Network intrusion detection systems in high-speed traffic in computer networks

Waleed Bul’ajoul  
Faculty of Engineering and Computing  
Coventry University  
Coventry, UK  
Bulajouw@coventry.ac.uk

Anne James  
Faculty of Engineering and Computing  
Coventry University  
Coventry, UK  
Csx118@coventry.ac.uk

Mandep Panu  
Faculty of Engineering and Computing  
Coventry University  
Coventry, UK  
Aa3371@coventry.ac.uk

Abstract—With the various and increasingly malicious attacks on networks and wireless systems, traditional security tools such as anti-virus programs and firewalls are not sufficient to provide free, integrated, reliable and secure networks. Intrusion detection systems (IDSs) are one of the most tested and reliable technologies to monitor incoming and outgoing network traffic to identify unauthorized usage and mishandling of computer system networks.

It is critical to implement network intrusion detection systems (NIDSs) in computer networks that have high traffic and high-speed connectivity. Due to the fact that software NIDSs are still unable to detect all the growing threats to high-speed environments, such as flood attacks (UDP, TCP, ICMP and HTTP) or Denial and Distributed Denial of Service Attacks (DoS/DDoS), because the main function of these kinds of attacks is simply to send more traffic in high speed to systems to stop or slow down the performance of systems.

Here we have designed a suitable real network to present experiments that use Snort NIDSs to demonstrate the weaknesses of NIDSs, such as its inability to process multiple packets at high speeds and its propensity to drop packets without analysing them. This paper outlines Snort NIDS’s failures in high-speed and heavy traffic and its propensity to drop more packets as the speed and volume of traffic increase. We ran some consecutive tests to analyse the Snort performance using the number of packets received, the number of packets analysed, the number of packets filtered and the number of packets dropped. We suggest a parallel NIDS technology to reduce dropping packets as a solution.

Keywords—network security; open source; IDS.

I. INTRODUCTION

Security is a major concern in every aspect of our daily life. New methods and equipment have been devised to ensure privacy. However, computer networks still face many threats [1]. There are usually three stages to achieving security in computer system networks: prevention, detection and correction [2]. Prevention is preferable to detection and correction, but it is impossible to prevent 100 per cent of attacks [1, 3]. Moreover, detection techniques provide more accurate results in detecting malicious attackers than correction techniques.

Despite the existence of a variety of security protection measures, an attacker often attempts to make services merely unavailable to intended legitimate users [3]. It is inadvisable to depend only on prevention techniques, especially when an attacker has successfully obtained vulnerable information from a network, but prevention can successfully and effectively restore a network before an attack is launched.

Correction techniques are adopted to protect computer systems. Along with prevention, they actively work to block intrusions, but can continue to battle a successful intrusion. Nevertheless, a number of successful attacks can be controlled using prevention techniques if an attack is detected at the interim stage of prevention systems. This is difficult, because some successful attacks can get through the prevention system [1]. It is a matter of a system being attacked, compromised, and consequently malfunctioning. Here we need an interim stage such as the detection phase, which should be positive during intrusions. Therefore, the detection method is preferred to minimize network costs and fill in the gap between correction and prevention mechanisms.

Intrusion Detection (ID) is one of the most tested and reliable technologies to monitor incoming and outgoing network traffic to identify unauthorized usage and mishandling of computer system networks [3, 4]. In addition, ID identifies the activity of malicious attackers.

Due to the fact that numerous computer systems are unable to prevent threats such as flood attacks, DoS attacks and DDoS attacks affect many systems, because the impact of such attacks is severe and irrevocable. The main function of these kinds of attacks is to send more high-speed traffic to a network address, which stops or slows down the performance of legitimate users’ computer network systems by exploiting vulnerabilities such as misconfigurations and software bugs generated from internal and external networks.

It is critical to implement ID systems (IDSs) in computer networks that have high traffic and high-speed connectivity [5]. IDSs consist of either software applications or hardware to listen for and detect malicious activities at the gateways (incoming and outgoing) of individual or network systems. Therefore, an IDS’s sniffing mechanism is effectively applied at the network gateway, which provides useful information about packets and traffic to security professionals.

Hardware-based NIDSs are effective and useful for large organizations and companies, but their high cost is an issue. However, one of the most popular NIDS software is Snort.
In this paper, we present an experiment to test Snort NIDS under high-volume and heavy traffic, demonstrating that it drops more packets as the speed and volume of traffic increase. We will also prove that the performance and capability of NIDS can be increased and the processing time of network traffic can decreased and effectively handled by an alternative technique known as parallel NIDS.

II. BACKGROUND

A. Security Products

Security products such as firewalls and antivirus programs are less efficient than IDSs and have different functionalities. IDSs analyse collected information and infer more useful results than other security products. The difference between IDSs and security products such as antivirus programs is that, while IDSs require more intelligence than security product software, they analyse gathered information and deduce useful results [1].

1) Firewall technology

Network traffic is usually filtered according to criteria such as origin, destination, protocol and service, typically through dedicated routers called firewalls [1].

The functionality of the firewall is based on filtering mechanisms specified by a set of rules, known as a policy, which can protect a system from flooding attacks [1]. The basic operation of firewalls is to filter packets passing through specific hosts or network ports, which are usually open in most computer systems [1]. It does not perform deep analysis (malicious code detection in the packet) and treats each packet as an individual entity [1].

The disadvantage of a firewall is that it cannot fully protect an internal network; it is unable to stop internal attacks [1, 6]. For example, malicious and unwanted web traffic can go through a firewall to strike and damage a protected computer system without a hitch.

2) Anti-virus technology

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 [6]. The functionality of an anti-virus program is a running process that examines executables, worms and viruses in the memory of guarded computer/network systems instead of monitoring network traffic.

Although an anti-virus program monitors the integrity of data files against illegal modifications, it is unable to block unwanted network traffic intended to damage the network [7].

3) IDS technology

Firewalls have been used for network security for a long time, but they can be easily bypassed, as a lot of techniques for deceiving firewalls have been developed [8]. IDSs are much more advanced and enhanced security tools than firewalls, because a firewall just drops packets—it cannot detect intrusion [1].

In addition, it is difficult to detect suspicious activities in the midst of high traffic and other such adverse circumstances in the network, which results in an inaccurate detection mechanism. IDSs are still unable to control all threats and malicious activities [1, 9]. To overcome such design and implementation difficulties, novel IDS outcomes have been obtained from multiple characteristics of advanced computer networks:

- Processing in real time;
- High speeds and high loads;
- Reducing difficulties for defenders; and
- Increasing difficulties for attackers.

The specialized IDS mechanism is based on how, where and what it detects, along with mandatory requirements. In particular, IDSs should be based on flexible and scalable network components to accommodate the drastic increase in today’s network environments. They should provide straightforward management and operational procedures and steps rather than complicating underlying tasks, and they should provide user-friendly ID mechanisms.

III. INTRUSION DETECTION SYSTEM

An intrusion detection system (IDS) is used to make security professionals aware of packets entering and leaving a monitored network. IDSs are often used to sniff out network packets, thereby providing a good understanding of what is really happening on the network. An IDS is based on either hardware or software, where incoming and outgoing individuals and/or network traffic have been listened to, and has the potential to detect and report any evidence of attacks [4, 10].

The typical actions of IDS software can be classified as follows:

- Monitoring entire and/or partial packets;
- Detecting suspicious activities;
- Recording required events; and
- Sending updates to the network administrator.

IDS are classified into three main types: network-based, host-based and hybrid.

A. Network-based IDSs

Network-based IDSs (NIDSs) have become a critical component of an organization’s security solution [24]. An NIDS is capable of detecting a broad range of malicious and unwanted attacks occurring in an application, network and transport layers, along with unexpected services based on multiple applications. In addition, NIDSs are able to detect and monitor network traffic and secure computer systems from network-based threats without network policy violations [11].

Disadvantaged NIDS are usually unable to execute entire network packets, which results in incomplete analyses and therefore considerable delays in high-speed and high-load environments [11].

B. Host-based IDSs

Host-based IDSs (HIDSs) are implemented to monitor suspected events happening in local host machines. HIDS are versatile due to their installation over servers,
workstations and notebooks, as compared to NIDS. In addition, HIDSs are capable of monitoring malicious networks and multiple events happening within the protected host. An HIDS is situated at the end point of a computer network that has anti-threat applications such as spyware detection, firewalls and antivirus software programs, which provide access to outside environments such as the Internet [11].

The disadvantages of an HIDS are as follows:

- It consumes computer system resources that should be allocated for services.
- It may conflict with existing security policies of firewalls and operating systems.
- It cannot easily analyse intrusion attempts on multiple computers.
- It can be very difficult to maintain in large networks with different operating systems and configurations.
- It can be disabled by attackers after the system is compromised.
- It requires many hosts to reboot after a complete installation or an update [5]. Many essential servers cannot support this operation.

C. Hybrid-based IDS

In some situations, HIDSs and NIDSs may unable to fulfil the requirements for intrusion detection because any one type of IDS has both inherent virtues and shortcomings. Therefore, a combination of an HIDS and an NIDS is known as a Hybrid IDS [5].

IV. NETWORK INTRUSION DETECTION SYSTEM METHODOLOGY

NIDS can be classified into three (3) fundamental categories:

A. Anomaly-based IDSs

Anomaly-based IDSs require a background of foundation-based information and need particular knowledge of the system being protected. Such systems have profound merit in gathering evidence in the form of statistics, data, facts, and figures, which are responsible for the formation of baselines during the learning period.

The baseline profile is the normal learned behaviour of the monitored system and is developed during the learning period, while the IDS learns the environment and develops a normal profile of the monitored system. This environment can be a network, users, a system, etc. [9]. Anomaly-based IDSs are further classified into the following anomalies:

- Protocol-based Anomaly
- Application Payload-based Anomaly

B. Signature-based IDSs

C. Protocol Anomaly detection (Models are built on TCP/IP protocols using their specifications).

Intruders usually use signatures which behave similarly to viruses used in computers. Protocol anomaly detection analyses data packets related to IPs, which contain known anomalies and single or sets of signatures. The detection system is capable of detecting suspicious activity in the logs and generates alterations based on these signatures and rules. On the other hand, anomaly-based IDSs generally depend on detecting packet anomalies available in the header parts of the protocol.

V. SIGNATURE-BASED DETECTION

A signature is generally based on an observable pattern inside the data packet. This technique helps to detect several kinds of attacks and the presence of intrusive activity, such as the presence of “scripts/iisad-min” in a packet used for web services. However, it is difficult to sort out the signatures in the headers of IP, UDP, TCP, Application Layer and Payload [12].

Signature-based IDSs are efficient at detecting predefined attacks, but they increase the size of databases because each available signature must be entered or made available in the database. Therefore, each arriving packet can be compared with the signatures available in the database, but this reduces the efficiency of the system in terms of time and throughput and increases network delays.

It is now common to test NIDSs in high-speed infrastructures with large amounts of data, but they are unable to rectify malicious activities and threats. NIDSs are effective and useful in controlling malicious activity and threats under circumstances where traffic is constantly growing [7, 13]. NIDSs are further classified into software- or hardware-based. It is also observed that software-based NIDSs still require enhancement for a network with a high volume of high-speed data, but they are useful for small networks [13, 14]. However, one of the most strong and popular open-source NIDS is Snort.

The Snort NIDS was introduced as a light-weight-based IDS [13, 14], but due to the drastic growth of technology, it has been significantly improved as well [15]. It consists of a combination of language rule driving and signatures, where protocol anomaly- and signature-based inspection methods are used [15]. In spite of the huge development over the years, Snort is still struggling to sustain its growth in the network industry and attacks. Many studies indicate that it is unable to cope with recent attacks [15, 16], because multiple threats and attacks such ICMP, HTTP, UDP (flood attacks) and DDoS attacks have adopted high traffic and speed to attack a system.

VI. SNORT OVERVIEW

Snort is accessible free of cost and is ranked among the top systems available nowadays with the best features. It is released as an open-source NIDS based on a rule-based IDS, which stores information in text files; such text files can be modified by a text editor. Rules are grouped into
categories where the rules that belong to each category are stored as information in separate files; such files are then integrated to the main configuration file, named “snort.conf”. The data is captured in terms based on described rules, which are read at the initialization of the Snort and used to construct the internal data structure [12].

A. Snort component functions

A Snort-based NIDSs consist of the following major components:

- Packet Decoder;
- Pre-processors;
- Detection Engine;
- Logging and Alerting System; and
- Output Modules.

The basic structure is represented in Figure 1. 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 organises such packets for pre-processing using a detection engine.

Pre-processor filters organise and modify the data packets before transferring them to a detection engine, such as multiple UPD and/or TCP packets and port numbers, during a short period of time.

The detection engine is time-critical and the most important part of the Snort. It utilizes different times to be processed based on the length of the packet, the specification of the system and the number of rules defined in the system. Snort can drop few packets because it runs in real time if the operation is in NIDS mode with heavy and high-volume traffic [12].

This technique employs the Snort rule to detect the intrusion action to be presented in the data packet. The Snort rule is capable of reading the chains (internal data structure), which have to be matched against all packets. If a packet does not match any rule, it will be dropped; otherwise appropriate action is taken [12].

Logging and alerts depend on the nature of what is detected inside the packets. If any suspicious activity is found inside the packet, the packet usually logs the malicious activity and/or generates an alert. Logs are usually stored in simple text-based files such as tcp-dump-style, etc. [12].

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 [12].

VII. EXPERIMENT EVALUATIONS

The main tests used for these experiments are represented in a real network in Figure 2. Under Intel core i5 and i7 and Windows 8 and 7, we ran 14 consecutive tests to analyse Snort’s performance using the number of packets received, the number of packets analysed, the number of packets filtered and the number of packets dropped.

We used the last version of Snort, “Snort_2.9.4.6 Apr, 2013” and NetScanPro tools to manage packets traffic through the network and hosts; we also used Packet Generator and WinPcap tools to send different numbers of packets at different traffic speeds through the networks and a specific IP host.

A. Experiment 1. Testing Snort in heavy traffic networks

The transmission rate of packets was kept to the same speed, 1ms, for a fair analysis between different values (packets). For this experiment we ran 3 consecutive tests; for each test, the number of the packets sent was increased. The tests started by running Snort.

The packets sent were 1024 bits long, and the time of each trip is approximate to the millisecond.

![Snort Architecture](image1.png)

![Simple network design](image2.png)
As a result, Snort analysed every single packet that reached the wire (shown in Figures 3, 4, and 5). As the number of packets increased, we observed that Snort dropped the packets; also, the Figures show that while the number of packets increased, the number of packets dropped increased as well. Our experiments show that Snort’s efficiency dropped more than 46 per cent, as the change in value occurred from 200 to 100,000 packets.

B. Experiment 2. Testing Snort under high-speed traffic
The number of packets was kept to the same value, 100,000, for a fair analysis between different speeds. Here, we ran 3 consecutive tests; for each test, the speed at which the packets were sent was increased.

As a result, Snort analysed every single packet that reached the wire (shown in Figures 6, 7, and 8). As the speed increased, we observed that Snort dropped the packets more rapidly; the Figures also show that the number of packets dropped increased as the speed increased.

Snort dropped more than 49 per cent as the change in speed changed from 4ms to 0.5ms.

C. Experiment 3. Testing Snort with large packets
For this experiment the number of the packets was kept to the same value, 5000, and the same speed (rate of transmission 2ms per packets) for fair analysis between different sizes and lengths of packets. Here, we ran 3 consecutive tests; for each test, the size of each packet sent, "Len", was increased. The test started by running Snort. We used NetScanPro tools to mange packet traffic through the network and hosts; we also used Packet Generator to send the same number of packets of different sizes through the networks at the same speed.
As shown in Figure 12, our experiment demonstrated that, as the volume and speed of traffic increased, the number of dropped packets increased drastically as well. Snort’s detection rate decreases as traffic speed increase in a computer network.

**Note:**

When we tested Snort under different Operating Systems (OSs) such as Win 8, 7, XP and Unix with the same processor, there was no significant difference in the number of packets Snort dropped, but when we tested Snort with different processors (Intel Pentium® D CPU 2.2GHz, Intel® corei5 2.27GHz and Intel® corei7 2.40GHz), the differences between them were considerable. As shown in Figure 13, Corei7 dropped fewer packets than the others.

**E. Experiment 5: Parallel Snort NIDS by configuration with vlan and Ports**

The main idea of this experiment was to confirm that parallel Snort NIDSs will reduce the number of dropped packets. We configured a switch to 4 vlans (v1, v2, v3 and v4) and ran 4 executive tests: in the first, we ran Snort in v1 and sent 8,000 packets to V1 using the Packet Generator tool. In the second test, we ran 2 Snorts in v1 and v2 and sent the same packets to V1 and V2 while configuring vlan bandwidth and priority to control the number of packets going through each vlan. Each time we
ran more Snorts with more vlan. The results are shown in Figure 14.

TABLE I.

<table>
<thead>
<tr>
<th>Snort</th>
<th>Number of packets sent</th>
<th>number of packets received</th>
<th>number of packets analysed</th>
<th>number of packets dropped</th>
<th>% Packets analysed</th>
<th>% Packets dropped</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Snort /V1</td>
<td>80000</td>
<td>8045</td>
<td>7056</td>
<td>989</td>
<td>78.707%</td>
<td>10.948%</td>
</tr>
<tr>
<td>2 Snort /V1 and V2</td>
<td>8000</td>
<td>8070</td>
<td>7562</td>
<td>490</td>
<td>93.893%</td>
<td>5.734%</td>
</tr>
<tr>
<td>3 Snort /V1, V2, and V3</td>
<td>8000</td>
<td>8076</td>
<td>8001</td>
<td>75</td>
<td>99.071%</td>
<td>0.920%</td>
</tr>
<tr>
<td>4 Snort /V1, V2, V3, and V4</td>
<td>8000</td>
<td>8108</td>
<td>8108</td>
<td>0</td>
<td>100.00%</td>
<td>0%</td>
</tr>
</tbody>
</table>

As shown in Figure 14, our experiment shows that when we sent a number of packets to one Snort some packets were dropped, and when we ran more Snorts in a multi-processor environment with the same number of packets, the number of dropped packets was reduced. Our experiment therefore demonstrated that when we used parallel Snorts, fewer packets were lost.

VIII. RELATED WORK

One of the most important weaknesses of NIDSs is that when processing the entire increase in traffic, it is crucial that more efficient approaches are developed until IDSs can process more and more traffic in less time. To find a solution to this problem, several nodes can be used to process network traffic concurrently and in parallel. Wheeler and Fulp [17] proposed a framework that is complementary to NIDS. Their research illustrates that three levels of parallelism can be used:

- the node level (the node plays a vital role in the running of several identical systems in a parallel fashion: rules are taken from their original groups and extended to all other running nodes);
- the component level; and
- the sub-component level.

In Snort, rules are placed into rule groups based on their source and destination: for example, rules associated with web traffic are usually placed in port number 80 [17]. It has been explained that Snort organises the rules into different groups, and each individual group is commonly recognised by its file name. This can be best described by the “pop-rules”, which are all the rules that are relevant to the Post Office Protocol (“POP”). Packet duplicators are used to duplicate incoming packets that run across all the nodes at the same time in such a way that identical rules are sent to different nodes. One can therefore assume that one packet may pass through the same inspection many times. The main flaw of this method is repetition, so that when a packet is sent to the node, the node is able to check whether its rules are related to that of the packet. Wheeler and Fulp [17] do not cover this issue in their research.

There are many discrepancies associated with the node level:

- if all communications are considered to be stateless, the node level is able to work. However, this seems to be unrealistic with today’s attack, as demonstrated by [18; 19];
- endless repetitions have been observed in it;
- no co-relation has been depicted in the packets that have been sent in multiple frames;
- a flaw in this method further reveals that it does not take into account the fragmentation issue, which is the latest primary technique used by abusers who overflow systems. The latest IDSs are unable to handle the process of fragmentation because of the sheer speed of the attacks [20].

Wheeler and Fulp [17] demonstrate that at the component level, some peculiar functions, including e-fragmentation, can be parallelised. However, it has not been properly clarified how this will take place in this system, although the risk may be exacerbated by the formation of a bottleneck at this level. This can even be increased if the top-level categorisation is not done in order to isolate the fragmented packets from whole and complete packets.

Vasiliadis, Polychronakis and Ioannidis [21] proposed a new model for a multi-parallel IDS architecture (MIDEA) for high-performance processing and stateful analysis of network traffic. They proposed a system for parallel network traffic processing and analysis at three levels: multi-queue NICs, multiple CPUs, and multiple GPUs. They took advantage of the parallelism offered by modern network interface cards, multiple CPUs, and multiple GPUs with parallel traffic using a multi-queue to improve scalability and running time. They showed that processing speeds can reach up to 5.2Gbit/S 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. Our approach differs from theirs in that we have shown how parallelism at a higher level of granularity, which is simpler to implement, can also make impressive improvements in terms of the number of lost packages.

IX. CONCLUSION AND RECOMMENDATIONS

A. Conclusion

Attacks can be categorized into multiple stages, although these are difficult to analyse and detect. Additionally, multi-attacks are difficult to detect in high-speed traffic, and are even more difficult to detect with the shift toward parallel IDS. Increasing the performance and capability of the NIDS and decreasing the processing time of the network traffic can be effectively handled by an alternative technique known as the parallel technique. In general, to develop a security system you have to incorporate four main functions: scanning, analysing, detecting and correlating. In this work, we focus on the scanning and analysing weaknesses of NIDS and suggest a parallel NIDS as a solution to improve its analysis performance.

B. Recommendation

Multi-core technology is one solution for high-speed NIDSs [22, 23]. Multi-core processors provide enhancement with high capabilities and secure networks from attacks, but they increase the complexity of the security system [23]. 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, we can look into some of the potential technological advancements in NIDSs that can be employed for beneficial purposes and objectives. Also, as hackers take advantage of multi-core processors to attack systems, we will focus on how multi-core environments benefit IDSs in high-speed networks.

REFERENCES


