Journal of Computer and System Sciences 81 (2015) 981–999

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

Improving network intrusion detection system performance through quality of service configuration and parallel technology

Waleed Bul'ajoul [a,*], Anne James [a], Mandeep Pannu [b]

[a] Faculty of Engineering and Computing, Coventry University, Coventry, UK
[b] Department of Computer Science, Kwantlen Polytechnic University, Surrey, British Columbia, Canada

ARTICLE INFO

Article history:
Received 10 March 2014
Received in revised form 12 September 2014
Accepted 22 September 2014
Available online 18 December 2014

Keywords:
Network security
Intrusion detection system
Intrusion protection system
Parallel processing
Switch configuration
Quality of Service

ABSTRACT

This paper outlines an innovative software development that utilises Quality of Service (QoS) and parallel technologies in Cisco Catalyst Switches to increase the analytical performance of a Network Intrusion Detection and Protection System (NIDPS) when deployed in high-speed networks. We have designed a real network to present experiments that use a Snort NIDPS. Our experiments demonstrate the weaknesses of NIDPSs, such as inability to process multiple packets and propensity to drop packets in heavy traffic and high-speed networks without analysing them. We tested Snort's analysis performance, gauging the number of packets sent, analysed, dropped, filtered, injected, and outstanding. We suggest using QoS configuration technologies in a Cisco Catalyst 3560 Series Switch and parallel Snorts to improve NIDPS performance and to reduce the number of dropped packets. Our results show that our novel configuration improves performance.

1. Introduction

In order to provide new developments and highest-quality services, companies implement the latest technologies in their infrastructure. A company's network plays a vital role in its business projects. Keeping the computer network up-to-date with the latest software and security techniques is essential for success and progress. Reliability and safety are the major concerns in enabling a company to achieve success and boost its progress. However, networks can also be considered a major risk in any business project. Security issues have increased as technology has advanced [1]. Fuchsberger [2] reported that, according to a survey conducted by the Federal Bureau of Investigation and Crime Scene of Investigation (FBI/CSI), viruses are behind many attacks on business networks. Moreover, Denial of Service (DoS) attacks and unauthorised user access (which can be initiated from external or internal LAN sources) have also increased dramatically. It is also noticeable that nowadays there are powerful intrusion tools available, allowing hackers to attack networks even if they know little of the software. Attackers can now use several tools simultaneously to achieve an objective. The 9th Annual Worldwide Infrastructure Security Report and ATLAS 2013 data report [3] said the number of Distributed Denial of Service (DDoS) attacks has grown significantly, nearly doubling on a year-to-year basis between 2006 and 2010. The size peak of attacks in 2013 increased by over 200 percent from the previous year, with the largest reported attack at 309 Gbps, and with multiple respondents reporting attacks larger than 100 Gbps, the previous largest reported attack size. Additionally, in

* Corresponding author at: EC building, Coventry University, Priory Street, Coventry CV1 5FB, UK.
E-mail addresses: Bulajouw@coventry.ac.uk (W. Bul'ajoul), a.james@coventry.ac.uk (A. James), mandeep.pannu@kpu.ca (M. Pannu).

http://dx.doi.org/10.1016/j.jcss.2014.12.012
0022-0000/

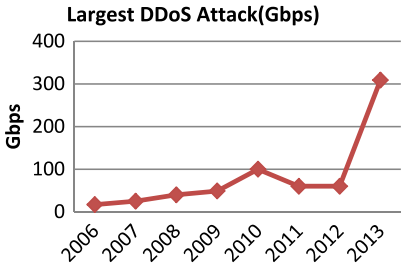

Fig. 1. Largest DDoS attack reported by Arbor networks [2].

2013, ATLAS observed more than 8x the number of attacks over 20 Gbps as compared to 2012 (see Fig. 1). Therefore, security products, such as firewalls, vulnerability assessment tools, antivirus programs, and Network Intrusion Detection and Prevention Systems (NIDPSs), are utilised to reduce the risk of attacks. However, even these measures are not 100 percent effective in protecting networks. One problem is that in heavy traffic, packets can be dropped prior to analysis [4–6]. It is becoming recognised that advantage could be taken of multi-core to overcome the problem of network traffic rate superseding the rate at which NIDPSs can process incoming data [7].

This paper, which builds on our previous work [8], describes research which aims to solve the problem of dropped packets which can be a prevalent issue for NIDPSs used in heavy and high-speed traffic environments. Our research uses Snort IDS (Intrusion Detection System), in Network Intrusion Detection System (NIDS) mode. Snort is currently the most popular NIDPS software. Snort can be installed in any machine and runs on different operating systems such as Windows and Linux. Snort, which was introduced as a lightweight IDS, has developed significantly in the last 10 years. We conducted experiments to test Snort NIDS analysis performance under heavy and high-speed traffic. We also demonstrated that Snort NIDS performance can increase the number of analysed packets and decrease the number of dropped packets using alternative technologies such as a Quality of Service (QoS) configuration and parallel technology.

The remaining part of this paper is organised as follows: Section 2 gives a background about security mechanisms, stages and intrusion detection technologies. Section 3 explains our experimental design and implementation. Section 4 presents the results of the experiments and the evaluation. Then Section 5 gives an overview related work. Finally Section 6 concludes the paper and suggests recommendations and further work.

2. Background

2.1. Security mechanisms and approaches

Security is a major concern in every aspect of our daily life. New methods and equipment are constantly being devised to ensure protection. Computer networks continue to face many threats. We can consider three stages to achieving security in computer system networks: prevention, detection and correction. Prevention stops attacks before they enter system. Detection catches the attacks after they have entered and then Correction rectifies problems, which could be detected attacks or mistakenly prevented non-attacks. Prevention is the ideal solution, as compared to detection and correction, but it is impossible to prevent 100 percent of attacks [9]. Detection techniques provide results that can be used to prevent further attacks and aid correction. Thus the three stages combined offer an effective approach to achieving security. Common security mechanisms are firewalls, antivirus programs and intrusion detection and prevention systems.

2.1.1. Firewall

In order to secure a corporate network or sub-network, network traffic is usually filtered according to criteria such as origin, destination, protocol or service, typically through dedicated routers called firewalls. Firewalls are a common security defence and nowadays are treated as an integral part of every network. A firewall may be software or hardware; its functionality is based on filtering mechanisms specified by a set of rules, known as a policy, which can protect a system from flooding attacks. The fundamental function of a firewall is to sort packets according to allow/deny rules, based on header-filed information. The disadvantage of firewalls is that they cannot fully protect an internal network since they are unable to stop internal attacks. For example, malicious and unwanted web traffic can go through a firewall to strike and damage a protected computer system. A firewall is a set of rules such as to allow or deny protocols, ports or an IP address. Today's Denial of Service (DoS) attacks are too complex for firewalls because they cannot distinguish good traffic from DoS attack traffic [10]. The firewall provides the benefit of added security to strengthen a network when used in conjunction with an IDS.

2.1.2. Antivirus programs

Computer viruses are programs which cause computer failure and damage computer data. Especially in a network environment, a computer virus poses an immeasurable threat and can be very destructive. Antivirus programs are software that can be installed onto a computer in order to detect, prevent and make decisions regarding whether to quarantine or

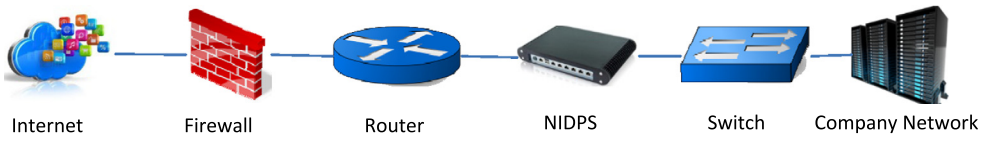

Fig. 2. An example of network-based IDPS.

delete malicious programs such as malware, worms or viruses. Although antivirus programs monitor the integrity of data files against illegal modifications, they are unable to block unwanted network traffic intended to damage the network. Antivirus software is installed only at explicit points of the servers, such as the interface between the network segment to be protected and outside environments [5].

2.1.3. Intrusion detection and prevention systems (IDPSs)

IDPS technologies detect and react to unauthorised access to network systems, providing real-time monitoring of network traffic. IDPSs can be software- or hardware-based, or can be a combination of both. Hardware-based IDPSs are effective for large organisations and companies, but are very expensive. However, software-based IDPSs running on the same devices or servers can identify and deal with attacks generated from inside or from outside the network, and can also protect the security policies of that network and their internal threats. Deploying a firewall with an IDPS is a useful way to provide extra security and thus strengthen the network [11]. Intrusion detection is one of the most tested and reliable technologies to monitor incoming and outgoing network traffic and to identify unauthorised usage and mishandling of computer system networks [12]. In addition, intrusion detection can identify the activity of malicious attackers. It is critical to implement intrusion detection and prevention in computer networks that have high traffic and high-speed connectivity [13].

A distinction is often made between intrusion detection systems (IDS) and intrusion prevention systems (IPS). This distinction is that the IDS detects intrusions and reports them, whereas the IPS detects, reports and prevents them through blocking. Therefore the IPS can be seen as an extension of the IDS. However nowadays the technologies have converged and most IDS systems cover prevention as well as detection. The mode of operation between detection and prevention may be selected via a configuration setting. Furthermore, the difference between a firewall and IPS can be indistinguishable to the user as the separate technologies are often combined to a single gateway sentry system. The firewall checks headers on packets and blocks depending on header information such as protocol type, source address, destination address, source port, and/or destination port according to network security policy. The IPS checks both headers and payload, blocking on recognisable known features according to network security policy. In our work we use Snort which can act as an IPS and IDS and can therefore be considered to be an IDPS. In this research the mode in which we ran Snort was Network Detection System (NIDS) mode.

IDPSs are still unable to control all threats and malicious activities [8,10,14]. To overcome the design and implementation difficulties, novel IDPS solutions are being sought in the context of multiple characteristics of advanced computer networks. These characteristics include: real time; high speeds and high loads; increasing difficulties for defenders; and decreasing difficulties for attackers. IDPSs need to be configured properly to achieve the desired output.

Many vendors are now trying to produce security appliances that can protect networks and which combine technologies. For example, the Cisco ASA 5500 series is a range of essential Cisco products that aims to secure an organisation's network from end to end [14]. The product comes in different sizes and has been a popular choice for network designers because of its high performance. The Cisco ASA 5500 Series integrates multiple full-featured, high-performance security services, including application-aware firewall, SSL (Secure Sockets Layer) and IPsec (Internet Protocol Security) VPN (Virtual Private Network), IPS with global correlation and guaranteed coverage, antivirus, antispam, antiphishing, and web filtering services.

2.2. Types of intrusion detection and prevention systems

Some of the existing types of IDPSs are: network-based (NIDPS); host-based (HIDPS); and graph-based IDS (GrIDPS). Hydrid systems also exist which combine one or more types into a single system [13].

2.2.1. Network-based IDPS

A Network Intrusion Detection System (NIDPS) is a common technique used to analyse traffic at all Open Systems Interconnection (OPI) layers by detecting the presence of normal traffic or suspicious activities [16].

To be effective a NIDPS must see the entire network and must be placed at an appropriate point in the network. In a hub-based network the NIDPS can be placed at the hub and can see all traffic but this is not possible in a switched network where there is no hub. In a switched network, port mirroring or spanning is used to enable a complete view but this causes overhead. The NIDPS itself is affected by DoS and DDoS attacks, similar to those made against IP gateways. Encrypted network traffic (packets) cannot be detected by the NIDPS (see Fig. 2).

2.2.2. Host-based IDPS

A Host Intrusion Detection System (HIDPS) is a software agent that can be installed in a particular computer in order to monitor and analyse events on that particular host to detect any suspicious behaviour [17].

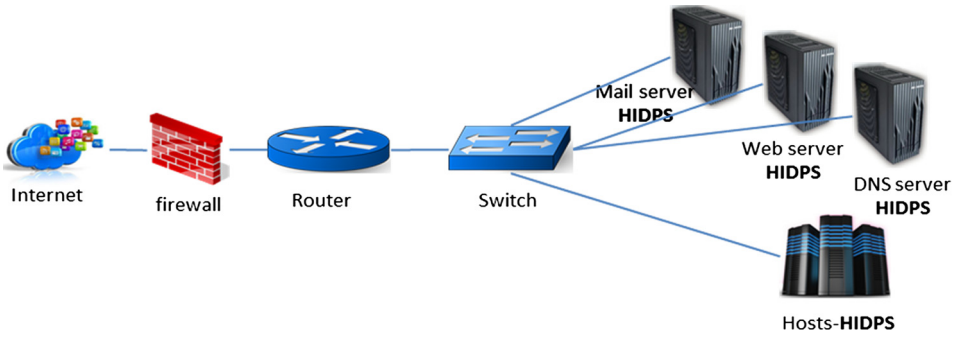

Fig. 3. An example of host-based IDPS.

HIDPSs are capable of integrating code analysis, monitoring system calls, detecting buffer overflows, privileging misuse, privileging abuse, file systems, library lists, applications, system configurations, system log analyses, and many others [11]. These things can be done by HIDPSs because they are designed to operate with a specific host and with respect to applications such as web servers, database servers, file servers, mail servers, and DNS servers. They are often integrated into server software and can be relatively easily implemented to communicate with other network components and operating systems. They can inspect encrypted traffic because they can analyse packets at the application ends. However HIDPSs have some disadvantages. They consume computer system resources that are needed for services and may conflict with existing security policies (such as firewalls) and operating systems. It is difficult for HIDPSs to analyse intrusion attempts on multiple computers. HIDPSs can be very difficult to maintain in large networks with different operating systems and configurations and can be disabled by attackers once a system is compromised. The HIDPS approach also requires many hosts to reboot after a complete installation or an update and many essential servers cannot support this operation (see Fig. 3).

2.2.3. Graph-based IDPS

Graph-based IDPSs (GrIDPSs) are designed to protect computer networks from large-scale malicious attacks, which severely affect computer networks. Network traffic and computers are linked through GrIDPs. The advantages of GrIDSs are that they can gather data about computer activity across a network and help to recognise comprehensive automated or coordinated attacks in real time. They allow network systems to state and implement policies specifying which users are permitted to utilise the particular services of an individual host or group of hosts. Assumptions made in this kind of system include the existence of related networks within a single organisation that has an independent infrastructure and sovereign departments. It is difficult to picture how this system would work to gain insight into the working of the GrIDS system in a modern and innovative enterprise environment where this kind of situation does not exist. It also assumes that no single component of the network is actively hostile, and therefore the IDPS must be designed to operate in non-hostile situations.

2.3. Network intrusion detection and prevention system methodology

The functional components of an integrated NIDPS are: events management; a data source; an analysis engine; and a response manager [14,18]. Events management gathers information on events to and from the monitored system (see Fig. 4). The data source is the event generator, which is classified into the following four categories: application-based monitors; host-based monitors; target-based monitors; and network-based monitors. The data source stores multiple events recorded by event management. The analysis engine collects data from the data source in order to analyse and determine whether the data is free of policy violations or other attacks. This engine can utilise anomaly/statistical detection, misuse/signature-based detection, or both. The analysis engine processes events and transmits alerts. The response manager neutralizes an attack once it is detected. The response manager responds to events and stops intrusions.

Most existing NIDPSs utilise either misuse detection or non-regular detection. The technique of misuse detection is employed to find known intrusions and/or a pattern of signatures. Due to its reliance on signatures, its detection speed is quite fast and has a low false positive rate. IDPS methodology is divided into the following four categories.

2.3.1. Misuse/signature-based detection

This type of detection system uses known signatures of malicious codes, which are stored in an IDPS database. Well-known patterns of attacks result from the use of malicious codes and known software vulnerabilities. This kind of detection system is highly efficient for use in a small NIDPS. The major drawback of such a system is that its database must be regularly updated, resulting in an ever-increasing database that must include as many available signatures as possible [14, 18].

2.3.2. Anomaly/statistical detection

Anomaly/statistical detection is a comparison-based method which compares any activity to the profile for all possible learned activities through statistical data, facts and figures. There are two types of profile, fixed and dynamic. A fixed profile

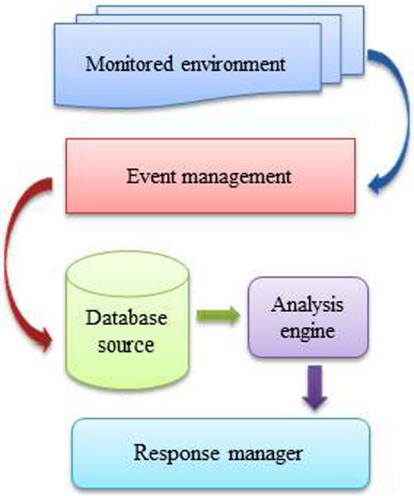

Fig. 4. General architecture for a NIDPS.

is the most efficient as compared to other schemes, because it terminates the occurrence of any unusual behaviour and it classifies the behaviour as anomalous. A dynamic profile cannot be created without an existing fixed profile; once the dynamic profile has been created, it allows the attacker to observe and alter his or her behaviour in long-term activities [14,18].

2.3.3. Protocol analysis detection

This NIDPS technique depends on the behaviour of the protocols. It observes the protocol behaviour and then compares it to those stored in its protocol behaviour database. It detects anomalies in the packet on the head part of the protocol. This technique is quite effective, but can be easily avoided by attackers working inside the protocol limitations [18].

2.3.4. Hybrid methodologies

This type of system combines two or more intrusion detection systems methodologies in order to analyse, detect and match any suspicious behaviour and signature malicious code that attempt to attack network. The power of combination means it can detect more types of intrusion, thereby providing relatively better results as compared to other methods.

3. Experiment design and implementation

3.1. Network design

A real network has served as a model for the purpose of analysis and data acquisition. We used several tools, including both software and hardware, to meet our objectives. Snort 2.9.4.6, which was issued in April 2013, was used as NIDS software; a WinPcap tool to capture packets on Windows 7 and 8 operating systems; a NetScanPro tool to manage a certain type of traffic in the network; a Packet Generator tool to generate/send network traffic of different values and speeds per ms; a Cisco Catalyst 3560 Series Switch [19], which supports QoS configuration; and a computer system consisting of four standard machines connected through VMware Virtual. Fig. 5 shows the network design for the experiment.

3.2. Snort component functions

Snort is an easy to use and popular open-source IDPS [6,20–22]. It is accessible free of cost and ranked among the top systems with the best features available nowadays. It was released as an open-source NIDS based on rule-based detection, which stored information in text files that could be modified by a text editor. Rules are grouped into categories, and the rules belonging to each category are stored as information in separate files, which are then integrated into the main configuration file named "snort.conf". The data is captured in terms of the described rules, which are read at the initialization of the Snort and comprise the internal data structure [6,22]. A Snort system consists of the following major components: a packet decoder; pre-processors; a detection engine; a logging and alerting system; and output modules.

The basic structure is represented in Fig. 6. When a packet arrives at the network, Snort listens and captures packets. In the beginning, the packet decoder receives packets from multiple network interfaces such as Point-to-Point Protocol (PPP) or Ethernet and Serial-Line-Internet-Protocol (SLIP), then pre-processes such packets ready for the detection engine [6,21]. The detection engine performs three main tasks: sniffing, analysis and detection. It can perform network traffic analysis and content searching/matching in both real-time and for forensic post-processing [21,22]. Snort can be configured in three

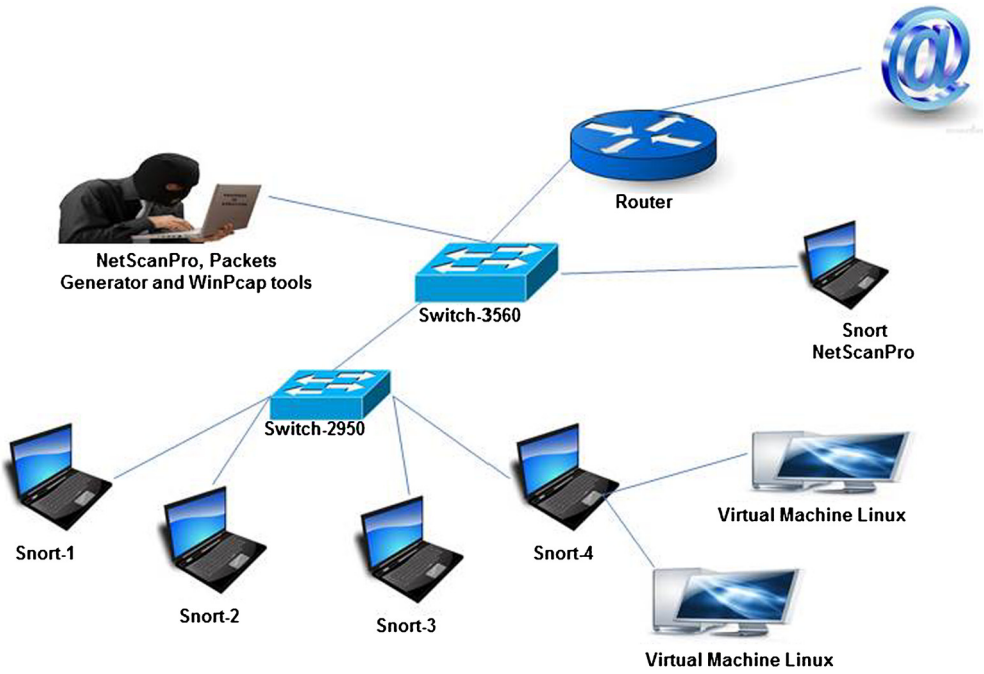

Fig. 5. Experiment network design.

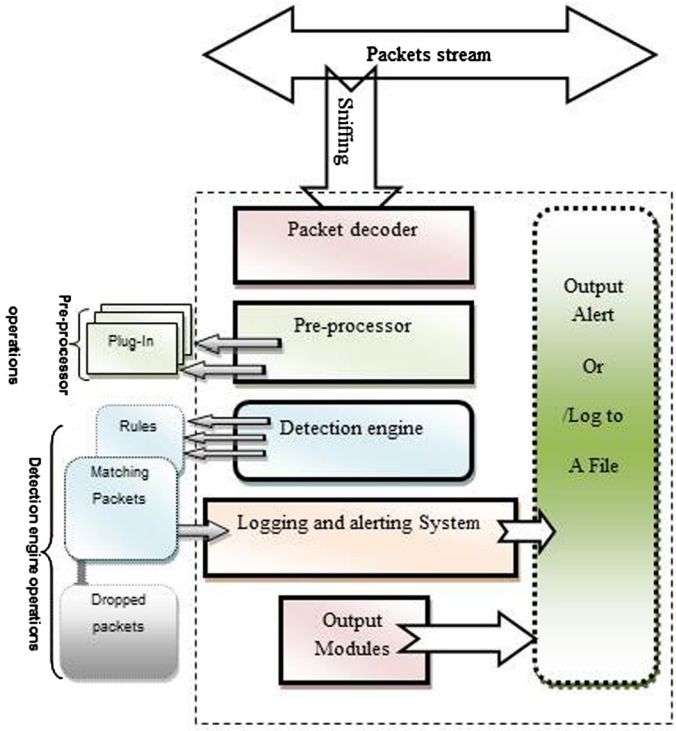

Fig. 6. Snort architecture [25].

main modes: sniffer, packet logger, and network intrusion detection. In network intrusion detection mode, Snort analyses the network traffic against a set of defined rules in order to detect intrusion threats. In our experiments, we focus on the Snort capability as a network intrusion detection system as we aim to see how many packets could be analysed by Snort under varying conditions. It has been shown previously that in high speed and heavy load conditions, packets are dropped

[4–6]. In our experiments Snort analyser is not set up to perform actions based on user defined rules. It simply analyses and identifies the packets.

The detection engine is time-critical and the most important part of the Snort. It utilises different processing times based on the length of the packet, the specifications of the system, and the number of rules defined in the system. Snort sometimes drops packets when it runs in real time NIDS mode, particularly when traffic is heavy and high-volume [6]. Snort rules are employed to detect intrusive actions to be presented in the data packet. In the detection mode, Snort is capable of reading chains (internal data structures), which have to be matched against all packets. If a packet does not match any rule, it will be blocked; otherwise appropriate action is taken [6]. Logging and alerts depend on the nature of what is detected inside the packets. If any suspicious activity is found inside a packet, the packet usually logs the malicious activity and/or generates an alert. Logs are usually stored in simple text-based files such as tcpdump style. Output modules (plug-ins) are capable of performing multiple operations depending on the results generated by the logging and alerting system of Snort. In general, output modules control the form of outcome produced by the logging and alerting system.

3.3. Cisco Catalyst 3560 Series switches

This category belongs to layer 2 and 3 switches. It provides support for IP-based software, for example Rate limiting, Access Control Lists (ACLs), QoS. IPv6, and advanced routing protocols. Policy and class enterprise features are supported by IP service software. Despite a packet's size and content, this switch provides the best effort services for each packet of network traffic. The packets are sent with no surety of delay bounds, reliability, or throughput [19].

3.4. Quality of service (QoS) technique

A QoS technique permits the control of traffic over a network and guarantees the throughput of traffic applications in terms of time scale. QoS concerns the performance of the network traffic over several technologies, including Asynchronous Transfer Mode (ATM), 802.1 networks, IP-routed networks, Frame relay, and Synchronous Optical Network (SONET) as seen from the user's perspective. Furthermore, QoS can use congestion avoidance and management techniques along with configuration of network traffic, and prioritizes traffic based on its importance [23]. QoS features can be classified into the following functions: classification and marking; policing; congestion management; and congestion avoidance.The features of QoS provide better and more reliable network services through the following features: support for dedicated bandwidth; improved loss characteristics; management and avoidance of network congestion; shaping network traffic; and setting traffic priorities across the network.

3.5. Performance metrics

Performance metrics are used in the experiments to measure the ability of the NIDS to perform a particular task and to fit within the performance constraints. These metrics measure and evaluate the parameters that impact NIDS performance. The following aspects were measured in the experiments.

3.5.1. Packet generation

The performance of TCP, UDP, and ICMP protocols was measured when running over the IPv4 header. The WinPcap and Packets Generator tool were used to vary the type of traffic in terms of IP header protocol (TCP, UDP and ICMP), speed, the number of packets and packet size.

3.5.2. Timing statistics

The Snort processor time includes total seconds and packets as well as packet processing rates.

3.5.3. Packets I/O totals

These are the percentages of the total packets processed by Snort. The specific metrics used are shown in Table 1.

3.5.4. Protocol statistics

All traffic for all protocols decoded by Snort are summarized in the Snort breakdown section which includes categories such as Eth (Ethernet interfaces); VLAN; IP4; Frag (Fragmented packages); ICMP; UDP; TCP and others.

3.5.5. Snort-NIDS throughput

This metric defines the level of traffic up to which the NIDS performs without dropping any packets. This metric is affected by the use of QoS configuration and parallel technology.

4. Experiment results and evaluation

The purposes of the experiments are:

Table 1
Snort performance metrics.

Performance metrics	Description
Packets received	The number of packets received by Snort.
Packets analysed	The number and percentage of packets analysed from the total packets received.
Packets dropped	The number and percentage of packets dropped from the total packets received.
Packets filtered	The number and percentage of packets filtered out and not handed to Snort for analysis
Packets outstanding	The number and percentage of the packets buffered waiting processing/or not processed.
Packets injected	The number of injected packets which are the result of active response, which can be configured for inline or passive modes.

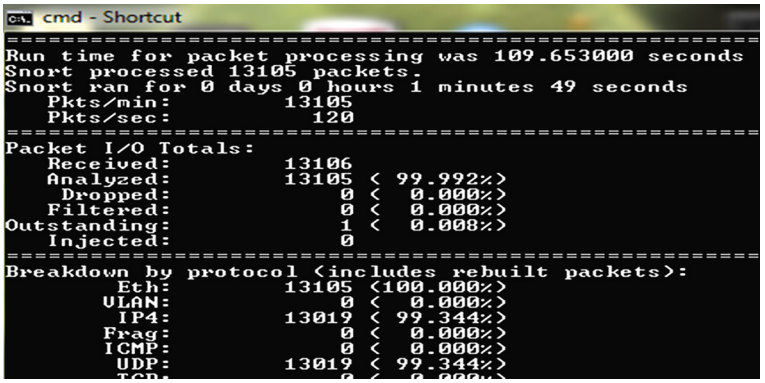

Fig. 7. Snort reaction to IP header at 8 ms transmission time intervals.

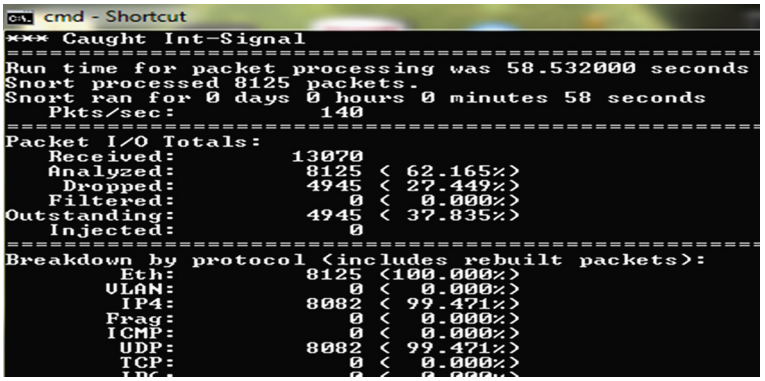

Fig. 8. Snort reaction to IP header at 4 ms transmission time intervals.

1) to show Snort-NIDS performance under (a) high speed traffic (Experiment 1), (b) heavy traffic (Experiment 2) and (c) large data traffic (Experiment 3).
2) to show how QoS configuration, which offers queue technology improves performance of Snort NIDS (Experiment 4).
3) to show how parallel technology and QoS improve performance of Snort NIDS (Experiment 5).

4.1. Experiment 1. Snort-NIDS reactions to high-speed network traffic

We used NetScanPro tools to manage IP traffic in the network and the packet generator tool to send a number of IP packets in different speeds per ms. We sent 13,000 packets (each packet carries 1 KB) at different time intervals (8 ms, 4 ms, 1 ms). Table 2 and Figs. 7, 8 and 9 show the Snort output and results of our experiments.

As demonstrated in the results shown in Figs. 7, 8, and 9, all the packets that were sent reached the wire. Snort analysed 99.992 percent of the packets in incoming traffic when packets were transmitted in 8 ms intervals (see Fig. 7), but when the speed of transmission was increased to 4 ms, Snort started dropping packets, analysing only 62 percent and dropping more than 22 percent of the total packets received (see Fig. 8). When the speed of transmission was increased to 1 ms intervals,

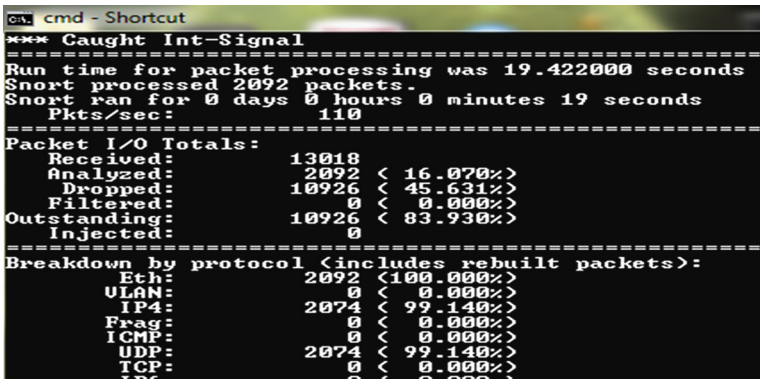

Fig. 9. Snort reaction to IP header at 1 ms transmission time intervals.

Table 2
Same number of packets but different speeds.

Packets sent (13,000)	8 ms interval	4 ms interval	1 ms interval
Packets received	100%	100%	100%
Packets analysed	99.992%	62.165%	16.070%
Packets dropped	0.00%	27.449%	45.631%
Packets filtered	0.00%	0.00%	0.00%
Packets outstanding	0.008%	37.835%	83.930%
Packets injected	0.00%	0.00%	0.00%

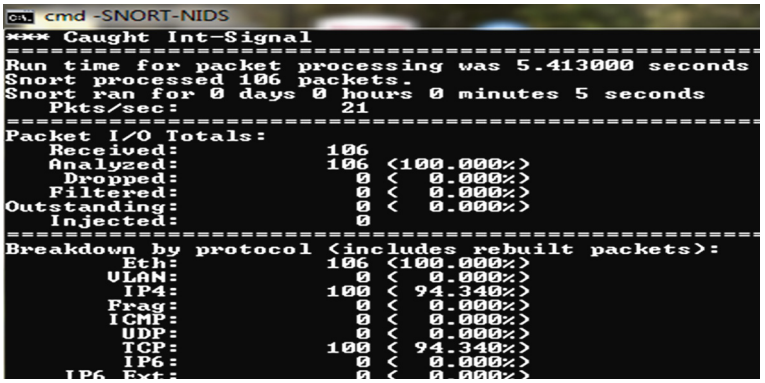

Fig. 10. Snort reaction to heavy traffic (100 KB packets).

Snort dropped more than 46 percent of packets (see Fig. 9). Our experiment demonstrated that Snort analysis performance was decreased as the speed of transmission increased.

4.2. Experiment 2. Snort-NIDS reactions to heavy-traffic networks

Here, the transmission rate of packets was kept to the same speed (1 ms intervals) to obtain a fair analysis of different numbers of packets (each packet carried 1 KB). We sent 100, 500 and 1000 packets batches at 1 ms intervals. Figs. 10, 11 and 12 show the Snort results (see Table 3).

As demonstrated by the results shown in Figs. 10, 11, and 12, all the packets that were sent reached the wire. In Fig. 10, when we sent 100 packets, Snort analysed 100% of the total packets that it received. As the number of packets increased to 500 and 1000, Snort started dropping packets (see Figs. 11 and 12). Our experiment shows that as the number of packets increases, more packets are dropped.

4.3. Experiment 3. Snort-NIDS reactions to large packets

For this experiment, the number of packets was kept to the same value (13,000) and the same speed (1 ms) to obtain a fair analysis of different sizes (lengths) of packets. We increased the size of each packet sent started from 1 byte, to 400 bytes, and to 800 bytes. Figs. 13, 14 and 15 show the Snort results.

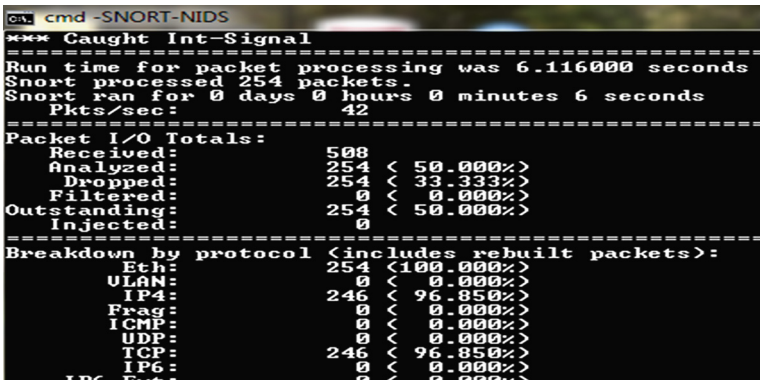

Fig. 11. Snort reaction to heavy traffic (500 KB packets).

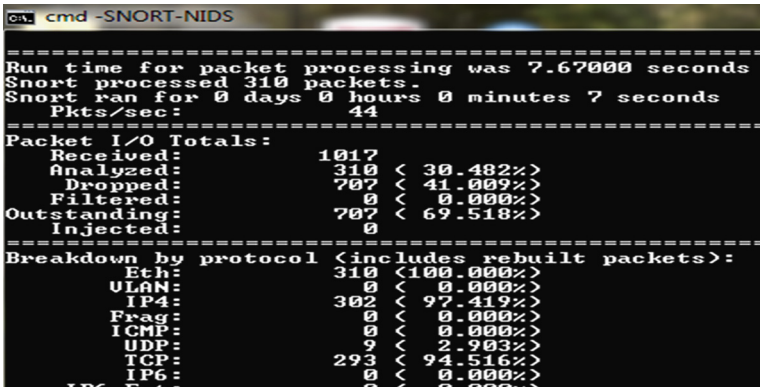

Fig. 12. Snort reaction to heavy traffic (1000 KB packets).

Table 3
Same speed limit and different numbers of packets.

Number of packets (Transmission interval 1 ms)	100	500	1000
Packets received	100%	100%	100%
Packets analysed	100%	50.000%	30.482%
Packets dropped	0.00%	33.333%	41.009%
Packets filtered	0.00%	0.00%	0.00%
Packets outstanding	0.00%	50.000%	69.518%
Packets injected	0.00%	0.00%	0.00%

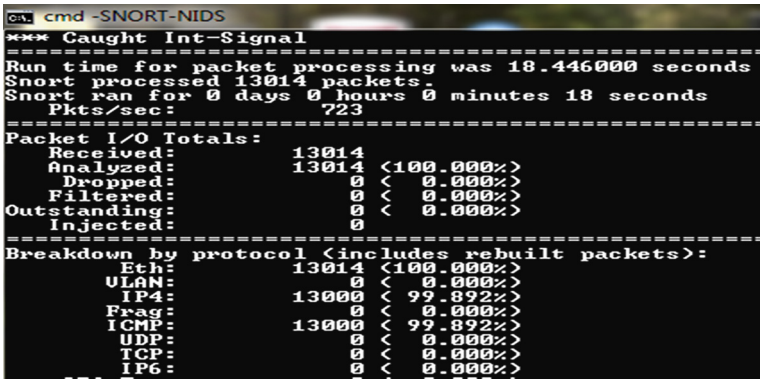

Fig. 13. Snort reaction to packet sizes (1 byte).

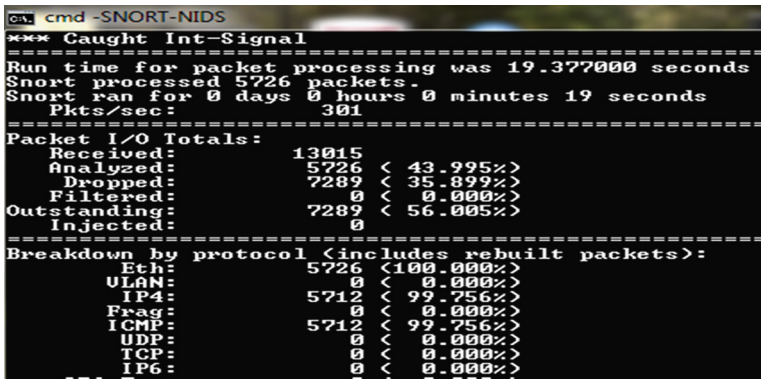

Fig. 14. Snort reaction to packet sizes (400 bytes).

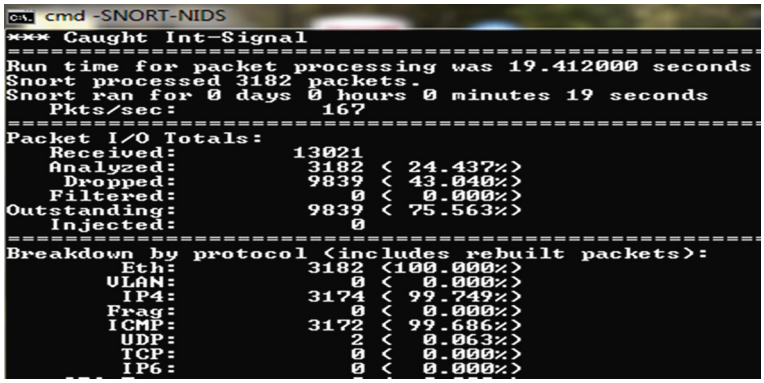

Fig. 15. Snort reaction to packet sizes (800 bytes).

Table 4
Same speed and value but different packet size.

Packet Number (speed 13,000 per 1 ms)	1 byte	400 bytes	800 bytes
Packets received	100%	100%	100%
Packets analysed	100%	43.995%	24.437%
Packets dropped	0.00%	35.899%	43.040%
Packets filtered	0.00%	0.00%	0.00%
Packets outstanding	0.00%	56.005%	75.563%
Packets injected	0.00%	0.00%	0.00%

As shown in Figs. 13, 14 and 15, when we sent 13,000 packets at 1 ms intervals (each packet carries 1 byte), Snort analysed 100 percent of the total packets received (see Fig. 13). As the size of the packets was increased to 400 bytes Snort dropped more than 35 percent of them (see Fig. 14), and when the packet size was increased to 800 bytes (each packet carries 800 bytes), Snort accordingly dropped more (see Fig. 15). Our experiment demonstrated that more packets will be dropped as packet size increases (see Table 4).

4.4. Experiment 4. Snort-NIDS using QoS configuration technology in high-speed traffic

Critical analyses were done for experiments 1, 2 and 3 (see Figs. 16, 17 and 18 respectively). The figures show that Snort performance analysis throughput is affected by high-speed and heavy traffic, and more packets are dropped as the number and size of packets and the speed of traffic increases. Snort has a limited time to process and analyse any traffic successfully and if a network's traffic speed limit is higher than Snort's limit, Snort will drop packets.

To solve this problem, we used a Cisco Catalyst 3560 Series switch, which supports QoS configuration, to load the traffic into a number of interfaces equally and divide traffic into streams in order to analyse each portion of traffic individually to determine whether it was free of malicious codes. We configured Snort and QoS to reorder and control traffic speed such that it is similar to processor time and load traffic speed.

In the IEEE 802.1 network protocol, the Class of Service (CoS) parameter which resides at layer 2, enables differentiation of the packet. This value can then be used to provide differentiated services for different types of packet. Other QoS

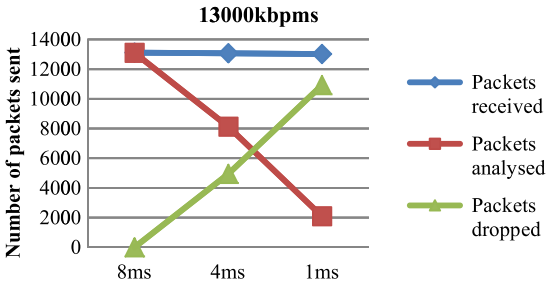

Fig. 16. Snort reactions to IP headers with increasing traffic speeds.

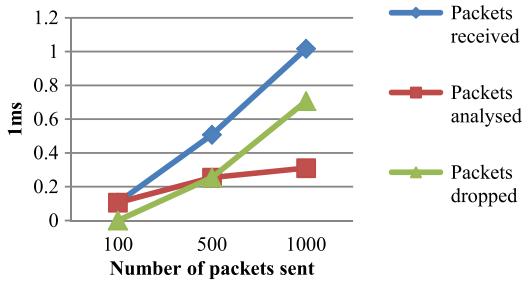

Fig. 17. Snort reactions to IP headers with increasing traffic values.

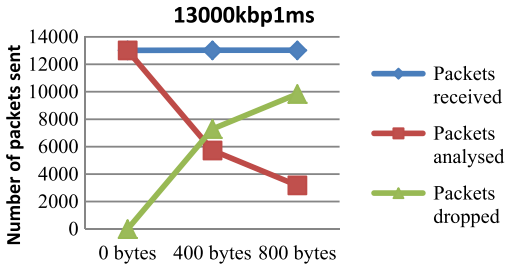

Fig. 18. Snort reactions to IP headers with increasing packet sizes.

mechanisms operate at layer 3, for instance DiffServ (Differentiated Services) which allow different types of service to be offered depending on a code. For instance there could be a policy to give a certain type of package priority. To implement QoS based on the DiffServ architecture, which specifies that each packet be classified upon entry into the network and adjusted for different traffic speeds, we changed/classified the switch frame to the default working from layer 2 to layer 3 by mapping the traffic values from CoS to Differentiated Services Code Point (DSCP) values. Fig. 19 illustrates the relative layers at which CoS and DSCP operate. The CoS values are from 0–7 and the DSCP values are from 0–63. To distinguish between packet classes, a class map and policy map functions were used to classify traffic inside the switch [23]. Classification is the process of distinguishing one kind of traffic from another by examining fields in the packet. Classification occurs on the physical interface or on a per-port, per-VLAN basis. Policing involves creating a policy that specifies the bandwidth limits for the traffic and applies it to the interface. Policing can be applied to a packet per direction and can occur on the ingress and egress interfaces.

One of the mechanisms that QoS offers is queue technology, which can give a switch a new logical throughput-traffic-forwarding plan [23,24]. QoS offers two input queues (ingress queues) and four output queues (egress queues) at the physical output interfaces (ports and VLANs) [23,24]. As shown in Fig. 20, we configured the switch to two input queues and four output queues, and each input queue has a policy (policy map) and marking (class map). We configured the bandwidth, threshold and priority for each input and output queue to treat traffic in the input and output queues. We also configured the speed limit for each ingress queue and egress queue using one of two functions inside the switch called Shaped or Share Round Robin (SRR) [23,24]. The Shaped function is only available on egress queues, and a queue reserves only a portion of a port's bandwidth. SRR is available on both ingress and egress queues. It guarantees a queue a portion of a port's bandwidth, but does not limit the queue to that guaranteed amount. The main idea here is to allocate a specific traffic weight and speed limit for each queue, which allows a number of packets to be sent at specific time intervals, thereby reducing traffic congestion even if the traffic is high-speed and heavy.

We sent 13000 packets (each packet carries 1 KB) at 1 ms intervals to the network. As the results shown in Fig. 21 demonstrate, Snort analysed 100 percent of the traffic that reached the wire with 0 percent dropped. Experiment 1 showed

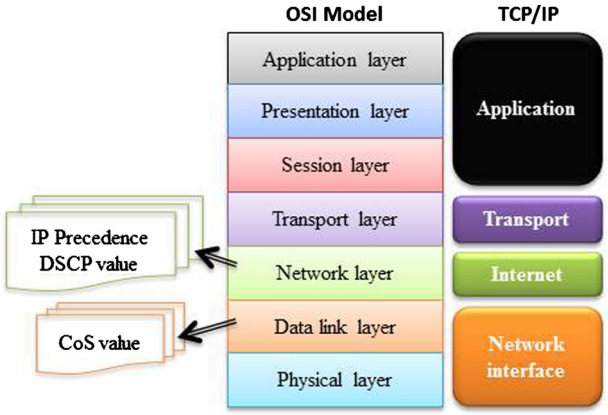

Fig. 19. Layer for CoS and DSCP values.

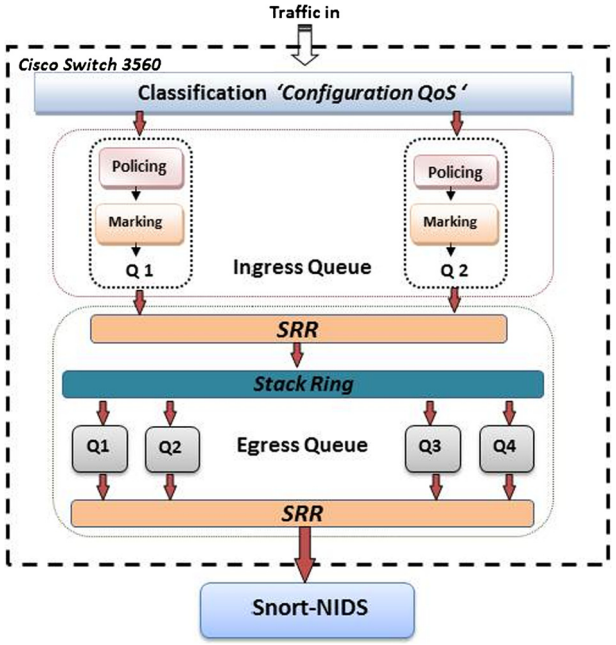

Fig. 20. Snort with QoS architecture.

that 45% of packets were dropped (see Fig. 9) when this QoS configuration was not used. The results show that Snort performance analyses are significantly improved when using QoS technology.

4.5. Experiment 5. Parallel Snort-NIDS with QoS technology in high-speed network traffic

In experiment 1, Snort dropped more than 45 percent (see Fig. 9). When we used Snort with QoS in experiment 4, Snort dropped 0 percent (see Fig. 21). However, the difference between experiments 1 and 4 is Snort's processor times, which were 33 s and 101 s respectively (see Fig. 22).

As a solution to reduce Snort's processor time, we suggest using parallel NIDS technology with QoS to increase NIDS performance analysis and decrease processor time. As we show in Fig. 23, we configured and treated traffic using QoS configuration, which produced four output queues. Then each queue was scanned using an access list function (ACL) which filters traffic according to classification and enabled different packages to be sent to specific Snorts. Each package was directed to a parallel Snort. Using an ACL and QoS configuration, you can analyse and classify separate types of traffic and send these to separate queues. In this experiment we increased the number of packets sent to 40,000. Each packet was 1 KB in size and the interval between each transmission was 1 ms.

As the results shown in Figs. 24 and 25 demonstrate, when we tested Snort as normal without any traffic treatment, we sent nearly 40,000 packets in 1 ms, Snort analysed 16 percent of the total packets received in 55 s, but when we used a single Snort NIDS with a QoS configuration and sent the same packets Snort analysed all the packets that reached the

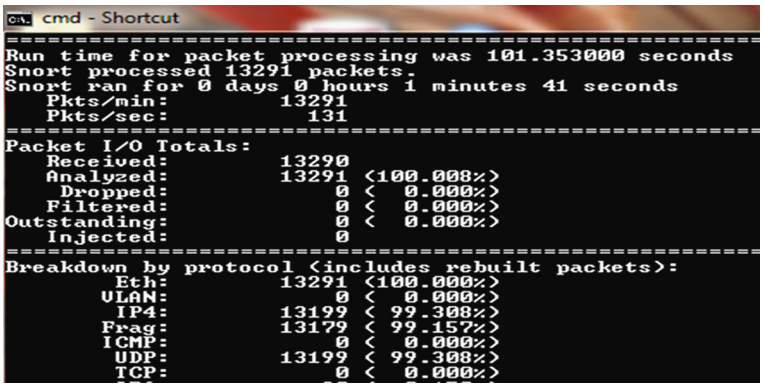

Fig. 21. Snort with QoS reactions to an IP header in high-speed and heavy traffic networks.

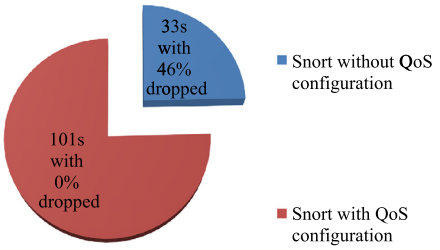

Fig. 22. Snort processor time (13,000 1 KB packets, at 1 ms intervals).

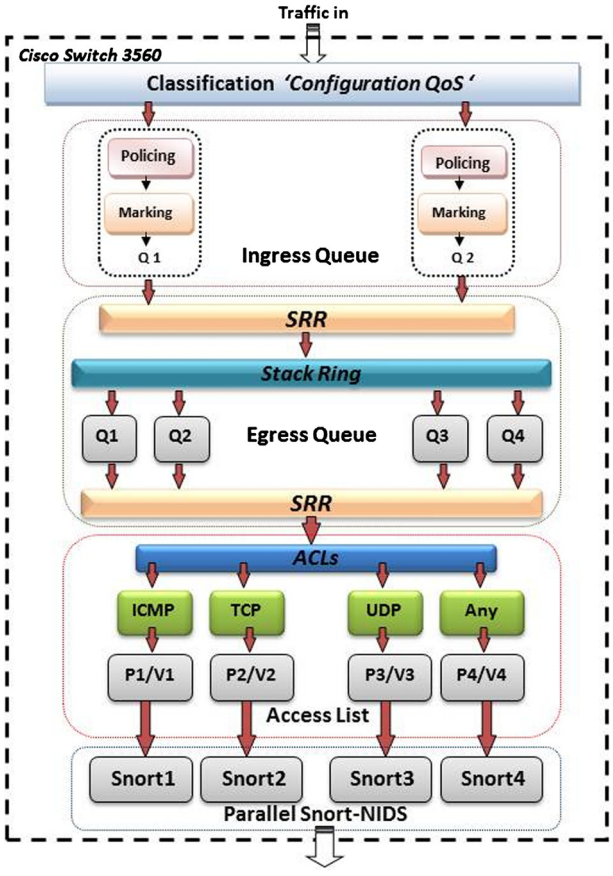

Fig. 23. Architecture for parallel Snort-NIDS using QoS and ACLs.

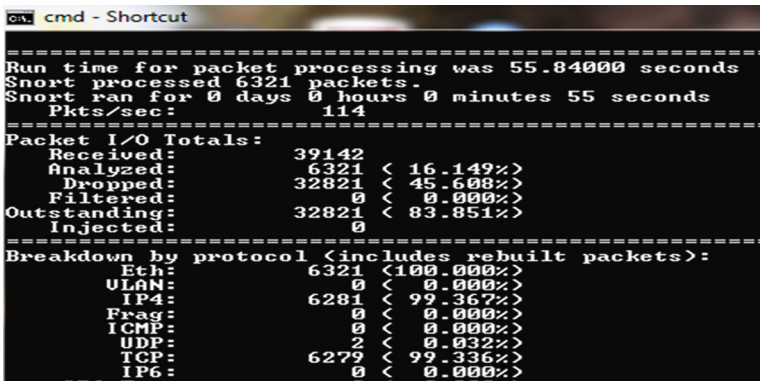

Fig. 24. Snort without QoS.

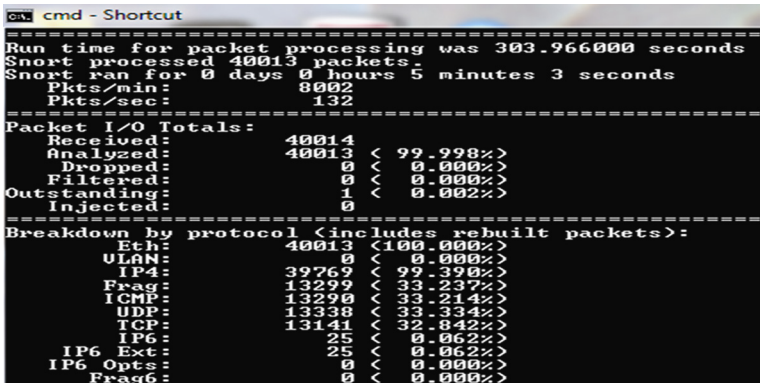

Fig. 25. Snort with QoS.

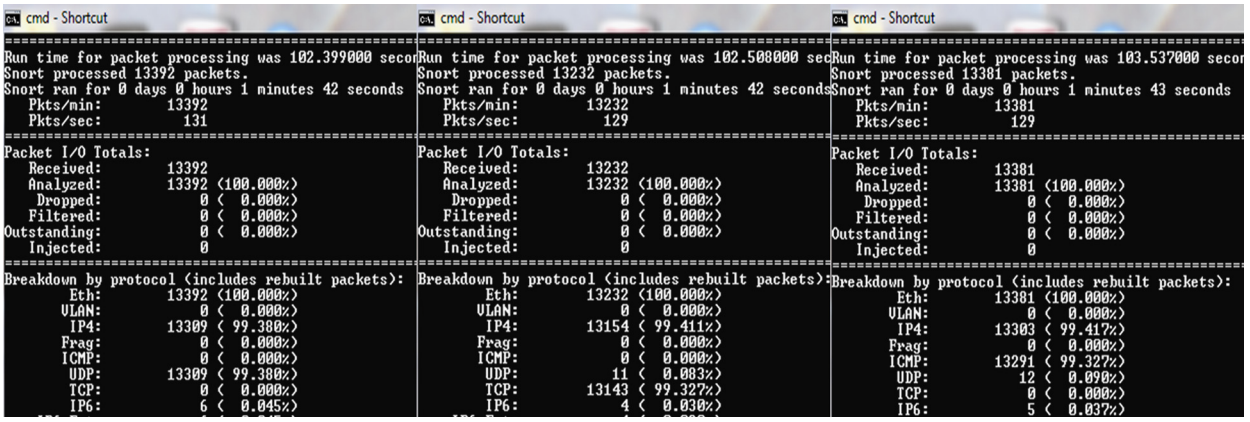

Fig. 26. Paralell Snort with QoS.

wire in 302 s without dropping any. Using parallel NIDS technology (in three queues), Snort analysed 100 percent of the packets in less time (103 s), showing an improvement of around 60% or roughly 3 times speed up. Our experiments prove that Snort performance analysis improves significantly using QoS and parallel NIDs technology. It has processed more than 40,000 KB in 103 s with 0 percent dropped (see Figs. 26, 27). Obviously speed up depends on the number of processors used and far greater speed up is possible.

4.6. Summary of experiments

Experiments 1 to 3 have shown that Snort drops packets in heavy and high speed traffic. In experiment 4 we show how QoS configuration within the Cisco Catalyst 3560 Series switch can enhance performance such that packets are no

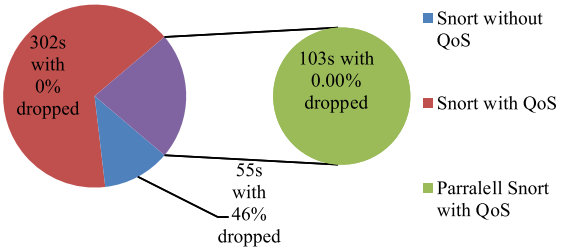

Fig. 27. Snort processor time for 40,000 KB sending in 1 ms.

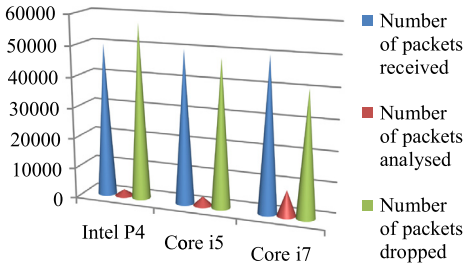

Fig. 28. Snort used with different processors.

longer dropped. However Snort takes longer to run. Experiment 5 shows how parallel technology can be used to speed up processing.

In the Snort outputs associated with our experiments (Figs. 7, 8, 9, 10, 11, 12, 14, 15, 21, 24, 25 and 26) we see that the number of packets dropped is the same as the number of outstanding packets. This is because in our context both of these metrics refer to the number of packets not seen by Snort. However we notice there is a discrepancy in the corresponding percentages output, a result of the internal Snort processing. We have taken the lower percentage associated with the dropped packets in our analysis. Had we taken the higher percentage associated with outstanding packets, the gains in performance and speed up through using our method would have been even greater.

It is important to note that when you test Snort with a QoS configuration under different processors, the QoS configuration will be different depending on which type of processor is running Snort, specifically its speed (see Fig. 28). We tested Snort at the same speed and for the same values, but with different processors: the Intel Pentium® D CPU 2.2 GHz (Intel P4), the Intel® corei5 2.27 GHz (Core i5) and the Intel® corei7 2.40 GHz (Core i7). Snort performance analysis was affected. It performed better with the Intel® core i7 processor than the others.

5. Related work

Due to the fact that numerous computer systems are unable to detect or prevent threats such as DoS/DDoS attacks, the impacts of these kinds of attacks are immeasurable and irremediable [8]. Such attackers amend, steal and destroy valuable information, and at worst damage a victim's computer system. Their main purpose is to stop or slow down the performance of legitimate users' computer network systems by exploiting vulnerabilities such as mis-configuration and software bugs generated from internal and external networks. Despite the existence of a variety of security protections, attackers often attempt to render services merely unavailable to intended legitimate users [25]. Here, it is insufficient to depend only on prevention techniques, especially when an attacker has successfully obtained vulnerable information from the network.

Some researchers [5,10,13,26] have investigated specific technologies and methodologies to detect network attacks that occur in real time, using open source IDSs, like Snort, working in high-speed, heavy-traffic networks, while others [27,28] have used comparisons between IDS tools to achieve the best throughput results with different IDSs. Our work addresses performance in a different way. It explores the use of parallel technology with network switch configuration to improve QoS and hence NIDS performance.

Salah and Qahtan [29] implemented a hybrid scheme in Linux OS to prove that a hybrid scheme can improve the performance of general-purpose network desktops or servers running network I/O-band applications when such network hosts were exposed to both light and heavy traffic load conditions. The standard on subscribed configurations of Linux networking subsystems, as revealed by Salah and Qahtan, failed to meet Snort's performance level. In order to achieve a high throughput of analysed traffic with Snort, they tuned the budget parameter of the Linux Network subsystem, which controls the utilisation time of the central processing unit cycle. Our work is similar to this in that it explores configuration in a general purpose environment but it is different in that it explores configuration at a multi-layer switch level. Thus it is network-based rather than the host-based.

Shiri, Shanmugam and Idris [5] proposed a parallel technique for improving the performance of a signature-based NIDS. Their idea was to send different types of packets to different parallel Snorts for analysis and they obtained a 40% improve-

ment in processing time. Schuff, Choe and Pai [30] proposed a multi-thread Snort called MultiSnort which executes multiple instances of the original Snort in parallel. Our research is similar to these in that we also send different types of packet to parallel Snorts. However, while our work confirms the findings of previous research, the main difference is that we have provided detail on how to achieve the improvement through QoS and parallelisation using industry standard software systems. Another difference is that we have concentrated on analysis and also provided further experiments with greater detail of influential parameters. Chen et al. [31] presented Para-Snort which revised the structure of the original Snort decoupling the decoding part so that this activity is carried out centrally before the parallel queues are formed. The approach also used central load balancing to distribute packets to parallel Snort processing units. Our work differs from Chen et al.'s work in that we parallelise the whole of Snort, forming the parallel queues in the switch before sending packets to Snort pre-processing and decoding. The problem with central decoding, pre-processing and load balancing modules is that they could become additional bottle necks in the system. Chen et al. also researched how to reduce the load balancing bottle neck issue.

Vasiliadis, Polychronakis, and Ioannidis [32] proposed a new model for a multi-parallel IDS architecture (MIDeA) for high-performance processing and stateful analysis of network traffic. Their solution offers parallelism at a subcomponent level, with NICs, CPUs and GPUs doing specialised tasks to improve scalability and running time. They showed that processing speeds can reach up to 5.2 Gbps with zero packet loss in a multi-processor system. Their solution offers parallelism at a subcomponent level, with NICs, CPUs and GPUs doing specialised tasks. Jiang et al. [7] proposed a parallel design for NIDS on a TILERAGX36 many-core processor. They explored data and pipeline parallelism and optimized the architecture by exploiting existing features of TILERAGX36 to break the bottlenecks in the parallel design. They achieved throughput of 11 Gbps. Jamshed et al. presented Kargus [33], a system which exploits high processing parallelism by balancing the pattern matching workloads with multi-core CPUs and heterogeneous GPUs. Kargus adapts its resource usage depending on the input rate, to save power. The research shows that Kargus handles up to 33 Gbps of normal traffic and achieves 9 to 10 Gbps even when all packets contain attack signatures. The various approaches described in this paragraph are not directly comparable in terms of throughput as different numbers of processors and data is used in each. However the experiments show what can be gained by parallelising NIDPSs in order to combat problems of higher speeds and increasing traffic. Our work differs from the work described in this paragraph in the architecture used. We have shown how QoS technology and parallelism can have impact in high speed and heavy traffic network using an industry standard switch and standard desktop processors. We believe, our solution is a more accessible way of receiving good results as it can be activated at a higher level, namely at the level of configuring the CISCO switch software and replicating Snort on standard machines. Further improvements could be made if higher performance equipment was used. However we believe that there is room for various approaches and more exploration of suitability of various methods in varying circumstances. In this way users can apply the solution that is most suitable for them.

In the context of big data and distributed systems, Zhao et al. [34] have developed a security framework in G-Hadoop. This work focuses on authentication and access rather than intrusion detection but offers an interesting new direction. The framework could be enhanced with intrusion detection and protection functionality to create a more complete solution. Our work has focused on standard business infrastructure and other work has concentrated on single cluster high-performance infrastructure. Cross-cluster security services in a high performance environment such as that afforded by G-Hadoop is an area where attention is welcome. Interesting work is being carried out in the classification of internet traffic which can be used to support attack detection. In order to counteract limitations of current internet traffic classification techniques, which are based only on header and payload inspection, Wang et al. [35] have developed a system which can classify traffic in terms of their intended application by considering packet and flow characteristics. A machine learning approach has been used to develop the classifier. Extra complexity introduced by more demanding rules, albeit with the purpose of producing better detection performance, supports our contention for parallel NIDS in high-speed and heavy traffic environments.

Vendor companies are aiming to develop security solutions to protect the enterprise network. Equipment has been designed to meet connectivity speed and load standards. The improvements in the throughput of NIDS we have shown are achieved by pairing the ASA Cisco equipment with Snort [15,19,20]. The principles of our work could be applied to other equipment combinations where similar facilities are offered.

6. Conclusion, recommendations and further research

6.1. Conclusion

For many years attacks made on networks have risen dramatically. The major reason for this is the unlimited access to and use of software (written and uploaded to websites by technical experts) by inadequately trained people. Network disruptions may be caused intentionally by several types of directed attack. These attacks are made at various layers in the TCP/IP protocol suite, including the application layer. Besides the external body, attacks can be made on the network by the internal body as well. However, an IDPS is considered to be one of the best technologies to detect threats and attacks. NIDPSs have attracted the interest of many organisations and governments, and any Internet user can deploy them. An NIDPS usually features four stages to secure a computer system network: scanning, analysing, detecting, and correcting. Our paper focused on NIDPS weakness in scanning and analysing in high-speed network connectivity. We suggest using QoS

configuration to improve NIDPS analysis performance and a parallel technology to reduce NIDPS processing time. As a result of our approach, systems can be configured such that attacks can be thwarted more easily.

6.2. Recommendation

There is much yet to be learned about QoS technology. Some features of QoS may boost NIDPS performance, such as congestion management and congestion avoidance. Congestion management is balanced queuing, which evaluates the internal DSCP and determines which of the four egress queues in which to place the packets. There are many items to configure when it comes to queuing: defining the priority queue, defining a queue set, guaranteeing buffer availability, limiting memory allocation, specifying buffer allocation, setting drop thresholds, mapping CoS to DSCP value to queue, configuring SRR, and limiting bandwidth on an outbound queue. A lot of things in congestion avoidance may help with NIDS performance, such as setting output queuing, configuring Weighted Tail Drop (WTD) parameters to a four-queue set, WTD thresholds for a queue, guaranteed buffer availability for a queue's maximum memory and allocation of a queue buffer for all output queues of an interface. We recommend that QoS technology is exploited to achieve better detection and protection. We also recommend the use of parallel technology.

6.3. Further research

This paper centred on the failure of NIDPSs to prevent attacks that occur in high speed network connectively. It described experiments which presented the weakness of NIDPSs and which improved NIDPSs in terms of performance, efficiency and effectiveness. Multi-core could be used as a solution for high-speed data and network connectively. Multi-core processors provide enhancement with high capabilities and can secure networks from attacks, but they increase the complexity of the security system. Advances in the utilisation of multi-core processors for intrusion detection have yet to be fully exploited. However, there are two major areas of concern in computer security: the speed and volume of attacks; and the complexity of multi-stage attacks. By using multi-core processors appropriately, we can advance NIDPSs to cope with such concerns. In the area of development of the NIDPS detection function, intelligent techniques can be exploited to develop new rules for more precise detection of attacks to counteract the growth in diversity and deviousness. The current and anticipated future demands for online security require the revision of existing systems towards the development of improved parallel systems as well as stronger rule sets.

References

[1] J. Jang-Jaccard, S. Nepal, A survey of emerging threats in cybersecurity, J. Comput. Syst. Sci. 80 (5) (2014) 973–993.
[2] A. Fuchsberger, Intrusion detection systems and intrusion prevention systems, Inf. Secur. Tech. Rep. 10 (2005) 134–139, http://www.sciencedirect.com.lcproxy.shu.ac.uk/science/article/pii/S1363412705000415. Accessed September 14, 2013.
[3] Arbor Networks, 9th annual worldwide infrastructure security report and ATLAS data, http://www.arbornetworks.com/resources/infrastructure-security-report, 2013. Accessed March 25, 2014.
[4] E. Albin, N.C. Rowe, A realistic experimental comparison of the Suricata and Snort intrusion-detection systems, in: Workshops of the 26th International Conference on Advanced Information Networking and Applications, WAINA, IEEE, 2012, pp. 122–127.
[5] F.I. Shiri, B. Shanmugam, N.B. Idris, A parallel technique for improving the performance of signature-based network intrusion detection system, in: Proceedings of 3rd International Conference Communication Software and Networks, ICCSN, IEEE, 2011, pp. 692–696.
[6] J. Beale, B. Caswell, T. Kohlenberg, M. Poor, Snort 2.1 Intrusion Detection, second ed., Syngress Publishing, 2004.
[7] H. Jiang, G. Zhang, G. Xie, K. Salamatian, L. Mathy, Scalable high-performance parallel design for network intrusion detection systems on many-core processors, in: Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems, IEEE, pp. 137–146.
[8] W. Bul'ajoul, A. James, M. Pannu, Network intrusion detection systems in high-speed traffic in computer networks, in: Proceedings of 10th International Conference on e-Business Engineering, ICEBE, IEEE, 2013, pp. 168–175.
[9] G.M. Nazer, A.A.L. Selvakumar, Current intrusion detection techniques in information technology—a detailed analysis, Eur. J. Sci. Res. 65 (2011) 611–624.
[10] S. Beg, U. Naru, M. Ashraf, S. Moshin, Feasibility of intrusion detection system with high performance computing: a survey, Int. J. Adv. Comput. Sci. 1 (2010) 26–35.
[11] K. Scarfone, P. Mell, Guide to Intrusion Detection and Prevention Systems (IDPS), Recommendations of the National Institute of Standards and Technology, NIST Special Publication 800-94, 2012.
[12] A. Patcha, J. Park, An overview of anomaly detection techniques: existing solutions and latest technological trends, Comput. Netw. 51 (2007) 3448–3470.
[13] W. Jiang, H. Song, Y. Dai, Real-time intrusion detection for high-speed networks, Comput. Secur. 24 (2005) 287–294.
[14] D. Mudzingwa, R. Agrawal, A study of methodologies used in intrusion detection and prevention system (IDPS), in: Proceedings of IEEE Southeastcon, 2012, pp. 1–6.
[15] Cisco Systems, Cisco ASA 5585-X adaptive security appliance, n.d. http://www.cisco.com/c/en/us/products/collateral/security/asa-5500-series-next-generation-firewalls/product_data_sheet0900aecd802930c5.html, 2014. Accessed July 17.
[16] T.M. Wu, Intrusion Detection Systems, Intrusion Assurance Technology Analysis Center, Herndon, VA, sixth ed., 2009, http://iac.dtic.mil/csiac/download/intrusion_detection.pdf, 2009. Accessed September 15, 2013.
[17] H. Kozushko, Intrusion detection: host-based and network-based intrusion detection systems, Independent study, 2003.
[18] M.S. Hoque, M.A. Mukit, M.A. Bikas, An implementation of intrusion detection system using genetic algorithm, Int. J. Netw. Secur. Appl. IV (2012) 111–112.
[19] Cisco Systems, Cisco Catalyst 3560 Series switches, n.d. http://www.cisco.com/en/US/products/hw/switches/ps5528/, 2013. Accessed October 11.
[20] M. Roesch, M. Snort, Lightweight intrusion detection for networks, in: Proceedings of the Conference on Large Installation System Administration, LISA, vol. 99, 1999, pp. 229–238.
[21] R.U. Rahman, Intrusion Detection System with Snort: Advanced IDS Techniques with Snort, Apache, PHP, MySQL and ACID, Pearson Education and Prentice Hall Professional, 2003.

[22] R. Chi, Intrusion detection system based on Snort, in: Proceedings of the 9th International Symposium on Linear Drives for Industry Applications, vol. 3, Springer, Heidelberg, Berlin, 2014, pp. 657–664.
[23] Cisco Systems, Quality of service design overview, n.d. http://www.cisco.com/en/US/docs/solutions/Enterprise/WAN_and_MAN/QoS_SRND/QoSIntro.html, 2013. Accessed October 20.
[24] Cisco Systems, Understanding queuing with hierarchical queuing framework, http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6558/sol_ov_c22-708224.html. Accessed November 15, 2012.
[25] H.J. Kim, A. Pamnami, M. Patel, State-of-the-Art in Intrusion Detection Systems, Department of Electrical and Computer Engineering Stevens Institute of Technology, Hoboken, 2007.
[26] M. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, K.S. Park Kargus, A highly-scalable software-based intrusion detection system, in: Proceedings of the 2012 ACM Conference on Computer and Communications Security, ACM, 2012, pp. 317–328.
[27] K.R. Karthikeyan, A. Indra, Intrusion detection tools and techniques–a survey, Int. J. Comput. Theory Eng. 2 (2010) 901–906.
[28] P. Mehra, A brief study and comparison of Snort and Bro Open source network intrusion detection system, Int. J. Adv. Res. Comput. Commun. Eng. 1 (6) (2012) 383–386.
[29] K. Salah, A. Qahtan, Implementation and experimental performance evaluation of a hybrid interrupt-handling scheme, Comput. Commun. 32 (1) (2009) 179–188.
[30] D.L. Schuff, Y.R. Choe, V.S. Pai, Conservative vs. optimistic parallelization of stateful network intrusion detection, in: Proceedings of the 12th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, 2007.
[31] X. Chen, Y. Wu, L. Xu, Y. Xue, J. Li, Para-Snort: a multi-thread Snort on multi-core IA platform, in: Proceedings of Parallel and Distributed Computing and Systems, PDCS, 2009.
[32] M. Vasiliadis, S. Polychronakis, S. Ioannidis, MIDeA: a multi-parallel intrusion detection architecture, in: Proceedings of the 18th ACM Conference on Computer and Communications Security, ACM, 2011, pp. 297–308.
[33] M. Jamshed, J. Lee, S. Moon, I. Yun, D. Kim, S. Lee, Y. Yi, K. Park, Kargus: a highly-scalable software-based intrusion detection system, in: Proceedings of the ACM Conference on Computer and Communications Security, CCS, 2012.
[34] J. Zhao, L. Wang, J. Tao, J. Chen, W. Sun, R. Ranjan, J. Kolodziej, A. Streit, D. Georgakopoulos, A security framework in G-Hadoop for big data computing across distributed Cloud data centres, J. Comput. Syst. Sci. 80 (5) (2014) 994–1007.
[35] Y. Wang, Y. Xiang, J. Zhang, W. Zhou, B. Xie, Internet traffic clustering with side information, J. Comput. Syst. Sci. 80 (5) (2014) 1021–1036.