Advances in Networks
2020; 8(2): 22-33
http://www.sciencepublishinggroup.com/j/net
doi: 10.11648/j.net.20200802.12
ISSN: 2326-9766 (Print); ISSN: 2326-9782 (Online)

Improving Intrusion Detection and Prevention System (IDPS) Performance in an IPv6 Environment

Adeel Sadiq*, Waleed Bul'ajoul

School of Science and Technology, Nottingham Trent University, Nottingham, UK

Email address:
11beeasadiq@seecs.edu.pk (A. Sadiq), bulajoul@gmail.com (W. Bul'ajoul)
*Corresponding author

To cite this article:
Adeel Sadiq, Waleed Bul'ajoul. Improving Intrusion Detection and Prevention System (IDPS) Performance in an IPv6 Environment. *Advances in Networks.* Vol. 8, No. 2, 2020, pp. 22-33. doi: 10.11648/j.net.20200802.12

Received: October 29, 2020; **Accepted**: November 9, 2020; **Published**: November 19, 2020

Abstract: This paper presents a comprehensive investigation, backed up by detailed simulations, that the default settings of the software based open source Intrusion Detection and Prevention Systems (IDPs) are not enough to thwart the network attacks in a modern high-speed IPv6-only environment. It aims to solve this problem by improving the processing capabilities of an IDPS in more than one way, with each method being totally independent from the other. The proposed solution can be implemented by any user running an IDPS, without needing escalated privileges. Using and IPv6 packet generator, it is shown that with the increase in IPv6 traffic in a fixed amount of time, the IDPS fails to analyse all the packets and starts dropping them. This phenomenon compromises the core functionality of IDPS which is to stop the unwanted traffic. A hybrid solution has been proposed to increase the performance of the IDPS. Our research involves only the system running an IDPS, with little to no tweaking of the other elements within a network like routers, switches and firewalls. The paper also talks briefly about the current and the future generation of the IDPSs. The simulation with the hybrid solution concludes that the performance is improved to a staggering 200%, approximately, compared to the built-in settings of the IDPS.

Keywords: Internet Protocol Version 6, Intrusion Detection and Prevention System, Maximum Transmission Unit, Fragmentation and Jumbo Packets, Kernel and Application Buffer, Packet Priority and Niceness

1. Introduction

IP addresses are needed to communicate in the online world, without these logical addresses the Interconnected networks will fall apart. IPv4 address pool has been depleted [1] and the new version of the protocol, IPv6 is on the rise. IPv6 is not widely understood and implemented. Researchers have focused more on prolonging the life of IPv4 than encouraging the deployment of IPv6 [2]. The future of Internet can only by sustained by IPv6, especially with Internet of Things (IoT) on the rise. According to the predicted network growth by Cisco [3], it can be reasonably assumed that the alternative technologies like Network Address Translation (NAT) will not be able to keep up for long. Foreseeing this, the World IPv6 Day was observed in 2011 and the protocol has seen considerable increase in its deployment ever since [4]. IPv6 was adopted as a technical standard in 2017 by Internet Engineering Taskforce (IETF), the global entity responsible for developing Internet standards. Its specification can be found in the Request for Comments (RFC) 8200 [5]. Therefore, the study talks about IPv6 only.

Security is a wormhole. It consists of multiple layers and hundreds of devices, protocols and standards, each spanning a universe of knowledge in its on. We are heavily reliant on IT and Networks infrastructure for our day to day operations, which makes their security a paramount importance. The network of an organisation is no longer an optional commodity but a critical asset, which is required for the growth and long-term sustainability. Networks share valuable data and information. Unfortunately, this essential communication opens a serious threat vector to the security of the interconnected machines and networks. A Denial of Service (DoS) attack can be mounted even on a complex service like cloud, making the use of IDPS immediately relevant [6]. IDPS is not the only comprehensive device against the security threats, it should be used in conjunction with other security devices in a layered form to provide adequate security [7].

This study has tried to use only one device, IDPS, in an

in-line mode. Intrusion systems work in either detection or prevention mode. However, this study improves the detection and prevention mechanism by allowing the IDPS to measure more packets in a fixed time. It is imperative to realise that IDPS – being a security device – provides its functionality by analysing packets. Once it cannot analyse all the packets and start dropping traffic, its function is compromised. Software based open source IDPS is the most common choice today.

This study is focussed on the state-of-the-art protocol and security measures, instead of improving the older soon-to-run-out protocol. It is logical to put efforts into securing new methods and work on their longevity, hence the motivation for this study.

However, due to the nature of present networks, IDPS needs to be extremely fast, capable of processing at least one gigabit person (Gbps) of traffic, which is the standard speed of any modern common ethernet port. However, the researchers have found that the present security devices, including IDPS, are unable to keep up in a high-speed network environment. With the default settings, no stable software based open source IDPS achieves this feat of Gbps.

Since insufficient work has been done on IDPS performance in IPv6, this novel research starts with investigating IPv6 behaviour in IDPS. A prototype network is designed to investigate the IPv6 IDPS performance, followed by a deep analysis on the output of the findings. Finally, a technical solution is implemented and evaluated to improve the IDPS performance in an IPv6 setting. Keeping in view the aforementioned objectives, the paper is organised into sections, each focussing on one aspect. Section 2 discusses the works already done in this domain. Section 3 describes the research methodology, followed by Section 4 on simulation results and analysis. Section 5 summarises the change in performance by changing different parameters and section 6 evaluates some of the proposed parameters that maybe be modified. Section 7 proposes and evaluates a hybrid solution to improve the IDPS performance while the final Section 8 concludes the study and gives some insight into the future works.

2. Related Work

Gehrke discussed how IPv6 impacts IDPS performance in a simulated environment [8]. He used Snort to observe the behaviour of the IPv6 packets in a network but did not mention any improvements. Our research evaluates a technical solution on improving IDPS performance.

Bul'ajoul started his comprehensive work on improving the IDPS performance using Snort and his work is the most relevant to this study [9]. In fact, this research is a carry forward to his work, but with the IPv6. He simulated the IDPS performance in a high-speed network, changing various parameters like number, size and speed of the packets and observed the IDPS degradation. He suggested to improve the performance using parallelisation.

Kumar and Kaur pointed out how IDPS Snort performs reasonably well with IPv4 nodes, but the same cannot be said when it comes to IPv6 [10]. They simulated many attacks on an IPv6 network and the IPv6 IDPS, Snort, did not show a satisfactory performance. Their ideas were further reinforced by detailed findings of Schütte who concluded that no current open source IDPS is capable to provide adequate security for IPv6 [11]. This work has tried to address this issue with an improved simulated IDPS performance in Snort.

Bul'ajoul found another way of improving the IDPS performance using Quality of Service (QoS) in addition to parallelisation [12]. However, the research was focused only on IPv4, confirming the fact that most of the network elements have been optimised for IPv4 over decades, while little work is done for IPv6 in comparison. The work is very relevant to the issue at hand but unlike this study, that used QoS feature in the network switches to improve the performance.

Elejla and team have proposed another method to improve the performance of the IDPS, but only for IPv6 Internet Control Message Protocol (ICMPv6), the protocol that provides the core functionalities of IPv6 [13]. They have argued that using the traditional packet based IDPS is not the ideal approach in high speed networks. Instead, they have shown an improved design with a higher accuracy and low false positives rate using flow based IDPS compared to the trivial packet inspection.

Finally, 'A New Architecture for Network Intrusion Detection and Prevention' [14], have presented a novel architecture that considerably improves the IDPS performance. They have used QoS in conjunction with Parallelisation that showed great processing enhancements under certain conditions. Again, that work is applicable only to IPv4 while this study is only useful for IPv6.

To solve these problems, the goal of the study is aimed at improving the IDPS performance for the newer IPv6, as much as possible, preferably up to a level that thrive a fast speed Gbps network. This paper is different to the previous studies since it only deals with the IPv6 and modifies only the IDPS parameters. In a network, a user may or may not have access to other routing and security devices, hence this research focuses mainly on configuring parameters that a person with access to IDPS can make use of.

3. Research Methodology

3.1. Network Traffic Generator

The network traffic throughout the study will be IPv6-only. An enterprise tool WAN Killer [15] is used for all the simulations. Almost all the modern machinery supports Ethernet interfaces, the speed of which is in Gbps. To analyse the IDPS performance, a bandwidth closer to Gbps needs to be generated. Unfortunately, there are not many tools capable of mounting a Gbps scale of IPv6-only attack. The famous open source tools like HPing3 and many others provide adequate options to generate IPv4 packets, but do not support IPv6 traffic in a Gbps capacity. Open source tools like Scapy, IPerf, NetScan Pro were not powerful enough to mount the required IPv6 bandwidth at the time of this study. Most of the tools are restricted to a few Mbps of pure IPv6 traffic. Using an

industrial and proprietary tool like WAN Killer was the only option and the way forward.

WAN Killer can target an IPv6 address, be it link-local or unicast address. It has the options of varying speed and size of the packets to user defined values. The bandwidth will be varied from 100Mbps to 1300Mbps, wherever required, with increments of 200Mbps. The MTU is also changed in later simulations to see how it affects IDPS.

3.2. Snort

Snort is an open source software based IDPS. Since it is free to use, it has received great interest of the research community and has become the most powerful and widely used IDPS software tool worldwide [16]. Snort consists of 5 main components which work together to output an intrusion: decoder, pre-processor, detection engine, logging and alerting system and output module [17]. There are many approaches when it comes to improving security through Snort. This research has focussed on packets processing capability, rather than writing rules to stop the malicious traffic. The packet processing is a precursor to the packet blocking. If Snort cannot process enough packets, the ability to discard malicious traffic will not matter. This study has made use of Snort in an in-line mode. IDPS will always be one of the first devices that a packet has to go through and running it in-line mode makes it a necessary hop that packets must traverse through. As an entry point of a network, an efficient and effective IDPS will solve most of the network security threats. If a threat is contained before entering a network, it cannot wreak havoc and will do minimum to no damage at its behest, hence the motivation for preferring this approach over others.

3.3. Prototype Network

The aim of the research is to improve IDPS performance. For the sake of simplicity and to keep the focus on the task at hand, this research considers a local network with a point to point connection, removing the complexities of the routing in a network. However, in the real world, the malicious user is usually well hidden behind strong proxies and VPNs, in an undisclosed location which may span over long geographical distances. Furthermore, once the traffic has reached the target machine, it will behave similarly irrespective of where it originated from, having little to no effect on IDPS in its functionality of processing and analysis, hence, the decision of using a non-complex network design.

Figure 1. *Simple Network Topology.*

The virtual machines were used to promote the learning curve, minimise real world implications and legal issues, with an 8GB of RAM and 4 cores of processor, which are typical of a modern computing system.

4. Simulations and Analysis

IDPS is running on the Ubuntu virtual machine while the traffic is generated for a fixed amount of time, mostly 5 seconds. It should be noted that due to the human error of starting and stopping the simulation manually, the value of seconds is a close approximate, which can result in a little deviation when the experiment is repeated. Only one parameter is changed in each simulation, keeping others constant.

A sample output of the simulation results mentions the duration for which the IDPS was run to process the packets and its frequency of packet analysis. The amounts of packets received, analysed, and dropped can be verified, along with the type of traffic which in all cases is IPv6. Similar simulations are generated with the bandwidth of 300, 500, 700 and 900, 1100, and 1300 Mbps, where necessary, and the individual detailed results are analysed after each experiment. A final subsection of Performance Comparison provides a better view and understanding of the effects of change in performance with the chance of each parameter, one at a time.

In all the tables, the bandwidth is in Mbps, the duration in seconds and the packet size in bytes.

4.1. Bandwidth

Instead of changing the number and size of packets, it is desirable to change the bandwidth. The bandwidth is a better index in judging IDPS performance instead of changing the size and number of packets. As a matter of fact, changing the number and/or size will change the bandwidth, essentially.

4.1.1. Experiment

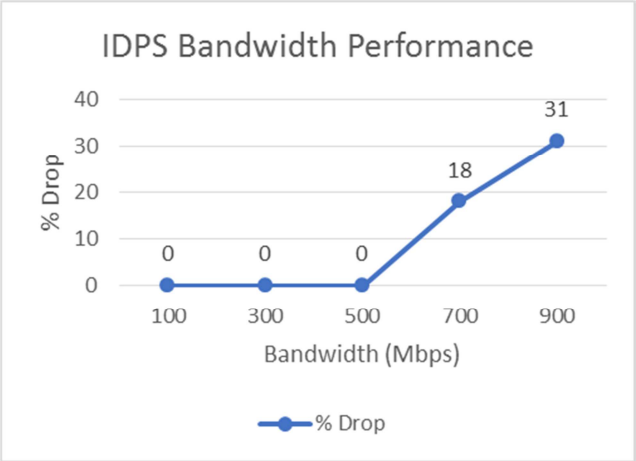

Figure 2. *Graphical IDPS Performance with Respect to Bandwidth.*

Table 1. *Tabular IDPS Performance with Respect to Bandwidth.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	5	32658	32938	100	0	0
300	5	120667	121887	100	0	0
500	5	149983	151662	100	0	0
700	5	167993	172316	100	37686	18
900	5	169057	171912	100	77778	31

4.1.2. Evaluation

The bandwidth simulation has showed that the IDPS is able to analyse all packets until 500Mbps easily but at reaching 700Mbps, the performance has decreased in the form of packets drop. IDPS is not able to keep up with high bandwidth especially when the bandwidth nears Ethernet capacity of Gbps. Evidently, the IDPS performance is reduced with the increase in bandwidth. When packets are sent at higher bandwidth, IDPS starts analysing the packet in run-time and stores the incoming packets in its buffer until it has reached its capacity. The packet drop occurs when the buffer is full, and no more packets can be entertained in either real-time or buffer storage. IDPS starts dropping these packets, irrespective of whether they are malicious or legitimate.

4.2. Time Duration

In this scenario, this time duration is increased to 10 seconds, keeping all other parameters the same.

4.2.1. Experiment

Table 2. *Tabular IDPS Performance with Respect to Time Duration.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	10	71979	72265	100	0	0
300	10	173238	174017	100	41113	19
500	10	175254	176820	100	175980	50
700	10	171419	174004	100	324221	65
900	10	172204	176716	100	450880	72

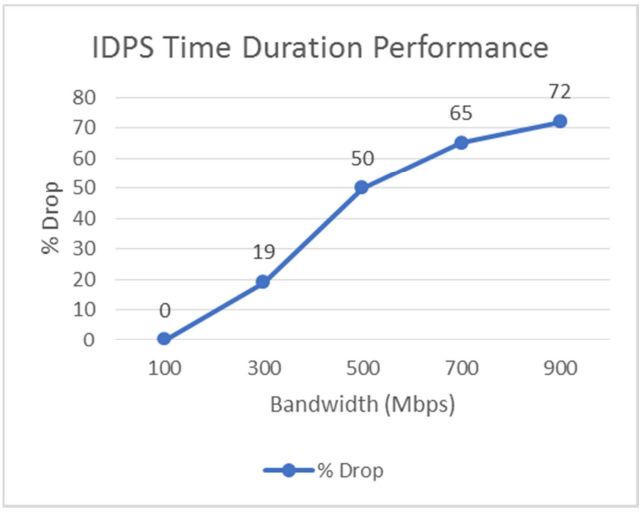

Figure 3. *Graphical IDPS Performance with Respect to Time Duration.*

4.2.2. Evaluation

When the IDPS runs for longer time, its performance is decreased considerably. Instead of crossing 500Mbps like in bandwidth simulations in section 4.1, the packet drop starts as early as 300Mbps. The buffer capacity is overflowed, and new packets have no space to be stored temporarily, hence the increase in packet drop.

4.3. Maximum Transmission Unit (MTU) and Fragmentation

MTU is advertised by routers in a network. When a packet size goes beyond MTU, fragmentation occurs since the network is unable to handle the packet size beyond a certain value. Usually, this value is set to 1500 bytes for the Ethernet networks as a standard [18]. In previous simulations, the packet size was set to a value lower than MTU, i.e. 1450 bytes. The following simulations changes the bandwidth value with a packet size greater than MTU, precisely to 2100 bytes, to see its effect on IDPS performance.

4.3.1. Experiment

Table 3. *Tabular IDPS Performance with Respect to Fragmentation.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	5	47869	48277	100	0	0
300	5	135336	136754	100	0	19
500	5	169949	171875	100	77646	31
700	5	167410	171936	100	153297	48
900	5	165319	172031	100	259478	61

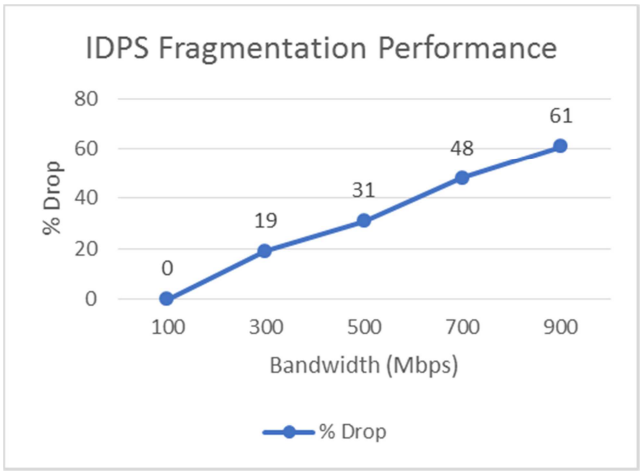

Figure 4. *Graphical IDPS Performance with Respect to Fragmentation.*

Table 4. *Processing Times with Different MTUs.*

Bandwidth	MTU	Time Taken	MTU	Time Taken	% Increase
100	1450	34	2100	46	36
300	1450	95	2100	127	34
500	1450	116	2100	152	31

4.3.2. Evaluation

Using 5 seconds time duration and 1450 bytes MTU, it was observed that IDPS can successfully analyse all packets without any drops up to 500Mbps. However, changing the packet size from 1450 to 2100 bytes changes the results altogether. A lot of resources are spent on fragmenting large packets and reassembling them. This fragmentation has drastic effects on the performance of IDPS. IDPS starts dropping packet right before 300Mbps with packet size greater than MTU, keeping all other factors constant. Another comparison in Table 4 also shows a huge rise in time taken to process the fragmented packets, further deteriorating the performance matrix to an enormous 30%, at least.

4.4. Hardware Specifications

Snort is just a software based IDPS, whose resources are dependent on underlying hardware. Improving the hardware will improve IDPS performance considerably. In this study, the memory and processing power of the Virtual Machine running the IDPS is reduced to half, 2GB of RAM and 2 cores of processor, to see the change it has on the performance of the IDPS.

4.4.1. Experiment

Table 5. *Tabular IDPS Performance with Respect to Hardware Resources.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	5	46534	46809	100	0	0
300	5	198759	114851	100	0	0
500	5	170273	172776	100	24600	13
700	5	170941	172821	100	105925	38
900	5	167202	172754	100	176776	51

Table 6. *Processing Times with Different Hardware Capacities.*

Bandwidth	Resources	Processing Time (s)	Resources	Processing Time (s)	% Change
100	High	34	Low	48	40
500	High	116	Low	132	14
900	High	135	Low	147	9

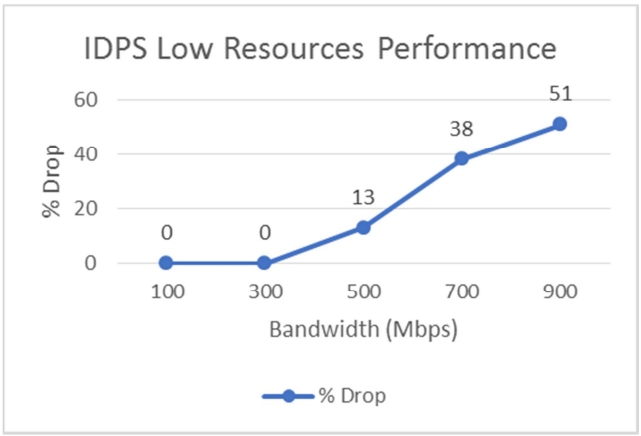

Figure 5. *Graphical IDPS Performance with Respect to Fragmentation.*

4.4.2. Evaluation

Reducing resource allocation to IDPS reduces its performance. IDPS requires more time to process the same packets and drops more packets given the same scenario with higher resources. A comparison with first simulation in section 4.1 reveals that with lower resources, IDPS starts dropping packets after 300Mbps instead of 500Mbps. This trend is continued at higher speeds, although at a reduced pace, demonstrating poorer performance with less resources.

5. Performance Comparison

To identify various factors that contribute to the performance matrix of IDPS, we controlled the size, speed, bandwidth, time duration and the underlying hardware in the simulations. The results of simulation have shown that IDPS performance is affected by various factors. Although we have changed one parameter at a time and kept others constant to analyse the effect of one criterion, the real networks work quite differently. Depending on the size of network and many other factors, more than one parameter will affect IDPS simultaneously, producing a deadly effect on overall performance. These parameters are presented on the same tables and graphs for the final side-by-side comparison.

Table 7. *Bandwidth Performance Tabular Comparison.*

Bandwidth	% Drop
100	0
300	0
500	0
700	18
900	31

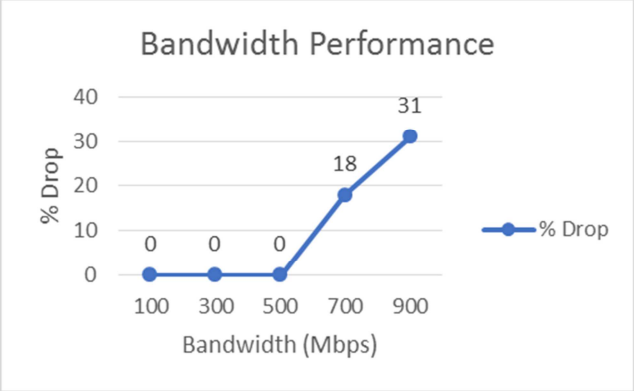

Figure 6. *Bandwidth Performance Graphical Comparison.*

Table 8. *Time Duration Performance Tabular Comparison.*

Bandwidth	Time	% Drop	Time	% Drop
100	5	0	10	0
300	5	0	10	19
500	5	0	10	50
700	5	18	10	65
900	5	31	10	72

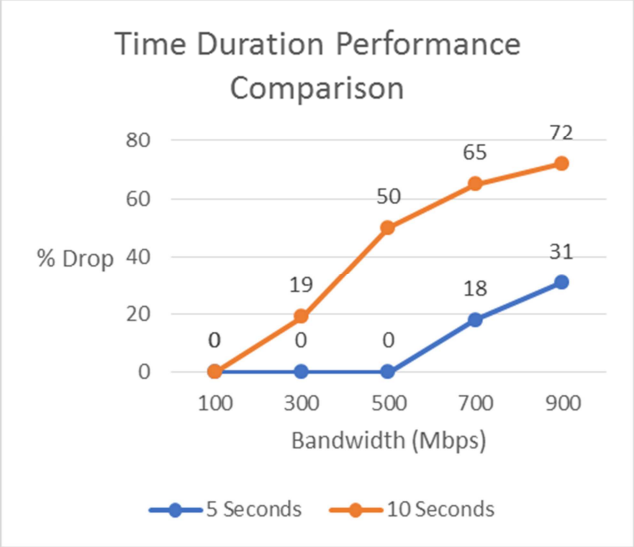

Figure 7. *Time Duration Performance Graphical Comparison.*

Table 9. *Fragmentation Performance Tabular Comparison.*

Bandwidth	Fragmentation	% Drop	Fragmentation	% Drop
100	No	0	Yes	0
300	No	0	Yes	19
500	No	0	Yes	31
700	No	18	Yes	48
900	No	31	Yes	61

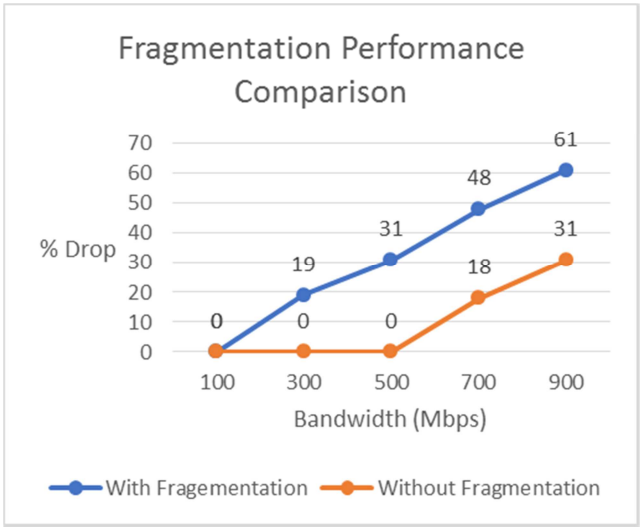

Figure 8. *Fragmentation Performance Graphical Comparison.*

Table 10. *Hardware Performance Tabular Comparison.*

Bandwidth	Resources	% Drop	Resources	% Drop
100	Low	0	High	0
300	Low	0	High	0
500	Low	13	High	0
700	Low	38	High	18
900	Low	51	High	31

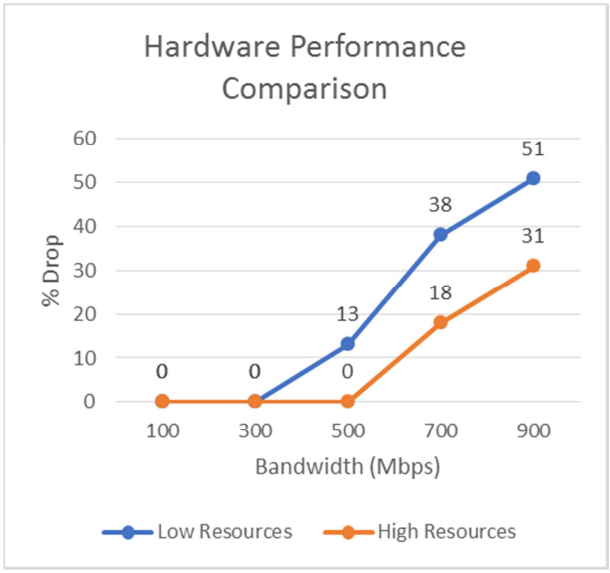

Figure 9. *Hardware Performance Graphical Comparison.*

The results of the simulations for IPv6 traffic can be summarised in below table, keeping in mind that only one parameter is changed at a time:

Table 11. *Factors Effecting IDPS Performance in an IPv6 Network.*

Parameter	Effects IDPS
Frequency of Packets	Yes
Size of Packets	Yes
Bandwidth	Yes
Fragmentation	Yes
Time Duration	Yes
Hardware Specifications	Yes

6. Improved Parameters

Based on the results of analysis in Section 4, IDPS can be improved by various methods. This section discusses some of them. Modern systems and networks support interfaces in Gbps. An IDPS is not very useful unless it can analyse at least one Gbps of traffic. However, a system is not always using that much traffic. Nonetheless, the security of a system cannot be compromised on the assumption of an average traffic consumption.

6.1. Hardware Resources

IDPS uses underlying hardware resources to perform its functions. Increasing these resources have a visible effect on its performance. The simulations in Section 4 used both high and low resources for the virtual machine, which in turn means for the IDPS. The simulations have shown that increasing hardware capability improves the performance adequately. More hardware resources will process more packets in a given time, keeping all other factors constant, hence improving IDPS performance. This led us to the conclusion that hardware resources, indeed, have a considerable effect on the performance of the IDPS, as can be seen from Figure 9.

6.2. Buffer Capacity

Whenever IDPS receives packets, it starts processing them in real time. Due to the nature of modern high-speed networks, IDPS starts buffering the packets that they have not processed yet. However, the buffer capacity is not unlimited. When the buffer capacity has reached its maximum value, then IDPS starts dropping packets, a point where it starts becoming dangerous since the network is more susceptible to security threats. There are two types of buffers; application and Operating System (OS). This section details how to use both types to possibly enhance the performance of IDPS.

6.2.1. OS Buffer

The OS buffer, sometimes referred as Kernel buffer, is the memory reserved by the underlying operating system for a short period of time before it is sent for processing, unlike cache which is the data that is already being processed. Both cache and buffer are used for improved performance of the services and processes.

Linux calls its buffer 'rmem' and 'wmem', short for receive memory buffer size and send memory buffer size, respectively. In the following simulation, the default values are changed from 212992 to 21299200 to see the effect on IDPS performance.

(i) Experiment

Table 12. *Tabular IDPS Performance with Respect to Increased Kernel Buffer.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	5	35093	35356	100	0	0
300	5	102557	103745	100	0	0
500	5	152218	153759	100	0	0
700	5	171579	173635	100	37656	18
900	5	165454	170259	100	85787	34

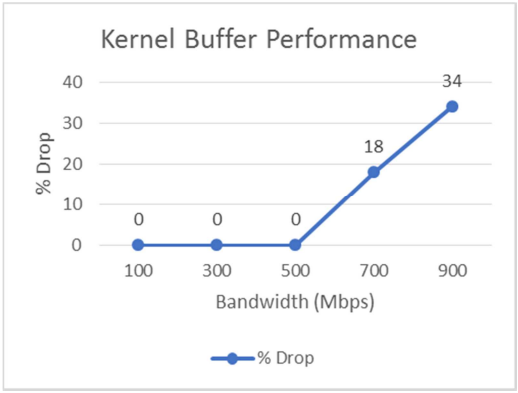

Figure 10. *Graphical IDPS Performance with Respect to Increased Kernel Buffer.*

(ii) Evaluation

Contrary to popular belief, IDPS has no effect on its performance due to the change in Kernel Buffer value as proven from the simulation. The results are like the standard first simulation in section 4.1 used to observe the bandwidth effect on IDPS. The tabular and graphical results show little to no effect at all.

6.2.2. Application Buffer

The second type of buffer is application buffer where every application assigns itself a buffer to handle its operations efficiently. This buffer is independent of kernel buffer. The kernel buffer serves the whole system while this type of buffer is local to every application. The Data Acquisition module in IDPS architecture, commonly known as DAQ, controls the buffer capacity, the value of which is defined in a complex configuration file called 'snort.conf". In all the previous simulations, we used the application buffer size of 1024 MB. In this simulation, the application buffer size of IDPS is increased to 2048 MB. The time is also increased from 5 seconds to 10 seconds to better analyse the effect of increased buffer. With 5 seconds tenure, the sensitivity of the simulation is decreased on account of latency and jitter and hence, not a good fit for the task at hand.

(i) Experiment

Table 13. *Tabular IDPS Performance with Respect to Increased Application Buffer.*

		Packets				
Bandwidth	**Duration**	**Received**	**Analysed**	**%**	**Drop**	**%**
100	10	73454	73925	100	0	0
300	10	230305	231924	100	0	0
500	10	327259	329828	100	0	0
700	10	336788	340112	100	163584	33
900	10	339672	344349	100	316002	48

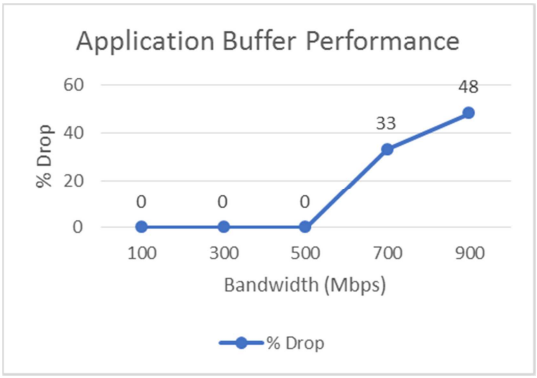

Figure 11. *Graphical IDPS Performance with Respect to Increased Application Buffer.*

(ii) Evaluation

With 1024MB memory buffer, IDPS started dropping packets at 300Mbps but it was able to sustain itself just after 500Mbps by using 2048MB buffer capacity, increasing the IDPS performance by almost two-folds.

Table 14. *Tabular IDPS Performance with Respect to Different Buffer Types.*

Bandwidth	Buffer Type	% Drop	Buffer Type	% Drop
100	Kernel	0	Application	0
300	Kernel	19	Application	0
500	Kernel	50	Application	0
700	Kernel	65	Application	33
900	Kernel	72	Application	48

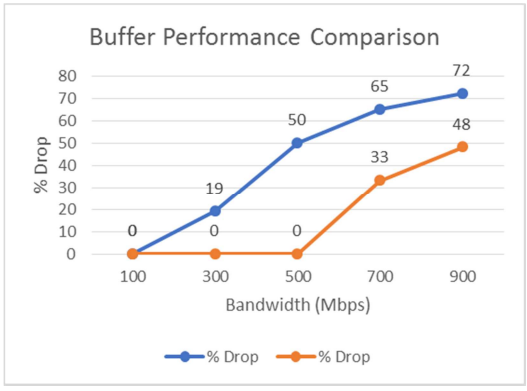

Figure 12. *Graphical IDPS Performance with Respect to Different Buffers.*

Although no change was observed by changing the kernel buffer, but a significant improvement was observed by modifying the application buffer size to an increased value.

Table 15. *Buffer Effecting IDPS Performance in IPv6 Network.*

Parameter	Effects IDPS
Kernel Buffer	No
Application Buffer	Yes

6.3. Process Priority and Niceness

Linux uses priority and niceness to assign preferences to the processes. Nice values range from -20 to +19 with +19 as the lowest priority whereas priority changes its value from 0 to 139 [19]. Whether the process is real time or user based, the logic of lower number corresponding to higher priority remains true. The priority for a real-time process can be changed by using 'renice' command [20]. However, it was found that in this scenario, changing the priority or niceness value did not yield any results because IDPS was the only main process running on the machine. This method may improve the result in a real-world scenario where many other processes may be running on the same machine where IDPS is installed. By changing the niceness, our simulation gave the same results as of the first bandwidth simulation in section 4.1. Hence, it can be concluded that the process priority in a standalone Linux system has no effect on the performance of the IDPS.

Table 16. *Priority Effecting IDPS Performance in IPv6 Network.*

Parameter	Effects IDPS
Process Priority	No
Process Niceness	No

6.4. Jumbo Packets

It was established before that fragmentation takes a heavy toll on IDPS performance. It causes break down of packets into smaller chunks and then reassembling them again, increasing the packet processing time to at least 30%. These packets of increased size are referred to as Jumbo Packets or Jumbograms. In this simulation, the MTU of the network is increased form 1500 bytes to 9000 bytes to allow the IDPS to process large packets without compromising its functional ability. The MTU of the network is increased both on sender, and receiver (and any other nodes in the network, if any) sides. The command line is used to change the MTU of ethernet interfaces in Linux while Network Interfaces in Device Manager is used for the same purpose in the Windows OS. In a traditional Snort output, the 'Frag3 Statistics' can verify the fragmentation occurrence.

6.4.1. Experiment

Table 17. *Tabular IDPS Performance with Respect to Changing MTU*

#	Bandwidth	MTU	Packet Size	% Drop
1	500	9000	1500	0
2	500	9000	2000	0
3	500	9000	2500	0
4	500	9000	3000	0

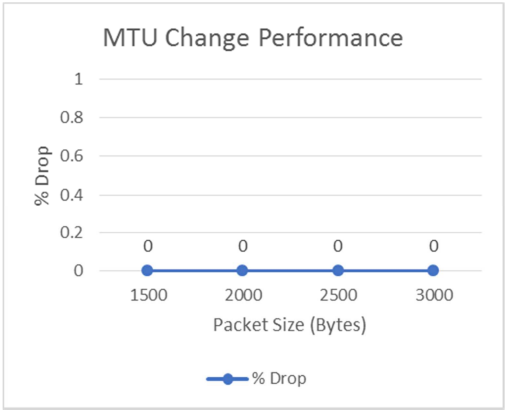

Figure 13. *Graphical IDPS Performance with Respect to Changing MTU.*

6.4.2. Evaluation

Changing the MTU value shows significant improvement for IDPS compared to a typical scenario. Snort started dropping packets at 300Mbps when the packet size was changed to 2100 bytes in a previous simulation, but it withstood 500Mbps with the increased change in packet size when its MTU was changed to 9000 bytes.

Table 18. Tabular IDPS Performance with Respect to Changing MTUs.

		MTU = 1500		MTU = 9000	
#	Bandwidth	Packet Size	% Drop	Packet Size	% Drop
1	100	2100	0	3000	0
2	300	2100	19	3000	0
3	500	2100	31	3000	0
4	700	2100	48	3000	18
5	900	2100	61	3000	31

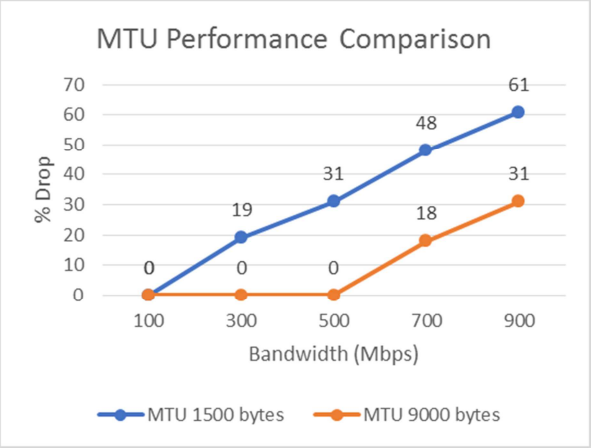

Figure 14. Graphical IDPS Performance with Respect to Different MTUs.

Increasing the MTU from 1500 to 9000 enables IDPS to process more packets since its resources are not used for reassembling. When the bandwidth is reached closer to 1Gbps, the increased MTU drops 30% traffic compared to 60% drop with a standard MTU.

For this to work, all network elements within a network should support the increased MTU and the value needs to be coherent throughout the lifecycle of the packet. Even if a single node is not supporting the extended MTU, it will fragment the packet it receives. Changing the MTU prevents the IDPS from dropping the packets when the packet size is increased considerably. However, changing MTU is a deterrent measure. It does not improve the IDPS performance directly per say, it helps it to prevent a denial of service condition. Once the DoS attack has been successfully carried out, more attacks can be mounted since the network security devices are no longer able to analyse every packet to prevent every malicious attempt.

Table 19. MTU Effecting IDPS Performance in IPv6 Network.

Parameter	Effects IDPS
MTU	Yes

6.5. Multithreading

According to the official documentation of Snort, the latest stable version does not support multithreading or parallel processing [21]. Irrespective of the cores of the processor, it always uses only one thread to carry out its activities. Multithreading is analogous to load distribution, which allows a single application to run in multiples processes in parallel, providing a boost to the performance. Suricata, another software based IDPS, has been supporting multithreading for quite some time [22]. Unfortunately, Snort had seen no development in this area. The beta version of Snort can emulate a condition of multithreading, but it does not support load balancing, yet. The beta Snort 3 is expected to support a maximum of 8 threads [23].

In this simulation, we will only test the multithreading capability of the underdeveloped Snort. If the multithread test succeeds, it is only a matter of distributing the packets to different instances of DAQ for load balancing, essentially increasing the performance of IDPS. However, it is a prerequisite for underlying hardware processor and software operating system to support multithreading if this feature is to be used in the applications.

6.5.1. Experiment

Table 20. Tabular IDPS Performance with Respect to Increasing Threads.

Threads	Packets Processing Frequency
1	4418
2	9001
3	13644
4	18139

The above table was obtained by running the simulation by incrementing the number of threads by one in each subsequent simulation.

6.5.2. Evaluation

The IDPS shows successful results of creating multiple threads. The packets received were copied and that same copy was sent to four different instances for parallel processing. The packets processing per seconds was increased from over four thousand in single thread to over eighteen thousand in four (multi)threads, showing an increase of almost four times, each thread behaving as a standalone process. However, it is not necessary that the performance is always increased to the number of times of instances. It all comes down to how the traffic is being distributed to different threads of the DAQ module.

Multithreading shows a visible increase in the output performance of an IDPS. Multithreading greatly improves the efficiency of any programme and process, IDPS is no different. However, since IDPS are security devices with very stringent requirements, configuring them to use this feature is not easy, especially using Snort. Snort, as of now, does not internally fan-out packets to other cores. Therefore, reproducing the results in this simulation will not be an easy task, not to mention that it may be changed entirely once the commercial version of the beta product is launched.

Table 21. Number of Threads Effecting IDPS Performance in IPv6 Network.

Parameter	Effects IDPS
Threads	Yes

7. Proposed Hybrid Solution

In its factory default state, software based IDPS are unable to support a Gbps traffic but can only work efficiently to a few hundred Mbps, as was seen throughout this research. After carefully implementing and technically evaluating different methods of improving IDPS performance, this section combines all the results from the research to produce a single viable solution that can be used in a live network based on the experiments and evaluations from Sections 4 and 5. The term hybrid refers to the fact that it will comprise not only the software aspects of the IDPS, but the hardware as well. Since the load balancing feature of the multithreading is still experimental, it is not considered in the final experiment, albeit a promising feature in the performance improvement. Combining the results from the study, the following parameters are used for the final simulation:

Table 22. *Verified Parameters for Maximum Performance for IDPS.*

Parameter	Value
RAM	8 GB
Processor	i7
Processor Cores	4
Application Buffer	7 GB
MTU	9000 bytes
Multithreading	Not Applicable
Time Duration	5 - 10 seconds
Bandwidth	0.5 - 1.3 Gbps

7.1. Experiments

The bandwidth is capped at 1.3Gbps due to the limitation of WAN Killer to generate purely IPv6 traffic. The time durations of simulation are set to 5 and 10 seconds, respectively.

Table 23. *Tabular IDPS Performance Comparison with Built-in and Proposed Parameters @ 5 seconds.*

Bandwidth	Duration	Traditional Drop %	Modified Drop %
500	5	0	0
700	5	18	0
900	5	31	0
1100	5	55	0
1300	5	64	0

Table 24. *Tabular IDPS Performance Comparison with Built-in and Proposed Parameters @ 10 seconds.*

Bandwidth	Duration	Traditional Drop %	Modified Drop %
100	10	0	0
300	10	19	0
500	10	50	0
700	10	65	0
900	10	72	21

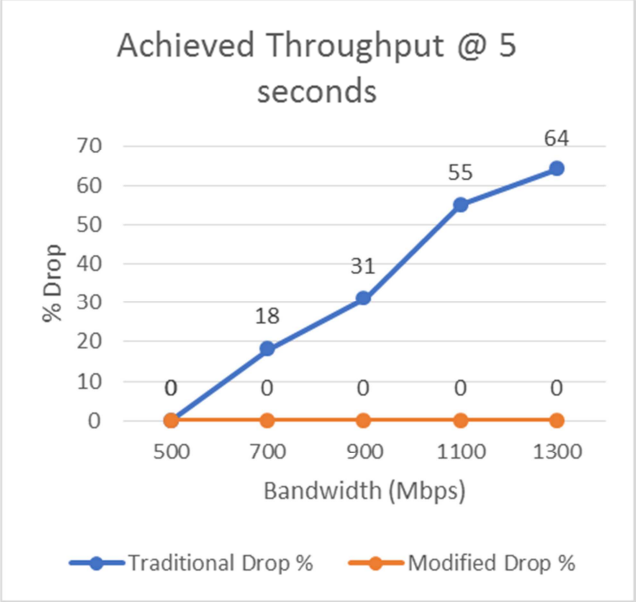

Figure 15. *Graphical IDPS Performance Comparison with Built-in and Proposed Parameters @ 5 seconds.*

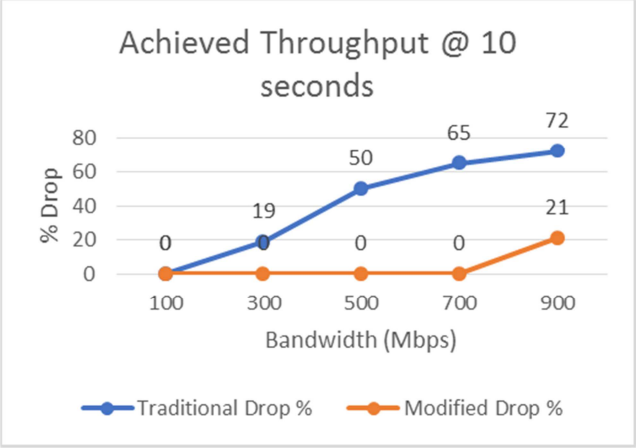

Figure 16. *Graphical IDPS Performance Comparison with Built-in and Proposed Parameters @ 10 seconds.*

Table 25. *Final Throughput Using Modified Parameters.*

Parameters	Duration	Throughput
Standard	5	500
Modified	5	1300
Standard	10	200
Modified	10	700

7.2. Recommendation

Using the verified parameters, the IDPS was able to handle traffic up to 1.3Gbps and 700Mbps without any drops for 5 and 10 seconds, respectively. The IDPS showed considerable depreciation at 900Mbps in longer duration, but using built-in values, this deflation starts as early as 300Mbps. Although the target of 1 Gbps was not achieved for prolonged 10 second tenure, but we still managed to improve the result by a huge margin of almost 250%.

7.3. Limitations

Verified from the simulations, IDPS performance is limited by both hardware and software. At this stage, software changes are more important since most of the modern machines have enough hardware capabilities, it is the efficient use of that hardware that becomes a bottleneck. Only through software, we can control and optimise the hardware performance. The beta version of Snort IDPS is expected to address some of these limitations, but nothing can be said for sure. The ability to generate a pure IPv6 traffic measuring up to tens of Gbps is also a problem. The unavailability of load balancing featuring in the alpha version of DAQ also restricts the users to enhance the performance further.

8. Conclusion and Future Work

8.1. Conclusion

The aim of the research was to improve the performance of an IDPS in an IPv6-only scenario. Different open-source software based IDPS are available, with Snort taking the lead worldwide. The technical solution proposed and verified in the simulations was able to achieve the goal set for this study. The performance was improved to an impressive 250% in longer duration as a bandwidth increase from 200Mbps to 700Mbps with zero packet loss. For a shorter tenure, the initial value of 500Mbps was increased by 160% to 1300Mbps, with further testing being restricted by the ability to generate more than 1300Mbps of pure IPv6 traffic. More research needs to be done and their findings can be inculcated to this study as an extended solution to take this key value to the scale of multiple gigabits per second. The authors believe it is too immature to draw any conclusions on the performance capabilities and comparison analogies of the trial Snort without an official release.

8.2. Future Work

The work done in this study can be taken forward in more than one way. The simulations were run for 5 and 10 seconds, this time can be increased, and a new solution devised that will work in a prolonged environment. A similar study can be carried out with a different packet generator. In future, maybe a strong open-source C++ based packet generator is developed, capable of generating IPv6 traffic up to 10 Gbps. Most of the packet generators now are python based, which allows only a few hundred Mbps of traffic, at most. Furthermore, a whole new era of research will be opened when the next generation of software based IDPS, Snort 3, is released. It is too early to decide whether the scope of this research will be applicable to Snort 3 in any way. It may obsolete all the research done on previous versions by all the researchers or may keep some of the features from the old releases. The latter is more likely. One of the most sought features in an IDPS is parallel processing. Once this multithreading is implemented and integrated in the Snort, new areas pertaining to handling and dissemination of traffic via DAQ module will open for the research community.

References

[1] RIPE NCC, 2019. The RIPE NCC has run out of IPv4 Addresses, *RIPE NCC* https://www.ripe.net/publications/news/about-ripe-ncc-and-ripe/the-ripe-ncc-has-run-out-of-ipv4-addresses [Accessed 1 Aug 2020].

[2] Bly, Jennifer. 2014. Why Is the Transition to IPv6 Taking So Long? *Team ARIN* https://teamarin.net/2014/08/13/transition-ipv6-taking-long/ [Accessed 1 Aug 2020].

[3] Cisco, 2016. Global – 2021 Forecast Highlights, VNI Complete Forecast Highlights, *Cisco* https://www.cisco.com/c/dam/m/en_us/solutions/service-provider/vni-forecast-highlights/pdf/Global_2021_Forecast_Highlights.pdf [Accessed 1 Aug 2020].

[4] Internet Society, 2018. State of IPv6 Deployment 2018, *Internet Society* https://www.internetsociety.org/resources/2018/state-of-ipv6-deployment-2018/ [Accessed 1 Aug 2020].

[5] Deering, S. and Hinden, R. 2017. Internet Protocol, Version 6 (IPv6) Specification, RFC8200, IETF https://tools.ietf.org/html/rfc8200 [Accessed 1 Aug 2020].

[6] Mishti D. et al. 2016. *International Journal of Applied Information Systems (Foundation of Computer Science)*, vol. 10, No. 5, pp 18-26.

[7] Chellappan, K. 2015. Layered Defense Approach: Towards Total Network Security, *International Journal of Computer Science and Business Informatics*, Vol. 15, No. 1, pp. 13-22.

[8] Gehrke, K. 2012. *The Unexplored Impact of IPv6 On Intrusion Detection Systems*, Master's Thesis, Naval Postgraduate School.

[9] Bul'ajoul, W. et al. 2013. Network Intrusion Detection Systems in High-Speed Traffic in Computer Networks, *IEEE 10th International Conference on e-Business Engineering*, pp. 168-175.

[10] Kumar, S. and Kaur, R. 2013. IPv6 Network Security Using Snort, *Journal of Engineering, Computers & Applied Sciences (JEC&AS),* Volume 2, Issue 8, pp. 17-22.

[11] Schütte, M. 2013. Design and Implementation of an IPv6 Plugin for the Snort Intrusion Detection System, *Magdeburger Journal zur Sicherheitsforschung*, 2, 409–452.

[12] Bul'ajoul, W. et al. 2015. Improving network intrusion detection system performance through quality of service configuration and parallel technology, Journal of Computer and System Sciences, Volume 81, Issue 6, pp. 981-999.

[13] Elejla, E. et al. 2018. Flow-Based IDS for ICMPv6-Based DDoS Attacks Detection, *Arabian Journal for Science and Engineering*, 43, pp. 7757–7775.

[14] Bul'ajoul, W. et al. 2019. A New Architecture for Network Intrusion Detection and Prevention, IEEE Access, vol. 7, pp. 18558-18573.

[15] SolarWinds, 2020. Network Traffic Generator and Stress Test, *SolarWinds* https://www.solarwinds.com/engineers-toolset/use-cases/traffic-generator-wan-killer [Accessed 1 Aug 2020].

[16] Snort, 2020. Snort – Network Intrusion Detection and Prevention System, *Snort* https://www.snort.org/ [Accessed 1 Aug 2020].

[17] Albin, E. and Rowe, N. 2012. A realistic experimental comparison of the Suricata and Snort intrusion-detection systems, *IEEE 26th International Conference on Advanced Information Networking and Applications (WAINA)*, pp. 122–127.

[18] Hornig, C. 1984. A Standard for the Transmission of IP Datagrams over Ethernet Networks, RFC894, *IETF* https://tools.ietf.org/html/rfc894 [Accessed 1 Aug 2020].

[19] AskUbuntu, 2020. Process 'niceness' vs. 'priority', *AskUbuntu* https://askubuntu.com/questions/656771/process-niceness-vs-priority [Accessed 1 Aug 2020].

[20] Mishra, C. 2019. A brief guide to priority and nice values in the linux ecosystem, *Medium* https://medium.com/@chetaniam/a-brief-guide-to-priority-and-nice-values-in-the-linux-ecosystem-fb39e49815e0#:~:text=In%20Linux%20system%20priorities%20are,default%20and%20%2B19%20is%20lowest. [Accessed 1 Aug 2020].

[21] Snort Users Manual. 2020. Snort Users Manual 2.9.16, *Snort*, https://snort.org/documents/1 [Accessed 1 Aug 2020].

[22] Suricata, 2016. Runmodes – Suricata 4.1.0-dev Documentation, *Suricata* https://suricata.readthedocs.io/en/suricata-4.1.3/performance/runmodes.html [Accessed 1 Aug 2020].

[23] Snort 3 User Manual. 2020. Snort 3 User Manual, *Snort* https://snort-org-site.s3.amazonaws.com/production/release_files/files/000/013/581/original/snort_manual.pdf [Accessed 1 Aug 2020].