PF_RING as depicted in Figure 5. It consists of the following two components:

- An accelerated kernel module responsible for efficiently copying packets to PF_RING rings;
- A user-space PF_RING SDK that enables user-space applications to use PF_RING functionality transparently.

This mode does not use customized NIC drivers rather it works with standard NIC drivers. Therefore a few packets can get dropped in this mode but it still gives much better performance as compared to AF_PACKET in inline mode. But in scenarios where a 1G line rate must be achieved, the Vanilla mode of PF_RING is not recommended for usage.

The second mode of operation of PF_RING is PF_RING ZC (Zero Copy) which consists of the following components.

- A user-space PF_RING SDK;
- Customized NIC drivers.

In this mode, customized ZC drivers are used that guarantee no packet loss for high-speed networks. In this mode, both the Linux kernel and PF_RING kernel module are bypassed and packets are directly copied in a zero-copy fashion from ZC NIC drivers to PF_RING rings. Table 2 depicts the driver families that are currently supported with PF_RING [31].

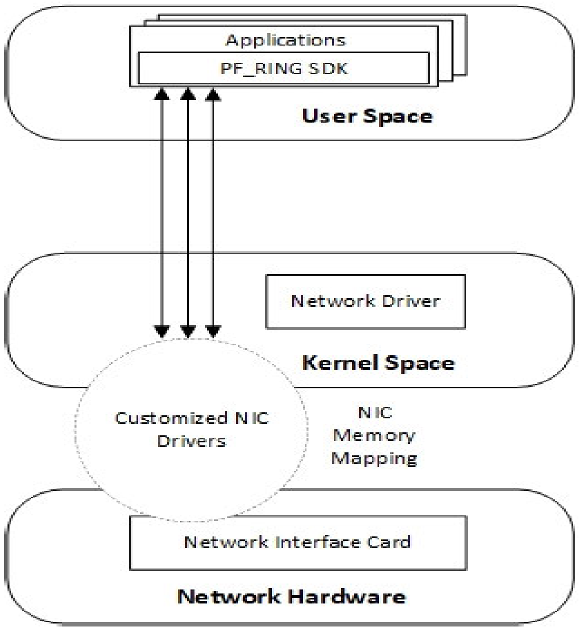


FIGURE 6. PF_RING ZC mode working.

The working of PF_RING ZC mode of operation is depicted in Figure 6. The vanilla version of the PF_RING library is free for use but the zero-copy version of PF_RING is not freely available. A license must be purchased for using the zero-copy version of the PF_RING library. However, the zero-copy version of PF_RING can be used for testing purposes. But it only supports the Snort IPS mode operation for a period of 5 minutes. The command to run Snort with PF_RING support is given below.

```
./snort -c /etc/snort/etc/snort.conf --daq pfring -
-daq-dir /usr/local/lib/daq --daq-mode inline -i
eth0:eth1 -Q
```

B. HARDWARE-BASED SOLUTIONS

Hardware solutions are generally better in performance as compared to corresponding software solutions. Typically, after a certain threshold level, the performance of a software solution cannot match the performance of a hardware solution. Therefore, it is often desirable to use a hardware solution for a specific problem rather than using a software solution. However, when opting for a hardware solution, it costs quite high compared to the software solution. Therefore although a hardware solution is better in terms of performance it also costs more. Hence not every end user opts for a hardware solution however big enterprises and high-tech companies certainly go for a hardware solution for high performance.

There are some hardware solutions available to be used in the inline mode of Snort IPS for achieving line rate network traffic capturing. Vendors developing network acceleration cards (NAC) for high-speed packet-capturing applications

TABLE 3. Napatech smart NICs.

	NT40A01-SCC	NT40A11	NT100A01-SCC	NT40E3-4	NT100A01-NEBS	NT40A01-SCC
Operating System	Linux, Windows	Linux, Windows	Linux, Windows	Linux, FreeBSD, Windows	Linux, Windows	Linux, Windows
Network Ports	4 SFP+	4 SFP+	4 SFP28	4 SFP+	4 SFP28	4 SFP+
Data Rate	4x1 Gbps	4x1/10 Gbps	4x1/10/25 Gbps	4x1/10 Gbps	4x1/10/25 Gbps	4x1 Gbps
No. of Memory Controllers	1	1	2	1	2	1
SDRAM	4GB DDR3	4GB DDR4	8GB DDR4	4GB DDR3	8GB DDR4	4GB DDR3
QSPI Flash Memory	2x128 Mbit	2x512 Mbit	2x512 Mbit	2x128 Mbit	2x512 Mbit	2x128 Mbit
Host Interface	PCIe3x8	PCIe3x8	PCIe3x16	PCIe3x8	PCIe3x16	PCIe3x8
Frame Size (Bytes)	64 - 10000	64 - 10000	64 - 10000	64 - 10000	64 - 10000	64 - 10000
Cooling Solution	Active	Active	Active	Active	Passive	Passive
Temperature	0^{0}C to 45^{0}C	0^{0}C to 45^{0}C	0^{0}C to 45^{0}C	0^{0}C to 45^{0}C	-5^{0}C to 55^{0}C	-5^{0}C to 55^{0}C
Power Consumption (Max)	27 Watts	58 Watts	44 Watts	27 Watts	44 Watts	30 Watts
FPGA Technology	XC7VX330T	XCKU11P XCKU15P	XCVU5P XCVU7P XCVU9P	XC7VX330T	XCVU5P XCVU7P XCVU9P	XC7VX330T
Regulatory Compliance (Product Specific)	KCC*	KCC*	KCC*	KCC*	KCC*	KCC*
Regulatory Compliance (Common)	PCI-SIG$^{(R)}$, CE, CB, RoHS, REACH, cURus (UL), FCC, ICES, VCCI, RCM				PCI-SIG$^{(R)}$, NEBS Level 3, CE, CB, RoHS, REACH, cURus (UL), FCC, ICES, VCCI, RCM	

include Napatech, Endace, Intel, Advantech, etc. Napatech hardware cards are Field Programmable Gate Array (FPGA) based and used by many companies around the globe such as Cisco, IBM, ntop, Dell, Nokia, Symantec, Facebook, Intel, Xilinx, Yahoo, and many more [32].

Napatech provides many FPGA-based hardware acceleration cards or smart NICs but we require a smart NIC that has at least 3 interfaces. This is because Snort IPS mode requires three interfaces for proper functioning i.e. two network interfaces for network traffic bridging purposes (receive traffic on one interface and send it to the other and vice versa) and one network interface for management purposes. Therefore such Napatech smart NICs include NT40A01, NT100A01, NT40E3, and NT40A11. Table 3 summarizes the technical specifications of these cards [32].

We have selected NT40A01-SCC Napatech Smart NIC because it has four physical network interfaces (we require a minimum of three) and supports a 1Gbps network line rate. Through intelligent hardware flow distribution, it can support up to 128 packet streams [32]. It can also be integrated with the Snort IPS in inline mode but its installation and integration are a bit complex. First of all, software drivers for Napatech smart NIC are installed. Then the service starts for loading drivers and initializing smart NIC operations. After that, the network interfaces of Napatech smart NIC become available for use and can be seen through the "ifconfig" command. The command to run Snort with Napatech smartNIC is given below.

```
./snort -c /etc/snort/etc/snort.conf --daq afpacket
--daq-dir /usr/local/lib/daq --daq-mode inline -i
napa0:napa1 -Q
```

The aforementioned solutions for optimization of Snort IPS convert an ordinary Snort IPS machine into a fast and efficient IPS machine that can be deployed on a 1 Gbps network. The software solutions including DPDK and PF_RING free the CPU from capturing the network packets. These solutions bypass the CPU and through the help of customized drivers process the network packets more efficiently and fast. In the same way the hardware solution i.e. Napatech smartNIC takes away the responsibility of network packets capturing from the CPU and processes them itself very fast. This not only makes the packet-capturing process extremely fast but it also frees up the CPU for other essential tasks.

The DPDK and PF_RING vanilla mode run in user space whereas PF_RING ZC mode and Napatech smartNIC drivers run in kernel space. From the security perspective, running programs in user space is typically preferred over running them in kernel space. However, both PF_RING ZC and Napatech smartNIC drivers running in kernel space have undergone comprehensive security analysis, extensive testing, and evaluation. Consequently, these software and hardware technologies are deemed secure and efficient, making them widely adopted at a professional level with commendable outcomes. Therefore, the use of these technologies is considered safe, secure, and efficient.

IV. EXPERIMENTAL SETUP

The experimental setup for the comparison of different solutions for throughput performance enhancement of Snort IPS in inline mode is depicted in Figure 7. In this experimental setup, the Snort IPS is connected to the IXIA Breaking-Point system for throughput performance testing. One Snort IPS physical interface is directly connected to one physical

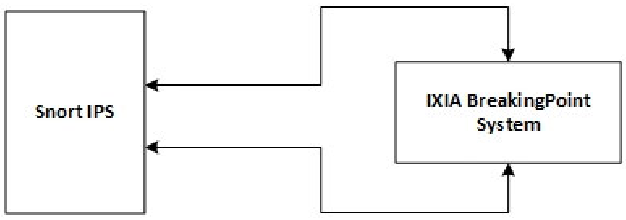


FIGURE 7. Experimental setup.

TABLE 4. IXIA breakingpoint machine details.

IXIA BreakingPoint Machine	
Server	HUAWEI RH2288 V3
Hard Disk	5 TB
RAM	96 GB
IXIA BreakingPoint Version	9.10

TABLE 5. Snort IPS machine details.

Snort IPS Server Machine	
Server	HUAWEI RH2288 V3
Operating System	Linux - CentOS-7_2009
Snort Version	2.9.13.0
RAM	64 GB
Hard Disk	2 TB
Network Interfaces	04

interface of the IXIA BreakingPoint system whereas the second Snort IPS physical interface is directly connected to the second physical interface of the IXIA BreakingPoint system.

In this configuration, network traffic generated from the IXIA BreakingPoint system is received at the first interface of the Snort IPS system. After the inspection of network traffic by the Snot IPS system, network traffic leaves the Snort IPS system from the second interface and is received at the second interface of the IXIA system. In this way, the Snort IPS operates in inline mode. Table 4 and Table 5 reveal the software and hardware specifications of the machines.

V. RESULTS AND DISCUSSION

The results obtained from the aforementioned setup reveal some interesting facts. During the experiment, the Snort IPS was run multiple times with no rules, 5 thousand rules, 10 thousand rules, 15 thousand rules, and 20 thousand rules. In addition, these experiments were performed for TCP and UDP traffic generated through the IXIA system. Figure 8 and Figure 9 illustrate the results obtained from these experiments. The graphs clearly show that the AF_PACKET packet-capturing library is outperformed by the rest of the software and hardware technologies. The PF_RING and DPDK libraries achieve almost 1G and Napatech smart NIC achieves a full 1G throughput line rate which is required in inline mode operation as shown in Table 6.

A sustained 1G traffic is generated by the IXIA BreakingPoint system. This traffic by large consists of benign traffic,

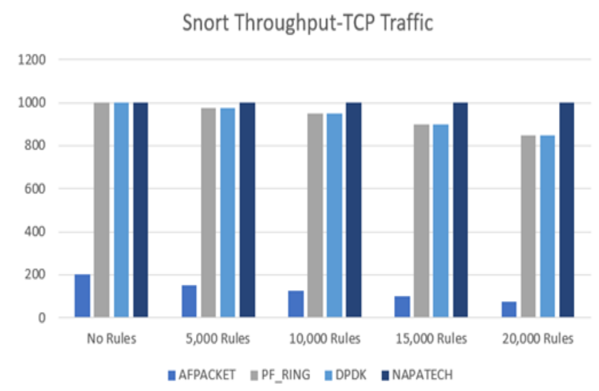


FIGURE 8. Snort throughput-TCP traffic.

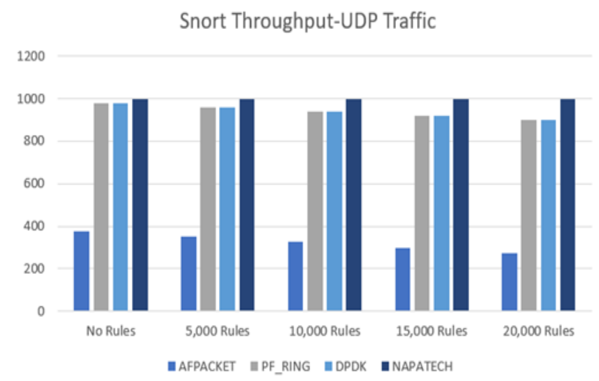


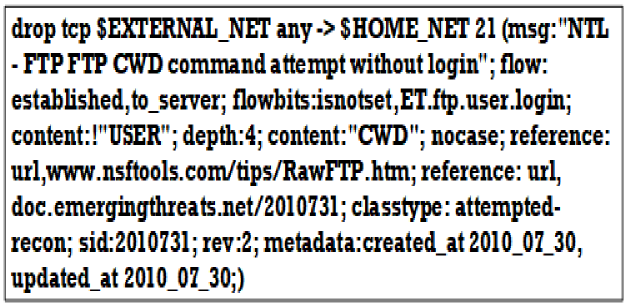
FIGURE 9. Snort throughput-UDP traffic.

although it contains a small portion of FTP malicious traffic. During the test, the rule-sets of 5, 10, 15, and 20 thousand rules contained rules for the detection of malicious FTP traffic along with other rules downloaded from the Snort community. We consider the "**FTP CWD**" and "**FTP MKDIR**" commands as malicious because these commands are used by the user before the login command.

Therefore as far as the attack traffic is concerned we are only injecting malicious FTP traffic through the IXIA BreakingPoint system. Although our rule-sets also contain rules for other types of attacks e.g. ICMP, DNS, HTTP, SMTP, SNMP, etc. The primary focus of the test was to check the Snort IPS throughput along with the detection of some traces of malicious traffic. The key point in the selection of the rule-set is that it should contain the rules related to the attacks that are to be launched. This means an attack will remain undetected if its detection rule is not present in the rule-set.

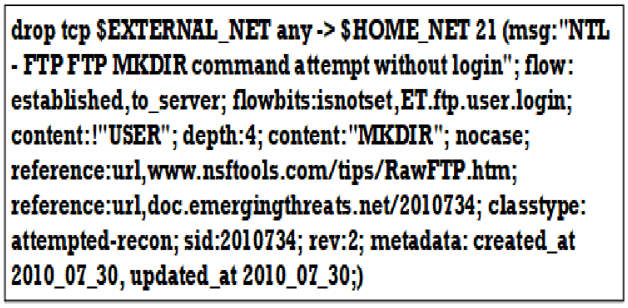
The first FTP rule as shown in Figure 10 with signature ID or sid 2010731, detects and blocks the "**FTP CWD**" command when a user uses this command without being logged in first. Similarly, the second FTP rule as shown in Figure 11 with sid 2010734, detects and blocks the "**FTP MKDIR**" command if the user uses this command without being logged in first.

In each test case, 100 instances of FTP malicious traffic were injected by the IXIA BreakingPoint system. The Snort



drop tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"NTL - FTP FTP CWD command attempt without login"; flow: established,to_server; flowbits:isnotset,ET.ftp.user.login; content:!"USER"; depth:4; content:"CWD"; nocase; reference: url,www.nsftools.com/tips/RawFTP.htm; reference: url, doc.emergingthreats.net/2010731; classtype: attempted-recon; sid:2010731; rev:2; metadata:created_at 2010_07_30, updated_at 2010_07_30;)

FIGURE 10. FTP Rule-1.



drop tcp $EXTERNAL_NET any -> $HOME_NET 21 (msg:"NTL - FTP FTP MKDIR command attempt without login"; flow: established,to_server; flowbits:isnotset,ET.ftp.user.login; content:!"USER"; depth:4; content:"MKDIR"; nocase; reference:url,www.nsftools.com/tips/RawFTP.htm; reference:url,doc.emergingthreats.net/2010734; classtype: attempted-recon; sid:2010734; rev:2; metadata: created_at 2010_07_30, updated_at 2010_07_30;)

FIGURE 11. FTP Rule-2.

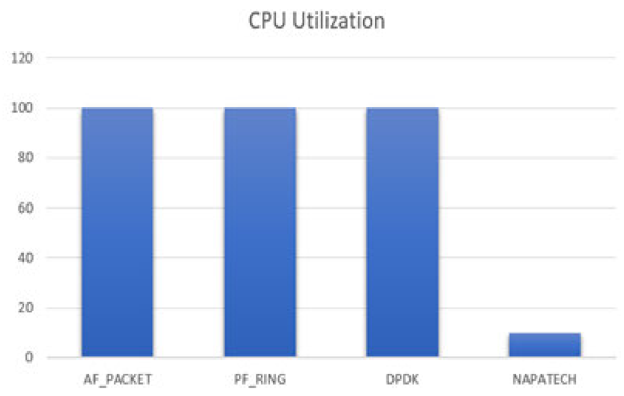


FIGURE 12. CPU utilization.

IPS successfully detected and blocked the malicious FTP traffic with 100 percent accuracy. This is because the Snort IPS contains the respective detection rule in its rule-set. This is quite obvious that if an attack is launched and its detection rule is not present in the Snort rule-set, Snort will not be able to detect the attack.

Figure 12 shows the CPU utilization of software and hardware technologies. It is very interesting that all the software technologies, during the execution, show 100% CPU utilization. This can be a matter of concern in a resource-intensive system where CPU utilization is already high. Therefore in such a scenario PF_RING and DPDK packet-capturing libraries are not recommended, although they achieve almost 1G throughput line rate. On the other hand, Napatech smart NIC not only achieves a full 1G throughput line rate but its CPU utilization (less than 5%) is also very low as compared to PF_RING and DPDK. This is very much suited for a heavily compute system. From the cost perspective, software technologies i.e. PF_RING and DPDK, are more suited for use as the hardware Napatech smart NIC is quite costly as compared to the software technologies. Generally, normal end users prefer a low-cost solution whereas, in an enterprise environment, a high-cost hardware solution is quite affordable.

Regarding the potential security considerations, these software-based and hardware-based solutions are quite established and mature. The customized drivers used by these solutions are quite capable of handling network packets efficiently and securely like the operating system TCP stack but at a much higher rate. Therefore there appears no evident security issue that can harm the normal functionality of the Snort IPS machine or even the organizational network.

The Snort configuration is quite simple and its important sections are shown below.

```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.1.0/24
ipvar EXTERNAL_NET any
var RULE_PATH /etc/snort/etc/rules
var SO_RULE_PATH /etc/snort/etc/rules/so_rules
var PREPROC_RULE_PATH
/etc/snort/etc/rules/preproc_rules
var WHITE_LIST_PATH /etc/snort/etc/rules/list
var BLACK_LIST_PATH /etc/snort/etc/rules/list
config logdir: /var/log/snort/
whitelist $WHITE_LIST_PATH/white_list.rules, \
blacklist $BLACK_LIST_PATH/black_list.rules
# syslog
output alert_syslog: LOG_AUTH LOG_ALERT
# site specific rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/app-detect.rules
include $RULE_PATH/NTL-attack_response.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/botnet-cnc.rules
.
.
.
```

The above experimentation reveals that the throughput of Snort IPS is also dependent on some other important factors such as:

- **Type of Rules:** The type of rules in the Snort IPS is a very important factor in the throughput of the Snort. If the rule set contains rules that inspect packet contents to a deeper level, the throughput of the Snort IPS drops. This is because in this scenario Snort IPS needs more time to perform deeper-level packet inspection. Therefore the more such rules the rule set of Snort IPS contains the less throughput is achieved by the Snort IPS and vice versa;
- **Type of Network Traffic:** The type of network traffic is also a very important factor in determining the throughput of the Snort IPS. If the whole network traffic or the majority of it is benign, the throughput of Snort IPS

TABLE 6. Experimental results.

Parameter	Software Based Solutions			Hardware Based Solutions
	AF_PACKET	PF_RING	DPDK	Napatech NT40A01-SCC
Version	Snort-2.9.13.0	7.0.0	17.02.1	11.2.0
Throughput	200 Mbps	≈ 1 Gbps	≈ 1 Gbps	1 Gbps
CPU Utilization	100%	100%	100%	< 5%
Hard Ware Dependency	No	No	Yes	Yes
Integration Complexity	Low	Low	High	High
Memory Mapping	No	Yes	Yes	On card memory
Circular Ring	No	Yes	Yes	On card memory
Large Memory Paging	No	Yes	Yes	On card memory
Execution Space	User space	Both	User space	Kernel space

will be higher. On the other hand, if the majority of the network traffic is malicious then the throughput of Snort IPS gets dropped;

- **Number of Rules:** The number of rules is also a very important factor for the throughput of Snort IPS. The throughput of Snort IPS is higher if the rule set of Snort IPS has less number of rules. Similarly, the throughput of Snort IPS is lower if the ruleset of Snort IPS contains a higher number of rules. Table 6 shows a comparison of different technical parameters of AF_PACKET, PF_RING, DPDK, and Napatech NT40A01-SCC smart NIC.

In the evaluation methodology, we have chosen some most relevant test parameters for the throughput comparison among different software and hardware technologies. These parameters including throughput, CPU utilization, hardware dependency, integration complexity, memory mapping, circular ring, and large memory paging have been chosen to ensure that the comparison focuses on the most pertinent and significant aspects. These parameters cover CPU, memory, and related important domains effectively for the throughput comparison of the aforementioned technologies.

As the Snort IPS is placed inline with the network, if it drops network packets due to low throughput, the whole corporate network communication gets disturbed. This communication could be crucial, i.e. it could be related to a remedial action against some sort of malicious activity in the corporate network. If these network packets are dropped due to the limitation of Snort IPS throughput, no remedial action would be taken against a detected malicious activity in the network. This could lead to devastating consequences for the entire corporate network. This attack scenario can be avoided by integrating one of the aforementioned solutions for achieving network traffic line rate.

VI. INTEGRATION WITH A SEIM

A security information and event management (SIEM) system has become an integral part of every organizational security management setup along with the IPS. Many latest sophisticated attacks remain undetected with IPS or any other single security device. Therefore a more comprehensive and holistic approach is needed to detect such sophisticated attacks. In other words, a single security device like IDS, IPS, antivirus, antimalware, firewall, etc. cannot guarantee to safeguard an organizational network on its own. Therefore such security devices should be deployed along with a SIEM system for the detection of such sophisticated attacks, especially zero-day attacks [33], [34]. We have integrated the Snort IPS with our proprietary SIEM system which is called Cyber Threat Management System or CTMS. The CTMS is capable of receiving alerts in syslog format generated by the Snort. This process is fast because the generated alerts are transferred by the syslog service of the operating system which is quite fast and efficient. The CTMS is capable of receiving all the formats that the Snort can output which is very flexible as compared to other SIEM solutions like OSSIM that do not offer such flexibility. Other open-source and proprietary SIEM solutions are also quite rigid in their functionality and often prove less pliable.

In this regard, it becomes obvious that the output of the Snort IPS should be integrated with the SIEM system to enhance the chances of attack detection [35]. For this purpose, some considerations must be taken into account to integrate IPS with the SIEM system:

- One of the main considerations for the integration of Snort IPS with the SIEM system is the format in which Snort IPS outputs its alerts. The Snort IPS should produce its output in an understandable format for the SIEM system. If the Snort IPS produces its output in a format that is not compatible with the SIEM system, then integration of Snort IPS with the SIEM system is not possible. The Snort IPS supports many output formats like Syslog, text-based alert, unified format, etc. It is necessary to adopt such an output format compatible with the SIEM system so that the output from the Snort IPS can be input into the SIEM system for the further attack detection process. Generally, the information that a SIEM system receives, is saved in some database. In this context, Syslog format is a good choice for the Snort IPS output format. If Snort IPS produces its output in Syslog format, it will not only be compatible with the SIEM system but it can also be sent through the operating system Syslog service to the SIEM system [36];

• From the integration of Snort with the SIEM system perspective, the SIEM system GUI is very important. The SIEM system GUI should provide a simple and intuitive graphical interface for the configuration of different parameters of the Snort. This facilitates the end user in the configuration of different performance parameters of Snort being compatible with the SIEM system;

• The machine on which the SIEM system runs must contain a dedicated network interface for accessing the SIEM system GUI which not only provides the SIEM system interface to the end user but also provides an interface for configuring the Snort IPS;

• Another very important factor is network traffic. All the network traffic traveling in the network should be input to the Snort so that it can apply the rule set on the network traffic for the detection of attacks.

VII. CONCLUSION AND FUTURE WORK

An intrusion prevention system (IPS) has become an integral part of every organization that prevents intrusions in the corporate network along with their detection. When deployed in inline mode, the Snort IPS must process the network traffic fast enough to match the network traffic line rate. But if we use Snort IPS with its standard packet-capturing library i.e. AF_PACKET, it remains unable to achieve network traffic line rate and packet loss starts to occur. To overcome this issue we must adopt some other packet-capturing mechanism to achieve the network traffic line rate.

This research work implements and analyzes some of the best software-based and hardware-based mechanisms for enhancing the network traffic throughput of Snot IPS. We have implemented PF_RING and DPDK packet-capturing libraries from the software domain and Napatech NT40A01-SCC smart NIC from the hardware domain. We have analyzed and compared the results obtained from these packet-capturing mechanisms along with their application domains. In the end, we have also discussed the integration of Snort IPS with a SIEM system that has proven to be very effective in the detection of sophisticated cyber-attacks when deployed along with other security devices. Therefore integration of Snort IPS with the SIEM system is very useful for the detection of the latest cyber-attacks.

In the future, additional software-based and hardware-based technologies, which are currently underutilized, could be incorporated and evaluated to boost the network traffic throughput of Snort IPS in its inline mode. In this regard, technologies like Netmap from the software domain and hardware solutions from Endace, Intel, etc. could be very productive.

ACKNOWLEDGMENT

The authors would like to thank SAUDI ARAMCO Cybersecurity Chair, Imam Abdulrahman Bin Faisal University for funding this project.

REFERENCES

[1] J. Li, C. Wu, J. Ye, J. Ding, Q. Fu, and J. Huang, "The comparison and verification of some efficient packet capture and processing technologies," in *Proc. IEEE Intl. Conf. Dependable, Autonomic Secure Comput., Intl. Conf. Pervasive Intell. Comput., Intl. Conf. Cloud Big Data Comput., Intl. Conf. Cyber Sci. Technol. Congr. (DASC/PiCom/CBDCom/CyberSciTech)*, Fukuoka, Japan, Aug. 2019, pp. 967–973.

[2] R. Khader and D. Eleyan, "Survey of DoS/DDoS attacks in IoT," *Sustain. Eng. Innov.*, vol. 3, no. 1, pp. 23–28, Jan. 2021.

[3] S. Saleem, M. Sheeraz, M. Hanif, and U. Farooq, "Web server attack detection using machine learning," in *Proc. Int. Conf. Cyber Warfare Secur. (ICCWS)*, Islamabad, Pakistan, Oct. 2020, pp. 1–7.

[4] W. S. Hwang, J. G. Shon, and J. S. Park, "Web session hijacking defense technique using user information," *Human-Centric Comput. Inf. Sci.*, vol. 12, p. 16, Apr. 2022.

[5] A. Alqahtani and F. T. Sheldon, "A survey of crypto ransomware attack detection methodologies: An evolving outlook," *Sensors*, vol. 22, no. 5, p. 1837, Feb. 2022.

[6] A. Thakkar and R. Lohiya, "A survey on intrusion detection system: Feature selection, model, performance measures, application perspective, challenges, and future research directions," *Artif. Intell. Rev.*, vol. 55, no. 1, pp. 453–563, Jan. 2022, doi: 10.1007/s10462-021-10037-9.

[7] Z. Ahmad, A. S. Khan, C. W. Shiang, J. Abdullah, and F. Ahmad, "Network intrusion detection system: A systematic study of machine learning and deep learning approaches," *Trans. Emerg. Telecommun. Technol.*, vol. 32, no. 1, p. e4150, Jan. 2021.

[8] M. G. Yaseen and A. S. Albahri, "Mapping the evolution of intrusion detection in big data: A bibliometric analysis," *Mesopotamian J. Big Data*, vol. 2023, pp. 138–148, Dec. 2023, doi: 10.58496/mjbd/2023/018.

[9] M. Dutta, and J. Granjal, "Towards a secure Internet of Things: A comprehensive study of second line defense mechanisms," *IEEE Access*, vol. 8, pp. 127272–127312, 2020.

[10] A. Gupta and L. S. Sharma, "Performance analysis and comparison of snort on various platforms," *Int. J. Comput. Inf. Syst. Ind. Manag. Appl.*, vol. 10, pp. 23–32, Jan. 2020.

[11] S. Thappa and A. Mailewa, "The role of intrusion detection/prevention systems in modern computer networks: A review," in *Proc. 53rd Midwest Instruct. Comput. Symp. (MICS)*, Apr. 2020, pp. 1–14.

[12] *Snort User Manual*. Accessed: Feb. 24, 2023. [Online]. Available: https://www.snort.org/downloads/

[13] H. Haugerud, H. N. Tran, N. Aitsaadi, and A. Yazidi, "A dynamic and scalable parallel network intrusion detection system using intelligent rule ordering and network function virtualization," *Future Gener. Comput. Syst.*, vol. 124, pp. 254–267, Nov. 2021.

[14] D. Jin, Y. Lu, J. Qin, Z. Cheng, and Z. Mao, "SwiftIDS: Real-time intrusion detection system based on LightGBM and parallel intrusion detection mechanism," *Comput. Secur.*, vol. 97, Oct. 2020, Art. no. 101984, doi: 10.1016/j.cose.2020.101984.

[15] M. Asif, S. Abbas, M. A. Khan, A. Fatima, M. A. Khan, and S.-W. Lee, "MapReduce based intelligent model for intrusion detection using machine learning technique," *J. King Saud Univ. Comput. Inf. Sci.*, vol. 34, no. 10, pp. 9723–9731, Nov. 2022, doi: 10.1016/j.jksuci.2021.12.008.

[16] H. Qadeer, A. Talat, K. N. Qureshi, F. Bashir, and N. Ul Islam, "Towards an efficient intrusion detection system for high speed networks," in *Proc. 17th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Islamabad, Pakistan, Jan. 2020, pp. 428–433.

[17] L. Shuai and S. Li, "Performance optimization of snort based on DPDK and hyperscan," *Proc. Comput. Sci.*, vol. 183, pp. 837–843, Jan. 2021.

[18] D. Zhang and S. Wang, "Optimization of traditional snort intrusion detection system," *IOP Conf. Ser., Mater. Sci. Eng.*, vol. 569, no. 4, Jul. 2019, Art. no. 042041.

[19] S. A. Changazi, I. Shafi, K. Saleh, M. H. Islam, S. M. Hussainn, and A. Ali, "Performance enhancement of snort IDS through kernel modification," in *Proc. 8th Int. Conf. Inf. Commun. Technol. (ICICT)*, Karachi, Pakistan, Nov. 2019, pp. 155–161.

[20] F. Erlacher and F. Dressler, "On high-speed flow-based intrusion detection using snort-compatible signatures," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 1, pp. 495–506, Jan. 2022.

[21] D. Fadhilah and M. I. Marzuki, "Performance analysis of IDS snort and IDS suricata with many-core processor in virtual machines against Dos/DDoS attacks," in *Proc. 2nd Int. Conf. Broadband Commun., Wireless Sensors Powering (BCWSP)*, Sep. 2020, pp. 157–162.

[22] A. Gupta and L. S. Sharma, "Performance evaluation of *Snort* and *Suricata* intrusion detection systems on Ubuntu server," in *Proc. ICRIC*, 2019, pp. 811–821.

[23] A. Waleed, A. F. Jamali, and A. Masood, "Which open-source IDS? Snort, Suricata or zeek," *Comput. Netw.*, vol. 213, Aug. 2022, Art. no. 109116, doi: 10.1016/j.comnet.2022.109116.

[24] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, Dec. 2019.

[25] T. Erney and M. M. Chowdhury, "A survey of intrusion detection and prevention systems," in *Proc. IEEE World AI IoT Congr. (AIIoT)*, Seattle, WA, USA, Jun. 2022, pp. 578–584, doi: 10.1109/AIIoT54504.2022.9817348.

[26] S. H. Abbas, W. A. K. Naser, and A. A. Kadhim, "Subject review: Intrusion detection system (IDS) and intrusion prevention system (IPS)," *Global J. Eng. Technol. Adv.*, vol. 14, no. 2, pp. 155–158, Feb. 2023.

[27] J. Guo, H. Guo, and Z. Zhang, "Research on high performance intrusion prevention system based on suricata," *Highlights Sci., Eng. Technol.*, vol. 7, pp. 238–245, Aug. 2022.

[28] *Myth Busting DPDK in 2020*. Accessed: Feb. 24, 2023. [Online]. Available: https://nextgeninfra.io/dpdk-myth-busting-2020/

[29] *Supported Hardware*. Accessed: Feb. 24, 2023. [Online]. Available: https://core.dpdk.org/supported/

[30] A. Belkhiri, M. Pepin, M. Bly, and M. Dagenais, "Performance analysis of DPDK-based applications through tracing," *J. Parallel Distrib. Comput.*, vol. 173, pp. 1–19, Mar. 2023.

[31] *PF_RING Documentation*. Accessed: Feb. 24, 2023. [Online]. Available: https://www.ntop.org/guides/pf_ring/

[32] *Napatech Reconfigurable Computing*. Accessed: Feb. 24, 2023. [Online]. Available: https://www.napatech.com

[33] A. R. Muhammad, P. Sukarno, and A. A. Wardana, "Integrated security information and event management (SIEM) with intrusion detection system (IDS) for live analysis based on machine learning," *Proc. Comput. Sci.*, vol. 217, pp. 1406–1415, Jan. 2023.

[34] S. E. Hajji, N. Moukafih, and G. Orhanou, "Analysis of neural network training and cost functions impact on the accuracy on IDS and SIEM systems," in *Proc. 3rd Int. Conf. Codes, Cryptol. Inf. Secur.*, Apr. 2019, pp. 22–24.

[35] T. Laue, T. Klecker, C. Cleiner, and K. O. Detken, "A SIEM architecture for advanced anomaly detection," *Open J. Big Data*, vol. 6, pp. 26–42, Aug. 2022.

[36] H. Studiawan, F. Sohel, and C. Payne, "A survey on forensic investigation of operating system logs," *Digit. Invest.*, vol. 29, pp. 1–20, Jun. 2019.

MUHAMMAD SHEERAZ received the master's degree in computer science (MSCS) from International Islamic University, Islamabad, in 2011. He is currently pursuing the Ph.D. degree from Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad. He has more than 12 years of working experience in the field of cyber security. His research interests include security information and event management, intrusion detection and prevention, secure programming, and cryptographic algorithm implementation.

MUHAMMAD HANIF DURAD received the first M.Sc. degree in physics from the University of Punjab, Pakistan, the second M.Sc. degree in systems engineering from Quaid-i-Azam University, Pakistan, and the Ph.D. degree in computer engineering from Beijing Institute of Technology (BIT), China. He is a Professor with Pakistan Institute of Engineering and Applied Sciences (PIEAS), Islamabad, Pakistan. He has over 40 publications in refereed international journals and conferences. He has received few research grants for ICT related projects from various national funding agencies. His research interests include computer networks, cyber security, cloud computing, parallel computing, grid computing, the Internet of Things (IoT), and computer architecture.

SHAHZAIB TAHIR (Senior Member, IEEE) received the B.E. degree in software engineering from Bahria University, Islamabad, Pakistan, in 2013, the M.S. degree in information security from NUST, Islamabad, in 2015, and the Ph.D. degree in information engineering from the City University of London, U.K., in January 2019. Currently, he is an Associate Professor and the Associate HoD of the Department of Information Security, NUST. He is also the Founder and the Chief Technical Officer of CityDefend Ltd., U.K. His research interests include applied cryptography and cloud security. He has been a TPC member of many international IEEE conferences. He is also an alumni of InnovateUK CyberASAP. He is a reviewer of many high impact journals, including IEEE Transactions on Dependable and Secure Computing, IEEE Journal of Biomedical and Health Informatics, IEEE ICC, FGCS (Elsevier), *Cluster Computing* (Springer), *Sadhna* (Springer), *Science China Information Sciences* (Springer), and *Neural Computing and Applications* (Springer).

HASAN TAHIR (Senior Member, IEEE) received the B.E. degree in software engineering from Bahria University, Islamabad, Pakistan, the M.S. degree in software engineering from the College of E&ME, NUST, and the Ph.D. degree in information security from the University of Essex, U.K. He is an Associate Professor and the Head of the Department Information Security, School of Electrical Engineering and Computer Science (SEECS), NUST. He specializes in computer security and the IoT. He actively researches applications of cryptography in one-to-one and group settings. His primary area of research is the use of physically unclonable functions for securing a group of devices. He was a recipient of the University of Essex Doctoral Scholarship Award.

SAQIB SAEED received the B.Sc. degree (Hons.) in computer science from International Islamic University Islamabad, Pakistan, in 2001, the M.Sc. degree in software technology from Stuttgart Technology University of Applied Sciences, Germany, in 2003, and the Ph.D. degree in information systems from the University of Siegen, Germany, in 2012. He is an Associate Professor with the Department of Computer Information Systems, Imam Abdulrahman Bin Faisal University, Dammam, Saudi Arabia. He is also a Certified Software Quality Engineer from the American Society of Quality. His research interests include human-centered computing, data visualization and analytics, software engineering, information systems management, and digital business transformation. He is a member of the advisory boards of several international journals. He is an Associate Editor of IEEE Access and *International Journal of Public Administration in the Digital Age*.

ABDULLAH M. ALMUHAIDEB received the B.S. degree (Hons.) in computer information system from King Faisal University, Saudi Arabia, in 2003, and the M.S. (Hons.) and Ph.D. degrees in network security from Monash University, Melbourne, Australia, in 2007 and 2013, respectively. He is currently an Associate Professor of information security, a Supervisor of the Saudi Aramco Cybersecurity Chair, and the Dean of the College of Computer Science and Information Technology, Imam Abdulrahman Bin Faisal University, Saudi Arabia. He has published two patents and more than 40 scientific articles in journals and premier ACM/IEEE/Springer conferences. His research interests include mobile security, authentication and identification, and ubiquitous wireless access.