PEOPLE'S DEMOCRATIC AND REPUBLIC OF ALGERIA

Ministry of high education and scientific research

University 20 aout 1955 Skikda

Faculty of sciences

Department of computer science

Option : Network and Distributed Systems

Study and Deployment of an Intrusion Detection System

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master in Computer Science

Presented by :

Bellachia Mohamed Nadir

Supervised by :

Touil Ghassen

Année Universitaire : 2021 / 2022

Dedication

In the name of Allah, Most Gracious, Most Merciful

I dedicate this work to:

My parents for their love, encouragement, endless sacrifices, and for being special parents, and making me a special person.

My brothers and sisters.

My friends.

All my teachers and all my family.

Acknowledgements

First and foremost, I would like to thank my supervisor Touil Ghassen for his guidance and valuable comments. I extend my gratitude to the teachers who helped me in this study and the members of examiners for having accepted to read and examine my dissertation. Last but not least, great thanks go to all who helped me with encouragement and support which have never ceased all along the preparation of my work.

Abstract

Internet has caused a great technological revolution in terms of the exchange of information, knowledge and science, which led to the inevitability of its use in personal and professional life, and the fate of people became hostage of this network. This necessity has led some to exploit it illegally by espionage, extortion, sabotage, data theft ... which has led to the emergence of protection methods such as Anti-virus systems, firewalls, security technologies, password, and encryption systems ... and other means of protection.

All the security systems had loopholes and tightening loopholes that exploited by hackers. This prompted researchers to invent intrusion detection systems, which corrected many of the shortcomings of previous solutions. It should be noted that there are several types of intrusion detection systems, some of which are based on the scenario approach like IDS Snort, and some of them are based on the behavioral approach of users and applications.

In our project, we installed & configured the Snort intrusion detection system, which is considered to be one of the leading systems in the intrusion detection field, and we installed the graphical display management software Splunk for easy reading and analysis of the alarms and logs that Snort generates as well as pulledpork a tool used for automatic management of Snort rules.

With the advancements in the attack techniques and viciousness, the question we ask ourselves is whether Snort is still viable as an Intrusion detection system or is it an obsolete tool that can no longer handle the latest attack developments.

Table of contents

List of figures:

List of Tables

General introduction

The Internet has become the most primordial medium and one of the main sources of information and data exchange in the world today. The Internet can be considered as one of the important tools in many sectors, educational commercial social military scientific, health....

It attracts more and more Internet users by the many advantages and the diversity of services made available. They can thus benefit from fast communication with minimum cost, share processing and storage resources of and storage resources (Cloud Computing), facilitate commercial and financial exchanges (e-Commerce, e-Banking) and, more generally, share and access to information. Indeed, the Internet must be a secure and reliable environment.

Computer security is one of the main concerns of companies and individuals today. The increase in the number of Internet users makes the Internet full with people of good and bad intentions. They can exploit the vulnerabilities of networks and information systems to try to access, view, modify, and use sensitive information in order to consult, modify or destroy it and thus the dysfunctioning of the system. These threats and attacks are becoming a real problem for companies and organizations. Our increasing dependence on computer systems in our daily lives inevitably raises the issue of securing these systems and secure the information circulating.

Various means have been invented to prohibit any attacks such as authentication servers, antivirus, firewalls, proxies, etc., these means are not sufficient to block all types of attacks that violate confidentiality, integrity or availability. To address this problem, a new security tool called Intrusion Detection Systems (IDS), introduced by James Anderson in 1980, has been developed.

CHAPTER 01:

Introduction to Cybersecurity

1. Introduction

In the beginning, the Internet connected American government and military institutions, which did not pose any security problems, since such a network can be considered private, according to the current terminology. Virus, attack, spam, intrusion ... etc., were not concepts to be considered in the computer field. Gradually, the network opened to everyone, and thus became both vector and target of attacks. Thus, for several years, a multitude of computer attacks have taken place. In this chapter we present the security aspects related to computer networks.

2. Cybersecurity

Although the targets for cyberattacks may vary widely, they are primarily focused on money, intellectual property and, of course, sabotage. Cybersecurity is a collection of defensive technologies (hardware/software), processes and practices designed to protect networks, computers, programs and information from attack, damage or unauthorized access in order to secure systems that are connected to the Internet. By definition, Cybersecurity protects against threats using defensive measures, including information assurance, computer systems, and applications hardening, malware protection, access control, information infrastructure protection, and network security.

3. Network Security [1]

Network security is a continuous process built around a security policy. This process ranges in complexity from configuring routers to not accept unauthorized addresses or services to installing firewalls, IDSs, centralized authentication servers, and encrypted virtual private networks (VPNs).

Network security is a continuing process following these steps:

- **Secure**: these methods are used to keep a network safe:
  - Authentication
  - Encryption
  - Firewalls
  - Vulnerability patching

- **Monitor**: it is important to monitor the state of security preparation in order to ensure that a network remains secure. Using security monitoring solutions, organizations can obtain unprecedented visibility into both the network data stream and the security posture of the network.
- **Test**: Testing security is an important as monitoring. Without testing the security solutions in place, it is impossible to know about existing or new attacks.
- **Improve**: Monitoring and testing provides the data necessary to improve network security. Administrators and engineers should use the information from the monitor and test phases to make improvements to the security implementation as well as to adjust the security policy as vulnerabilities and risks are identified.

3.1. Network Security Threats [2]

The following are the main type of threats to network security:

3.1.1. Unstructured threats

These threats consist of random hackers using various common tools, such as malicious shell scripts, password crackers, credit card number generators. hackers in this category are more interested in the intellectual challenge rather than creating havoc.

3.1.2. Structured threats

These threats are created by hackers who are more highly motivated, organized and technically competent. They aim to understand, develop, and use sophisticated hacking techniques to penetrate unsuspecting small or big businesses.

3.1.3. External threats

These threats consist of structured and unstructured threats originating from an external source. These threats may have malicious and destructive intent, or they may simply be errors that generate a threat.

3.1.4. Internal threats

These threats typically involve disgruntled former or current employees. Although internal threats may seem more ominous than threats from external sources, security measures are available for reducing vulnerabilities to internal threats and responding when attacks occur.

4. Attacks and Mitigations

4.1. Cyber Attack Lifecycle [3]

The cyber-attack lifecycle also known as the kill chain depicts the stages of cyber-attack:

- **Recon**: involves observation, research, and planning of and into a target that fits the needs of the mission of the attacker
- **Weaponize and delivery:** the attackers breach the network and install malicious software
- **Exploit**: the initial attack on target is executed using an exploit kit.
- **Control**: after ensuring a continued control over the network the attacker installs the appropriate set of malwares according to his plan.
- **Execute**: leveraging several techniques the attacker executes his plan for the attack for example data exfiltration, destruction of critical infrastructure, or creating any means of extortion.
- **Maintain**: long term access is achieved in case if a new mission is presented

4.2. Types of Attacks

There are four major types of attacks

- **Reconnaissance attacks**: attacks with the goal of learning information about a target network by using readily available information and applications.
- Access attacks: attacks that aim to retrieve data, gain access, or elevate access privileges
- **Denial of service attacks:** a form of attack that damages and corrupts a computer system or denies the users and administrators access to the network, systems and services
- **Malwares:** The primary vulnerabilities for end-user workstations, a malicious software is inserted onto a host in order to damage a system, corrupt a system, replicate itself, or deny services or access to networks, systems, or services.

4.2.1. Reconnaissance attacks and mitigations

Recon is an important stage of cyberattack life cycle, where attackers search for vulnerabilities that they can use to attack targets [4]. There are three types of reconnaissance attacks:

4.2.1.1. Packet Sniffer

Packet sniffer is a piece of software application that uses a network adapter card in promiscuous mode. The functioning principle of Sniffers is to examine the stream of data packets that flow between computers on a network or between networked computers or the larger internet. It is used to exfiltrate sensitive information from unencrypted messages.

4.2.1.2. Port Scans and Ping Sweeps

Port scans and ping sweeps are testing applications designed to identify vulnerable services on a targeted host or a device. These tests can identify all services, hosts and devices on the network it can also identify the operating systems as well as the vulnerabilities on that same network.

4.2.1.3. Internet Information Queries

There are several queries on the internet with various tools that can be used to retrieve the information on those queries. For example, IP address queries that can reveal the ownership of certain IP addresses it's range and the domains associated with them.

4.2.2. Access Attacks and Mitigations

Access attacks use known vulnerabilities in authentication, ftp and web services to gain access to confidential and sensitive information and databases. These attacks consist of:

4.2.2.1. Password attacks

Password attacks usually refer to repeated attempts to identify a user account a password or even both, these repeated attempts are called the brute force method. Other methods implemented by hackers to initiate a password attack are trojan horse programs, IP spoofing and packet sniffers. Several techniques have been implemented to mitigate such attacks:

- Use strong passwords (a password that is at least eight characters long and contain uppercase letters, lower case letters, numbers, and special characters).
- Disable account after a certain number of failed authentication attempts.
- Forbid the use of a same password on multiple systems.

4.2.2.2. Man in the Middle attacks

A man in the middle attack requires access to network packets that come across the network. An example of an attacker could be an employee working for your ISP who has access to all network packets. This type of attacks is used for various reasons such as information theft, traffic analysis, corrupting transmitted data, hijacking of an ongoing session. Man in the middle mitigation is achieved by encrypting traffic in an IPsec tunnel, which will show hackers a cipher text instead of clear text.

4.2.3. Denial of Service Attacks and Mitigations

The most publicized form of attack, Dos attacks require little to no effort to execute and still cause significant damage. Dos attacks consist of the following:

4.2.3.1. IP Spoofing

IP spoofing occurs when a hacker impersonates the conversation of a trusted computer using a within-range IP address or a trusted authorized external IP address. IP spoofing is usually used for the injection of malicious data or commands into an existing stream of data This use is not as common however if an attacker was able to modify the routing table to point towards the spoofed IP address, he can receive all the network packets addressed to the spoofed address and be able to reply just as any trusted user. The effectiveness of IP spoofing can be reduced but not completely eliminated following these measures:

- Access Control: deny any traffic from the external network that has a source address that should reside on the internal network. (The internal addresses are the only trusted addresses. If some external addresses are trusted, this method is not effective).
- RFC 2827 filtering: prevents users on the network from spoofing other networks.
- Require additional authentication that does not use IP-based authentication.

4.2.3.2. Dos and Distributed DOS

Dos attacks do not aim at gaining access or collecting information instead these attacks focus a service unavailable for normal use.[4] these attacks are carried by taking advantage of protocol weaknesses or using native traffic (normal allowed network traffic). The danger of DOS attacks can be minimized using several methods:

- Proper configuration of Antispoof features

- Proper configuration of AntiDos features
- Traffic rate limiting

4.2.4. Malware

The malware is a computer software program developed to install on the computers without any consent of the users. It Is mainly used to either establish an access to the targeted computers without any permission or to create annoyance for the users [5]. There are many types of malwares:

4.2.4.1. Worm attacks

The main feature of a computer worms and viruses is that they replicate themselves in order to spread to other computers on the network or through other data transmission media. Worms are programs that attack and try to exploit a vulnerability in the targeted system. It goes through multiple stages:

A worm installs itself using an exploit vector on a vulnerable system. After gaining access to the targeted device, it replicates and selects a new target. Once the device is infected with a worm, the attacker uses local exploits to escalate their privileges level from a super user to administrator.

Worm attacks mitigation requires coordination between system admins, network engineering and security operations in following these steps for an effective incident response:

- Containment: contain the spread of the worm inside and within the network.
- Inoculation: scanning for vulnerabilities and patching the systems.
- Quarantine: disconnect, remove, or block any infected part of the machine from the network.
- Treatment: Clean and patch each infected system.

4.2.4.2. Virus and trojan horse attack

Viruses are malicious software that are attached to another program. A virus provides illegal access to a host's resources and infects it for example through an email attachment, and may contain spyware, trojans or worms. It is also capable of spreading itself to other hosts.

Trojans are software applications that are made to look like anything but an attack tool. It may run a simple game on the user's workstation while it mails a copy of itself to every other user in the user's address book, thus spreading the trojan horse. Trojans serve as a backdoor for illegal access to a host. These kinds of applications can be contained by using an up-to-date antivirus software and intrusion protection.

4.2.4.3. Spyware, rootkits, and botnets

- Spyware records keystrokes and other crucial activities and uploads the information to a collection site.
- Rootkits is a type of malware that gets the administrator-level privileges on the OS of the computer without showing its presence on the computer.
- Botnet is short for robotic net; it is a connected set of programs designed to operate together over a network to achieve a certain goal. Some programs are used to help and support internet connection others are used to take control of computers functions to support large scale service attacks (DOS).

An anti-malware is a useful computer program that is designed to detect, isolate, and remove any malware attack it detects.

5. Methods of Technical Defence

Attacks can happen anywhere on the attack surface (data, network, devices, applications, systems), so we should consider the primary methods of defence and where they can be deployed. The primary technical methods of defence can be considered to be:

5.1. Firewalls

5.1.1. Network firewall

Network firewalls are often placed at the front line of defence, filtering of traffic and dividing the network into multiple password-protected segments. A firewall can monitor the inbound and outbound traffic of an enterprise network or the traffic between sub-networks of the enterprise network. A firewall has multiple physical interfaces, each corresponding to an internal sub-network or the external network [6]. If needed, multiple interfaces can be grouped into a security zone such that the traffic within a security zone is not monitored but the inbound and outbound traffic of a security zone is monitored otherwise, each interface corresponds to a

security zone. The monitored traffic will be examined according to some security policies or rules, which specify what kinds of traffic are authorized [7].

5.1.2. Host-Based Firewalls and Personal Firewalls [8]

Host-based firewalls for servers and personal firewalls for desktop and personal computers (PC) provide an additional layer of security against network-based attacks. These firewalls are software that reside on the host that they are protecting, each monitor and control the inbound and outbound network traffic for a specific host.

Host-based firewalls are considered as part of server operating systems, and they can also be installed as third-party add-ons. They protect against malicious activity from all hosts (including those on the same subnet) by configuring them to only allow the necessary traffic to the server. Limiting outgoing traffic from a server may prevent certain malware from spreading to other hosts. Host-based firewalls are able to perform logging, as well as address-based and application-based access controls if configured to.

A personal firewall is software-based application that runs on a desktop or laptop PC with a user-focused operating system such as Microsoft Windows Vista or Macintosh OS X. Because the computer being protected is meant for regular users, a personal firewall is slightly different than a host-based firewall, the provided interface is usually less complicated and easier for end users to understand. A personal firewall can restrict inbound communications and can often limit outbound communications as well. This allows personal firewalls to protect PCs from incoming attacks as well as limiting the spread of malware from infected PCs and the use of unauthorized software such as peer-to-peer file sharing utilities (such as BitTorrent).

5.2. DMZ (Demilitarized Zone):

DMZ is a security zone that typically hosts Internet-facing servers. DMZ can isolate a designated area in a network from the outside network by making external computers have no valid reasons to directly communicate with the computers in the designated area of the network. DMZs can be built to enhance security, used for testing or lab environments or for guest networks [9]. A DMZ can house various different resources that both public users and trusted network user will have access to.

A DMZ represents a barrier for your important resources and for the rest of the network. However, it comes at the cost of performance where traffic now has to go through multiple security measures. The level of degradation depends on the design of the DMZ.

A DMZ is comprised of either a separate IP address segment or using a segment from the current network using VLAN. The zone is separated from the internet and the trusted network utilizing firewalls, routers and/or switches. The IP address scheme can either be public or private addressing. Network Address Translation or NAT is also used to mask the DMZ addresses (it is used to translate private IP to public as traffic moves in and out of the zone).

Considering the design of the DMZ, there are four levels of basic DMZ designs to build from. With each level of design, comes an increase in security but also comes at a cost of complexity. In the Level 1 Design, the DMZ is a separate part of the network that is accessible off a port from the boundary firewall. This creates a single point of both protection and filtering. With a Level 2 Design, there are multiple DMZs utilizing multiple ports off the firewall. We move to the Level 3 DMZ Design. With this design, we are creating an external and an internal boundary within each DMZ by using multiple firewalls. This design gains an increase in security as firewall rules or access from the internet can be granted through the external firewall but blocked at the internal firewall. The last design is the Level 4 DMZ Design. Similar to level 3 it utilizes a double boundary design. However, level 4 design utilizes a multiple firewall pairing to create the boundaries between DMZs. This allows the spreading of resources between each boundary firewall pairs, and also allows the separation of DMZ areas into functional or business areas which can be helpful for business organizations [10].

The security provided by the DMZ is based on the complexity of its design. all designs utilize a firewall scheme and require the same basic filters and access controls. Isolating the DMZ resources also assists in security. If one DMZ resource is attacked, the other DMZs can be assumed safe.

5.3. Proxy servers

A proxy server is a server that sits between a client application, such as a Web browser, and a real server. It intercepts all requests to the real server to see if it can fulfil the requests itself. If not, it forwards the request to the real server [11].

The basic principle of operation of the proxy server is to receive client requests, these requests are analysed and then send to the target servers (to whom they are acting as a client), and then the answers are passed to the original client (in original or modified form). The proxy server operates on the 7th layer of the OSI model (application layer) to analyse incoming requests [12].

The proxy server is used as a security and privacy measurement because it destroys information about the user's computer in the request's header. So, the user can safely surf the internet and his data will never be used by hackers and spammers. Sometimes we encounter some issues while accessing to web server (for example, web-chat). We are mistaken while working with some data and/or the server administrator restricted the access from our IP. So, we can use the anonymous proxy and try to access again.

A proxy server can be placed in the user's local computer or at various points between the user and the destination servers or the Internet. With the fast development of network, the network issues such as viruses, attacks, hacks are increasing day by day. So, network monitor and analysis are becoming more and more necessary nowadays.

5.4. Antivirus

Antivirus is a computer protection software that is designed to evaluate data such as web pages, files, software and applications to help find and delete any malicious software or piece of code. Most antiviruses provide real-time protection, which can protect your devices from incoming threats, and scan your entire computer regularly.

The basic principle of operation of the antivirus is to check the computer programs and files against a database of known types of malwares. It will also scan for the possibility of a new or unknown type of malware, since the attacks are constantly developing.

Typically, most programs will use three types of detection: specific detection, which identifies known malware; generic detection, which looks for known parts or types of malware or patterns that are related by a common codebase; and heuristic detection, which scans for unknown viruses by identifying known suspicious file structures. When the program finds a file that contains a virus, it will usually quarantine it and/or mark it for deletion, making it inaccessible and removing the risk to your device.

5.5. Encryption/Cryptography

Cryptography is a strategy for framing and transmitting information in a specific manner so that those for whom it is intended can read and process it. The term is commonly associated with scrambling cleartext message into ciphertext in a procedure known as encryption, then back once more, and this procedure is called decoding. There are three types of cryptographic plans used to achieve these objectives: mystery key (or symmetric), open key (or hilter kilter), and hash works [13].

The key can be numeric or alpha numeric manuscript or can also be a unique figure. The clear text is the message that the first individual wishes to speak to the other. The cipher Text is the message that can't be comprehended by anyone, it is also known as an aimless message.

Encryption is a procedure of changing over plain content into figure content. This procedure requires two things an encryption calculation and a key. Calculation implies the system that has been utilized as a part of encryption. Encryption of information happens at the sender side. Where, Decryption is a turn-around procedure of encryption. In this procedure Cipher content is changed over into Plain content. Decoding process requires two things an unscrambling calculation and a key. Calculation implies the method that has been utilized as a part of Decryption. By and large the both calculations are same.

5.6. Intrusion Detection Systems

Intrusion detection system is a computer program that monitors and inspects electronic communications for the purpose of detecting and alerting the user of any malicious or unwanted stream of information. Based on the generated alerts, a security operator or incident responder **can investigate the issue and take the appropriate actions to remediate the threat.**

This defence method will be discussed in detail in the next chapter.

6. Conclusion

It is difficult to overstate the importance of computer networks and cybersecurity in today's world. They have become such an integral part of our existence that only a moment's reflection is required to delineate the many ways in which they impact essentially every aspect of our lives. Thus, several defence mechanisms were deployed in order to protect those aspects of our

lives. In our next chapter we'll learn about one of the main technical methods of defence the IDS, it's primary functions and its goals.

CHAPTER 02:

Intrusion Detection Systems

1. Introduction

Security of a network is an important issue. With the continuously growing network, the basic security mechanisms like firewalls, virus scanners are easily bypassed by attackers who are very resourceful in utilizing and manipulating software vulnerabilities to attain their goals. In order to prevent such attacks, we have to use a more evolved security mechanism that can act more proactively and intelligently. an intrusion Detection System is that solution for such requirement.

2. Definitions

2.1. Intrusion

Intrusions are attempts to compromise or to bypass the security mechanisms of a computer or network, perpetrated by unauthorized users or users who attempt to gain additional privileges for which they are not authorized for.

2.2. Intrusion Detection

Intrusion detection is the ability to detect attacks against networks (including network devices and hosts), whether by matching signatures to previously known attacks or by detecting abnormal behaviour.

2.3. Intrusion Detection Systems

Intrusion Detection Systems are very important software or hardware security tools to remove threats that would otherwise occur when carrying information, to prevent unauthorized access or abuse, and to report attacks to those responsible for security [14].

3. Why should I use Intrusion Detection Systems? [15]

Intrusion detection allows to protect the systems from the threats that include increasing network connectivity and reliance on information systems. With current dangerous level of network security threats, the question for security professionals is not whether to use intrusion detection, but which intrusion detection features and capabilities to use. IDSs are used as a necessary component in every security system infrastructure. There are several advantages to using an IDS:

- **increase the perceived risk of discovery and punishment of attackers**

An important goal of computer security management is to prevent security issues in the information system by limiting the behaviour of the individual user. Intrusion detection systems helps accomplish this goal by increasing the perceived risk of discovery and punishment of attackers. This makes it very difficult to attempt a violation on the security policy.

- **Detecting problems**

An IDS can detect when an attacker has infiltrated a system by exploiting an uncorrected flaw. It serves an important role in protection, by alerting the administrators who can contain and recover any damage that results. This is much preferable than ignoring network security threats and allowing the attackers continued access to systems and also the information on them.

- **Detecting the preambles to attacks**

Before attempting an attack, intruders usually probe and examine a system or a network searching for weak points and optimal points of entry. IDS can observe and identify these suspicious probes and may actively block the attacker's access to the target system after alerting the security personnel of this intrusion attempt.

- **Documenting the existing threat**

Understanding the frequency and characteristics of attacks allows you to understand what security measures are appropriate to protect the network against those attacks. This documenting feature works as a feedback system assisting you in making proper decisions regarding your allocation of computer security resources. The information that IDSs give you regarding the source and nature of attacks allows you to make decisions regarding security strategy driven by demonstrated need.

- **Quality control for security design and administration**

Overtime the IDS can be used as a feedback system for the quality of the security design and administration which will help in its improvement and avoid future issues

- **Providing useful information about actual intrusions**

IDSs still collects a detailed amount information about the attack that helps incident handling and recovery efforts after a failed attempt to block it. This information can enable and support

criminal or civil legal approaches to the incident. Ultimately, such information can identify problem areas in the security configuration or policy.

4. Indicators of compromise [15]

When talking about detection, it's important to speak about Indicators of Compromise (IoC). When new threats are found within the wild, they sometimes have a pattern of behavior and that they leave their footprint within the target system. as an example, Petya ransomware ran the subsequent commands within the target system to reschedule a restart:

```
    schtasks /Create /SC once /TN "" /TR "<system folder>shutdown.exe /r
/f" /ST <time>
    cmd.exe /c schtasks /RU "SYSTEM" /Create /SC once /TN "" /TR
"C:Windowssystem32shutdown.exe /r /f" /ST <time>
```

Figure 2.1: petya ransomeware execution

Another Petya IoC is the local network scan on ports TCP 135 and TCP 445. These are important indications that there is an attack taking place on the target system and, based on this footprint, Petya is the one to blame. Detection systems will be able to gather these indicators of compromise and raise alerts when an attack happens. Using Azure Security Centre as an example, some hours after the Petya outbreak, Security Centre automatically updates its detection engine and was able to warn users that their machine was compromised, as shown in the following screenshot:

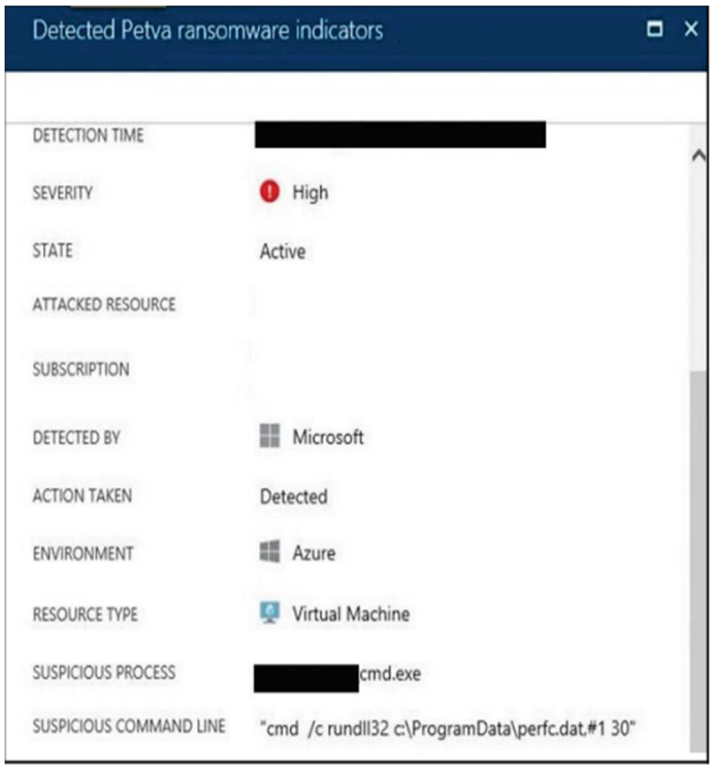


Figure 2.2: Dtetcted petya ransomeware indicators

5. Major types of IDSs

IDS are classified based on various categories:

5.1. Functional components of intrusion detection [15]

5.1.1. Information Sources

This component is responsible for collecting and providing data for the next component to be analysed. The data can be audit trails, system logs or network packets. The information can be drawn from different levels of the system, network, host, or application.

5.1.2. Analysis

Analysis engine organizes and makes sense of the data derived from the information sources, deciding whether the entered data is an attack or a normal activity. The analysis approaches that are commonly used are misuse detection and anomaly detection.

5.1.3. Response

This component controls the reaction mechanism of the IDS and decides on the appropriate measures to respond when the analysis engine detects an attack. There are two types of responses, active measures involve an automated intervention by the system, and passive measures involve reporting analysis engine's findings to the user, who then takes action based on those reports.

5.2. IDS based on Information sources

Some IDSs analyse network packets, captured from network backbones or LAN segments, to find attackers. Other IDSs analyse information sources generated by the operating system or application software for indications of intrusion.

5.2.1. Network IDS or NIDS

Network based intrusion detection systems analyse network packets. They can detect malicious packets designed to be overlooked by simple filtering rules. If a packet is matched with an intruder signature on its database, an alert is generated or the packet is logged to a file or database.

5.2.2. Host IDS or HIDS

Host-based intrusion detection systems or HIDS are installed as agents on a host. These intrusion detection systems can look into system and application log files to detect any intruder activity. Some of these systems are reactive, meaning that they inform you only when something has happened. Some HIDS are proactive; they can sniff the network traffic coming to a particular host on which the HIDS is installed and alert you in real time.

5.2.3. Application IDS or AIDS

Application-based IDSs are a special subset of host-based IDSs that analyse the events transpiring within a software application. The most common information sources used by application-based IDSs are the application's transaction log files. The ability to interface with the application directly, with significant domain or application-specific knowledge included in the analysis engine, allows application-based IDSs to detect suspicious behavior due to authorized users exceeding their authorization. This is because such problems are more likely to appear in the interaction between the user, the data, and the application.

5.2.4. Hybrid IDS

The Hybrid intrusion detection systems Combine information from a number of sensors. Often, both Host and Network-based intrusion detection systems in a central analyser that is better able to identify and respond to intrusion activity.[16]

5.3. IDS based on Analysis

Based on the detection method, IDSs can be divided into three different types: misuse, anomaly, and specification based IDSs. [17]

5.3.1. Misuse detection

A misuse-based IDS also known as signature-based IDS, looks for any malicious activities by matching the known signatures or patterns of attacks with the monitored traffics. This IDS suits for known attack detection; however, new or unknown attacks also called as a zeroday exploit are difficult to be detected. misuse-based IDS has certain advantages as well as disadvantages to its method:

- **Advantages of misuse-based IDS:**
  - Misuse-based IDS can help users and administrators to know and monitor their systems. Even if they are not security experts because each signature is related to an attack.
  - The Misuse detection can be deployed without knowing the network, and it doesn't take time to learn about users and behaviours and data. The rules are defined from the beginning and updated each time they are available.
- **Disadvantages of misuse-based IDS:**
  - Misuse-based IDS cannot detect new attacks. For this reason, the system needs to be constantly updated with the newly discovered signatures.
  - When a well-known attack is changed and a variant of this attack is created, the IDS will not be able to detect this attack.

5.3.2. Anomaly detection

Anomaly-based IDS detects an attack by profiling normal behavior and then triggers an alarm if there is any deviation from it. The strength of this IDS is its capability for unknown attack detection. Misuse-based IDS usually achieves higher detection performance for known attacks than anomaly-based IDS.

- **Advantages of misuse-based IDS:**
  - IDSs adapts this method of detection to be more accurate by the time.
  - No need to update the signature database to detect new threats.
  - Very little maintenance once the system is installed it continues to learn about network activity and continues to build its profiles.
- **disadvantages of anomaly-based IDS:**
  - Anomaly-based IDS generally flags many false alarms, because deviating from normal behaviour does not always mean that an attack is occurring.
  - The anomaly-based approach requires a large set of training data that consist of system event logs in order to construct normal behaviour profile.
  - Requires a lot of time to be ready to be used with the real data to protect the user's system.

5.3.3. Specification detection

A specification-based IDS manually defines a set of rules and constraints to express the normal operations. Any deviation from the rules and the constraints during the execution is flagged as malicious.

5.3.4. Comparison between analysis techniques

The following table gives a comparison between IDS types based on their detection method, and it present each IDS with its strong points as well as their shortcomings.

	Misuse-based	Anomaly-based	Specification-based
Method	Identify known attack patterns	Identify unusual activity patterns	Identify violation of predetermined rules
Detection Rate	High	Low	High
False alarm rate	Low	High	Low
Unknown attack detection	Incapable	Capable	Incapable
Drawback	Updating signatures is burdensome	Computing any machine learning is heavy	Relying on expert knowledge during defining rules is undesirable

Table 2.1: comparison between IDS types

5.4. IDS based on Response

5.4.1. Active response

The IDS can take countermeasures against experiencing an attack by terminating the malicious process and the session of the user, and by blocking the IP source of the attacker machine and disconnect it from the network. Active IDS responses are automated actions taken when certain types of intrusions are detected.

5.4.1.1. Collect additional information

this active response involves increasing the level of sensitivity of information sources. The additional information collected can help resolve the detection of the attack by assisting the diagnostic process of the system.

5.4.1.2. Change the Environment

Another active response is to halt an attack in progress and then block subsequent access by the attacker. IDSs are unable to block a specific person's access so this method instead blocks IP addresses of the source of the attacks.

5.4.1.3. Strike back

An approach involves attacking or actively trying to gain information about the attacker. This method is to be approached with caution because of its legal ambiguity and because of the high risks it imposes on innocent internet sites and users in case the attackers use false network addresses.

5.4.2. Passive response

Passive IDS responses provide information to system users, relying on humans to take subsequent action based on that information.

5.4.2.1. Alarms and notifications

Alarms and notifications are generated by IDSs to inform users when attacks are detected. The most common form of alarm is an onscreen alert or popup window. This is displayed on the IDS console or on other systems as specified by the user during the configuration phase of the IDS.

5.4.2.2. SNMP Traps and Plug-ins

IDS use SNMP traps and messages to post alarms and alerts to central network management consoles.

This method provides several merits:

- The entire network infrastructure will adapt and respond to a detected attack.
- Switch the processing load associated with an active response to a system other than the targeted one.
- Gain the ability to use common communications channels.

5.5. Other criteria of classification

5.5.1. Architecture

The architecture of an IDS refers to how the functional components of the IDS are arranged with respect to each other. The primary architectural components are the Host, the system and the Target.

5.5.1.1. Host Target Co location

Most IDSs ran on the systems they protected. This was due to the fact that most systems were mainframe systems, and the cost of computers made a separate IDS system a costly extravagance. This presented a problem from a security point of view, as any attacker that successfully attacked the target system could simply disable the IDS as part of the attack.

5.5.1.2. Host Target Separation

To improve the security of IDSs and hide its existence from the attackers, most IDS architects moved towards running the IDS control and analysis systems on separate systems separating the IDS host and target system.

5.5.2. Goals

There are many goals associated with security mechanisms in general, however there are two main goals usually stated for IDSs.

5.5.2.1. Accountability

Accountability is the ability to hold the parties responsible for the attack accountable for their actions which requires the ability to link the attack to them. Accountability is difficult in networks that use TCP/IP protocols or in any system that employs weak identification and authentication mechanism where it is easy for attackers to forge their identity.

5.5.2.2. Response

Response is the capability to recognize a given activity or event as an attack and then taking action to block or otherwise affect its ultimate goal. the requirements of detection are quite different for response than for accountability.

5.5.3. Timing

Timing refers to the elapsed time between the events that are monitored and the analysis of those events.

5.5.3.1. Batch Mode

the flow of information from monitors to analysis engines is based on “store and forward” rather than continuous. This is a timing scheme used by host based IDSs that rely on audit trail which were stored in files. Interval based IDSs are do not perform active responses.

5.5.3.2. Real Time

In real time IDSs, the information flow from monitoring points to analysis engines is continuous. This timing scheme used by network based IDSs that gather data from the traffic stream of the network. Real time technique allows faster detection and permits the IDS to take actions that affect the detected attack.

5.5.4. Control Strategy

Control Strategy dictates the behaviour of the IDS elements and controls the management, the inputs and outputs of that said IDS.

5.5.4.1. Centralized

centralized control strategies, a central location directly controls all monitoring, detection and reporting.

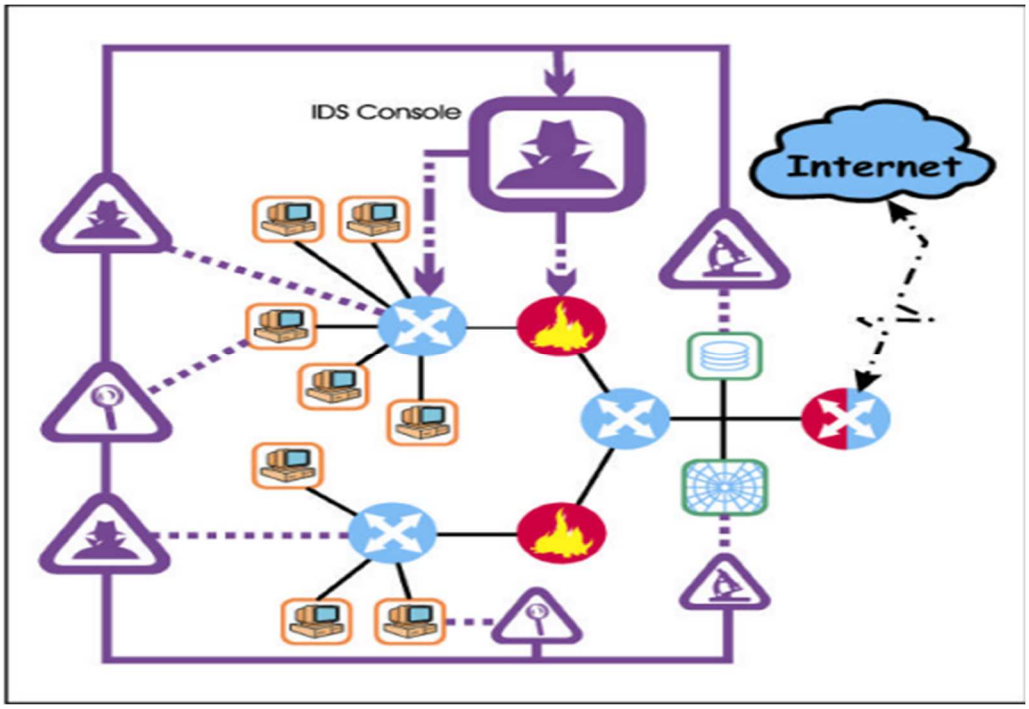


Figure 2.3: Central control strategy

5.5.4.2. Partially Distributed

A local control node controls the monitoring and detection, whereas reporting is controlled by one or more central location(s).

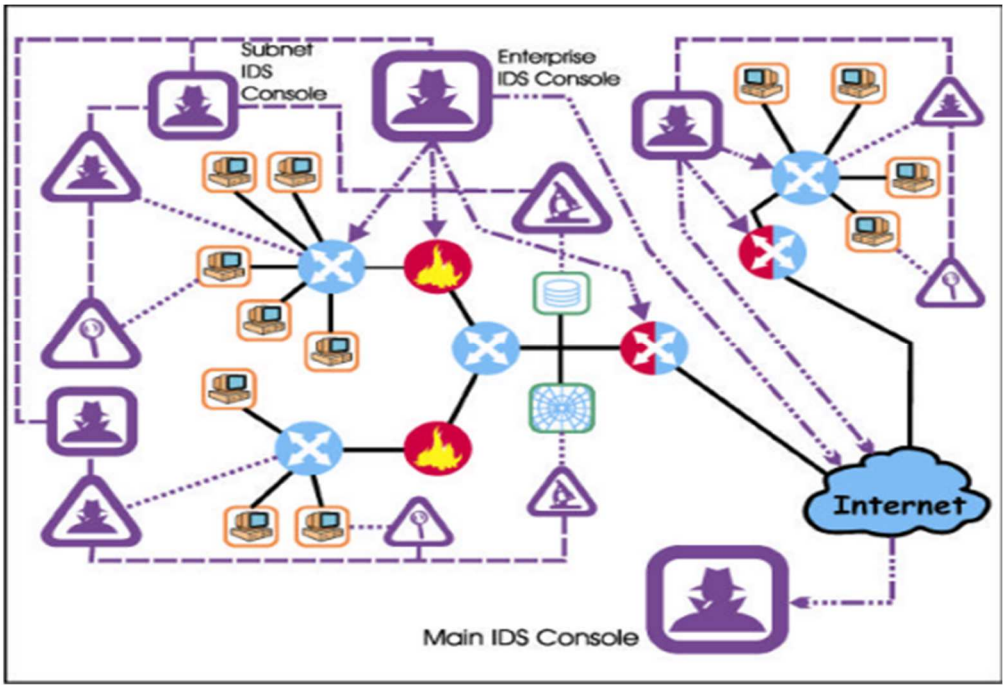


Figure 2.4: partially distributed control strategy

5.5.4.3. Fully Distributed

Using an agent-based approach to manage monitoring and detection. Response decisions are made at the point of analysis.

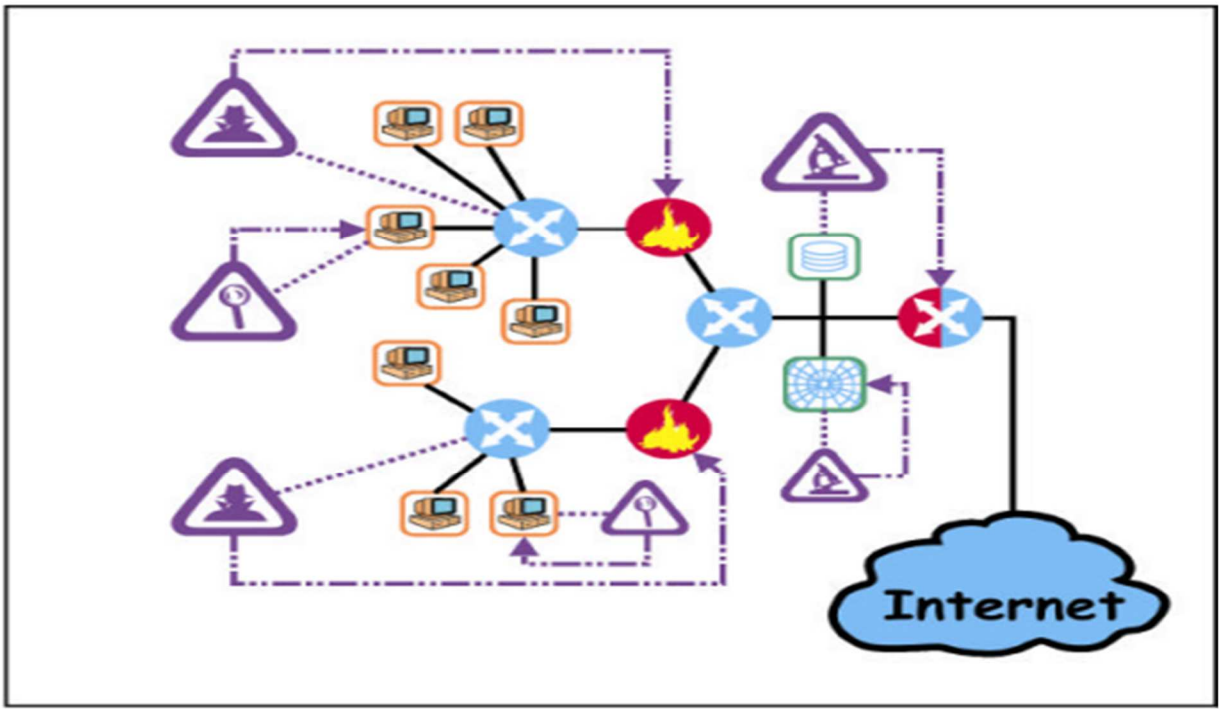


Figure 2.5: fully distributed control strategy

6. Components and architecture [18]

6.1. Typical Components

The typical components of and IDS are sensors or agents, management server, database server and consoles.

6.1.1. Sensors or agents

Gathers information on the evolution of the state of the system and provides a sequence of events that reflect this evolution (memoir main). There are three types of sensors based on their functionality system sensors, network sensors and application sensors. The term sensors are used for network based IDSs whereas the term agent is used for host based IDSs.

6.1.2. Management Server

This server receives and manages the data sent by the agents or the sensors. The management servers can perform analysis on the provided data and can identify malicious events within the data provided that the sensors or the agent alone cannot.

6.1.3. Database Server

The database server is a repository for event information recorded by sensors, agents and management servers.

6.1.4. Console

A Console is a program that is used to present the user or the administrator of the IDS with an interface. Consoles are typically used for monitoring and analysis however some IDS consoles are capable of providing both administration and monitoring services. The consoles that are responsible for the sensors and agents' configurations and applying software updates can be used strictly by an administrator.

6.2. Network Architecture

The components of the IDS can be connected to each other through a standard network or a separate network that is designed for security software management called a management network. A management network helps to protect the IDS from attack and to ensure it has sufficient bandwidth under any conditions. Each sensor or agent host has an additional network interface known as a management interface that connects to the management network. Also, each sensor or agent host is unable to pass any traffic between its management interface and any of its other network interfaces. The management servers, database servers, and consoles are located on the management network only. This architecture effectively isolates the management network where the management servers, database servers and consoles are attached from the production networks. The benefits of doing this are to conceal the existence and identity of the IDPS from attackers.

6.3. Deployment of IDS

IDS is a necessary addition to every computer network security infrastructure. However, due to the deficiency of the current IDS product and the weak skill level of most security

administrators, an effective deployment of an IDS requires a good strategy, preparation, testing and a specialized training.

Before deployment, it is important to have a proper configuration for the IDS. For example, a configuration designed for Windows OS will prove ineffective in detecting intrusion attempts on a Linux machine.

A good deployment strategy consists of using both NIDS and HIDS to protect large scale networks, starting by network-based IDSs as they are the simplest to install and maintain. Next protect critical servers with host-based IDSs.

6.3.1. Deploying Network-based IDS

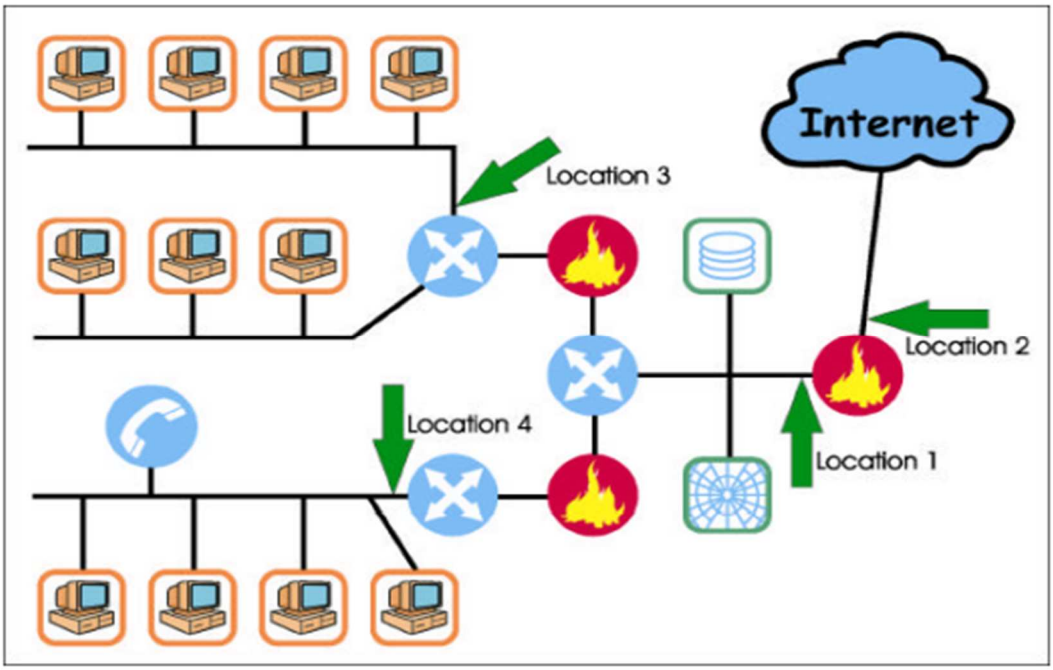


Figure 2.6: IDS locations in a network architecture

6.3.1.1. Location 1: Behind each external firewall, in the DMZ

Advantages:

- Identifies the shortcomings of the network firewall policy or performance.
- Sees attacks that might target web server or ftp server which commonly reside in the DMZ.

- Sees attacks that generated from outside and penetrated the network security perimeter.
- The IDS can sometimes identify the outgoing traffic coming from the compromised server even if the attack went undetected.

6.3.1.2. Location 2: Outside an external firewall

Advantages:

- Documents the number of attacks coming from the internet targeting the network.
- Documents the type of attacks coming from the internet that targets the network.

6.3.1.3. Location 3: On major network backbone

Advantages:

- Monitors a large amount of network traffic which increase the possibility of detecting attacks.
- Detects unauthorized activity by authorized users within the organization's security perimeter.

6.3.1.4. Location 4: On critical subnets

Advantages:

- Detects attacks targeting critical systems and resources.
- Allows focusing of limited resources to the network assets considered of greatest value.

6.3.2. Deploying Host-Based IDSs [15]

After properly deploying network-based IDSs, adding host-based IDSs can offer a higher level of security for the system, however installing the IDS on each host on the network can prove to be extremely time-consuming, as each IDS has to be installed and properly configured to the requirements of each specific host.

It is recommended to prioritize the critical servers when installing host-based IDSs. This will decrease the deployment costs and allow personnel to focus on the alarms generated from the

most important hosts on the network. Once this operation is routine, you can move on to installing host based IDSs on the rest of the hosts.

Much of the effectiveness of any IDS, but particularly host-based IDS depends on the operator's ability to tell the difference between true and false alarms. It is best when using host-based IDSs allowing the operators to become familiar with the IDS in a sheltered but active environment. Over a period of time an operator working with an IDS in a particular environment, will gain a sense of what is normal for that environment.

7. Intrusion Detection Approaches

In intrusion detection systems; techniques have been developed for modelling the data and create tables by classifying the modelled data.[14]

7.1. Data Mining approaches

7.1.1. Decision Tree (DT)

Decision tree is one of the classification algorithms in data mining. The decision tree is initially constructed from a set of pre-classified data. Each data item has certain attributes that are used by the DT to classify them. The principle of functioning of the decision tree is to select the appropriate attributes, that best divides the data items into their classes. the data items will then be partitioned Based on the values of their attributes. This process will be recursively applied to each partitioned subset of the data items until all the data items in current subset are in the same class.

7.1.1.1. Structure of DT

Decision Tree consists of various nodes, leaves and edges. A node specifies the attribute that the data will be partitioned based on. Each node has a number of edges that are labelled according to the value of edges and the value of the attribute in the parent node. An edge can connect two nodes or a node and a leaf. Leaves are labelled with a decision value for the categorization of the data.

7.1.1.2. Decision Trees as Intrusion Detection Model

Decision trees can be implemented in Intrusion detection as long as it is considered a classification problem where each connection or user is identified as an intrusion or normal based on some existing data. This classification problem of intrusion detection can be solved

by DT as they learn the model from the data set and can classify the new data items into one of its specified classes. Decision trees can be used as well for misuse detection since they are able to learn a model based on training data then predict the future data as an intrusion or normal based on the learned model. One of decision trees most important features is that they work well with large datasets. Since large amounts of data flow across computer networks, their high performance makes them suitable for real-time intrusion detection. There can be some new attacks on the system which are small variations of known attacks after the intrusion detection models are built and finalized. However, decision trees are able to detect these new attacks due to its property "generalization accuracy".

7.1.2. Support Vector Machine

It is the most preferred method for intrusion detection systems. Used for selection of feature vector. Support Vector Machines aims to distinguish between data from two classes in a most appropriate way with a feature vector. They are used in many classification problems such as face recognition systems, sound analysis.

An SVM is a supervised learning method. It classifies data by constructing an N-dimensional hyperplane that separates the data into various categories. In the basic classification, SVM classifies the data into two categories labelled in pairs {(x, y)}, where y is the label of instance x, SVM works by maximizing the margin to obtain the best performance in classification [22].

7.1.2.1. SVM Intrusion Detection System [23]

The construction of an SVM intrusion detection system consists of three phases:

* Preprocessing: using automated parsers to process the randomly selected raw TCP/P dump data in to machine-readable form.

* Training: in this process SVM is trained on different types of attacks and normal data. The data have input features that fall into two classes: normal (+1) and attack (-1).

* Testing: measure the performance on testing data.

7.1.2.2. Development of SVM IDS

The data is first partitioned into two classes: normal and intrusive, where intrusive represents a collection of different attacks. The objective is to separate normal (1) and intrusive (-1) patterns.

* Training: The SVMs are trained with normal and intrusive data. The processed data consists of **D** data points: **Tr** for training, **Te** for testing. Each point is located in the N-dimensional space, with each dimension corresponding to a feature of the data point. We use a training set of **Tr** data points with **F** number of features. Data points contain actual attacks and normal usage patterns.

* Testing: The SVMs are tested by applying them to a set of intrusion data consisting of **D** data points with **F** features. We use the SVMs to tell the difference between normal and intrusive behaviour.

*Evaluating [24]: For evaluating the SVMs we use the following metrics:

Accuracy: Accuracy is calculated as "the total number of two correct predictions, True Positive (TP) + True Negative (TN) divided by the total number of a dataset Positive (P) + Negative (N)".

$$Accuracy = \frac{TP + TN}{P + N}$$

Precision: Precision is computed as "the number of correct positive predictions (TP) divided by the total number of positive predictions (TP + FP)". Precision is also known as a positive predictive value.

$$Precision = \frac{TP}{TP + FP}$$

Recall: Recall is computed as "the number of correct positive predictions (TP) divided by the total number of positives (P)". Recall is also known as the true positive rate or sensitivity.

$$Recall = \frac{TP}{P}$$

7.2. Deep Learning in IDS

Deep Learning is advanced machine learning. It consists of multilayers and more deep layers for a Deep Neural Network (DNN). Deep learning is a neural network with more numbers of inputs and more complex neural layers [25]. Deep learning is divided into three types of learning:

- Supervised features learning: used for the extraction of features. These features will be supplied to straightforward machine learning methods for performing tasks such as classification and detection.
- Unsupervised feature learning: uses optimum feature extraction on the entire model.
- Hybrid deep: using generative feature learning models to enhance the training of deep neural networks.

Deep Learning is developing artificial neural networks (ANN). Their main goal is feature learning and classification tasks, as well as finding correlations between features among a large amount of data [26].

7.2.1. Deep Learning Algorithms in IDS

Different deep learning techniques have been used for intrusion detection systems, due to its abilities to analyse and discover information from extensive volume of data. The implementation of deep learning techniques was mainly for features extraction and tasks classification. The classification technique depends on the dataset used for training and testing the model, applied deep learning architecture.

- Generative Architectures (AE, RNN, DBN): Unlabelled data are more numerous than the labelled data. However, deep learning approaches can still be applied to unsupervised learning.
- Discriminative Architectures (CNN): Discriminative method denotes to a class of models used supervised learning data labelled especially in classification task.
- Hybrid Deep Learning: Hybrid architectures implements both generative and discriminative models. A hybrid deep learning model that usefully combines different deep learning methods (LSTM with GRU, BiLSTM, and CNN with other techniques).

7.3. Biological Models

7.3.1. Artificial immune system (AIS)

The human body has a unique property. The human immune system (HIS) is able to protect the human body from malicious bacteria, viruses, parasites and fungi mostly without prior knowledge of the structure of these pathogens [27]. the HIS is seen as form of anomaly detector. Researchers tried to understand the core mechanisms of its function. To imitate its detection and protection capabilities with low false positive and false negative rates, several artificial immune systems have been built for a various types of applications including fraud detection, document classification and host and network-based intrusion detection [28].

Two important mechanisms used in AIS research: network-based models and negative selection models, although this distinction is somewhat artificial as many hybrid models also exist. The first of these mechanisms refers to systems which are largely based on Jerne's idiotypic network theory which recognises that interactions occur between antibodies and antibodies as well as between antibodies and antigens. Negative selection models use the non-self-matching selection process, with similarity to T-lymphocytes in the thymus in order to generate a population of detectors. Building IDS this approach has been used frequently along with some newer more advanced algorithms. The table presents the AIS features for IDS [29]:

Feature	Description
Distributed	Robust: a distributed IDS allocates intrusion detection processes across several hosts. Configurability: each intrusion detection process can be simply tailored for the local requirements of a specific host. Extendibility: The new intrusion detection processes running on different operating systems does not require modification of existing processes.
Lightweight	Efficiency: It places minimal work on each component of the IDS. dynamic features: It dynamically covers intrusion and non-intrusion pattern spaces at any given time instead of keeping the entire intrusion and non-intrusion patterns.
Self-organised	Adaptability: highly adaptive system because there is no need for manual updates of its intrusion signatures as network environments change.
Multi-layered, disposable, diverse	Robust: a multi-layered IDS places different levels of sensors at one monitoring place. A disposable IDS does not depend on any single component. Any component can be easily and automatically replaced with other components. A variety of different intrusion detection processes spread across hosts will slow an attack that has successfully compromised one or more hosts.

Table 2.2: AIS features for IDS

8. Intrusion detection system evaluation

Evaluation of intrusion detection system can prove to be a difficult task due to numerous reasons. First, it is difficult to collect data representative of the threat. Since the threat is constantly changing as new attacks are developing, it is important that an IDS adapts to these changes and developments. It is impossible to make predictions outside one's data, and this is exactly what is expected of IDS evaluation. Second, if real data are used in the evaluation process, the evaluating team cannot be sure if there are any hidden attacks that were not discovered which will negatively impact the evaluating process results by affecting the calculations as well as the probability of detection and false alarms. Third, not so many IDSs are automated so the human factor that involved in operating the IDS should also be included in the evaluating process. The analyst should be treated as part of the system.

8.1. Intrusion detection systems measurable characteristics [30]

8.1.1. Coverage

The coverage of an intrusion detection system are the attacks that an IDS can detect under ideal conditions. The number of dimensions that form each attack makes assessing the coverage of the IDS a difficult task. Different sites consider some attacks at different priorities, which greatly affects the assessment process. For example, streaming services may prioritise detecting distributed denial of service attacks whereas governmental websites prioritize detecting surveillance attacks.

8.1.2. Probability of false alarms:

False alarms are alerts caused by normal non-malicious background traffic. The probability of false alarms is the rate at which the IDS produce false alarm in a known environment. Calculating false alarm rate is a difficult task due to the following reasons:

- An IDS may have different false alarm rates in different network environments.
- The diverse aspects that are associated with host activities and network traffic makes it difficult to determine which aspects are responsible for the false alarms.
- Unable to determine the right configuration for a particular false alarm when using configurable IDS.

8.1.3. Probability of detection

Probability of detection also known as the hit rate is the rate of attacks detected correctly by an IDS in a given environment at a certain period of time. the set of attacks used during the test affects greatly the result of this measurement, that is because the IDSs ability to detect the attack is associated to its ability to identify and categorise them.

Since the probability of detection and probability of false alarms are linked, we can also say that finding the right configuration for a particular hit rate test is a difficult task. Several methods are used to show how an IDS performs under these two measurements. The method that is most commonly used is the Receiver Operating Characteristic curve or ROC curve.

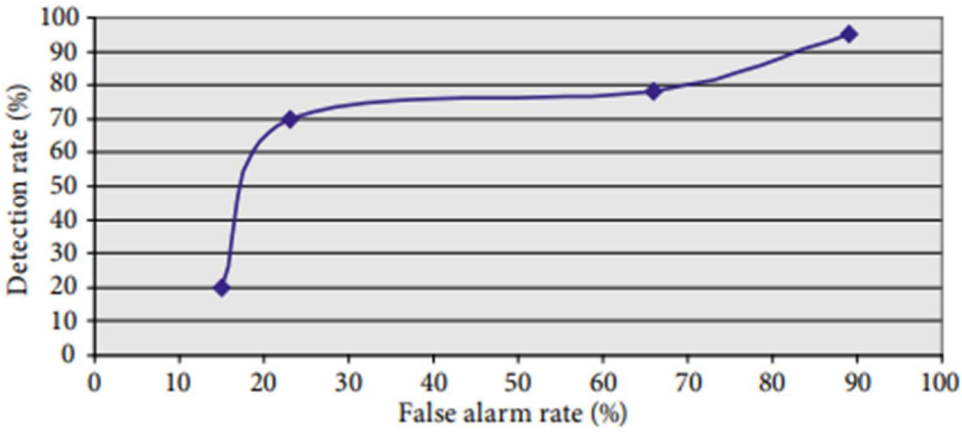


Figure 2.7: ROC curve

8.1.4. Ability to Handle Stressful Network Conditions:

This characteristic revolves around the performance of the IDS when receiving heavy traffic. Most intrusion detection systems are expected to drop packets as the network traffic raises in volume. this can result in attackers exploiting this method to raise the probability of their attacks going undetected by the IDS. The evaluation team is responsible for recognizing the threshold at which the IDS and the monitored system drops in performance.

8.1.5. Ability to Detect Novel Attacks

This characteristic determines how well and IDS detect zeroday attacks. This measurement is designed strictly for anomaly-based IDSs as well as specification-based IDSs. Signature-based IDSs, due to their method of detection they are not subject to this measurement.

8.2. Defeating an IDS [19]

Intrusion detection systems are an excellent security measure however that doesn't mean they are invincible. All types of IDSs can suffer from information overload in their bandwidth. Signature based IDS require frequent updates otherwise they will be ineffective against new vulnerabilities. If a new network application were to be added or altered on the network, anomaly-based IDS must redefine a new normal network state.

the network infrastructure must be suitable for the use of IDS especially in large network environment where multiple types of IDSs are used. Attackers may render IDS ineffective using different methods such as DOS attacks and other IDS focused attacks for example if a hacker overloads a network with decoy attack signatures, he can simultaneously and secretly exploit other code to remain hidden from the IDS.

Another way attacks may render IDS useless is by sending a malicious payload over multiple packets this is called session slicing and it can defeat pattern and signature-based mechanisms. These payloads can also be sent over long time periods exploiting another IDS vulnerability; slow scanning. If the attacker is patient enough many IDS are unable to detect attacks that happen over extended periods of time. IDS can also be bypassed by altering the manner in which network communications and applications function. Using proxy attacks and spoofing are methods that makes the attack traffic appear from internal trusted host therefore it may be ignored by IDS. Even if an IDS is well maintained and updated, the security team still has to respond to the threats properly and quickly.

9. Detection and output of IDSs

9.1. IDS Detections

9.1.1. Scanning attacks

A reconnaissance attack used by hackers to analyse the network for any vulnerabilities or entry points.

9.1.2. Denial of service

Dos attacks violate the availability of a resource. Dos is one which degrades or completely disables a server, host or a network. Attackers try to prevent legitimate users access to a service.

9.1.3. Penetration attack

Penetration attacks violate the integrity and control of system resources and privileges.

- **Remote to Local:** An attacker who does not have a user account but can send packets to it over a network gains access to the machine.
- **User to Root:** An attacker who has a legitimate user account on the machine can exploit vulnerabilities or bug on the computer system to misuse or elevate his privileges.

9.1.4. Excessive attack reporting [17]

IDSs that report separate attack each time an attacker accesses a different host will generate a thousand report if an attacker were to scan a subnet of a thousand host. It is impossible for the IDS operator to handle this much of reports. Modern IDSs filters the high number of reports and present the operator with attacks of highest importance first.

9.2. Evasion Techniques

Many evasion techniques can be used to evade detection by signature-based detection systems, such as packet splitting, Denial of service, Payload, and shellcode mutation. The process of detecting attacks disguised by evasion techniques can prove to be a difficult task for signature-based IDS. In this part, we focus on the effectiveness of shellcode mutation against signature-based IDSs. [16]

9.2.1. Shellcode Mutation

A. Shellcode

Shellcode is a set of instructions injected and then executed by an exploited program. The shellcode is used to override the flow of execution after exploiting a vulnerability process, so it is generally written in assembler and translated into hexadecimal opcodes. [20]

B. Polymorphic Shellcode

Fundamentally, there are two approaches used by IDSs to intrusion detection: signature detection and anomaly detection. When a signature of the shellcode already exists in IDS database and is reused, or when the shellcode behaves in a pattern that is considered as malicious. Then the IDS will easily detect and prevent the attack. This is where polymorphic shellcode takes place to avoid detection by the anti-virus, and the intrusion detection system

by changing the exploit signature every time it executed without changing the main function and the end result.

9.2.2. Evasion Tools

There are many well-known open-source tools than can be used to evade IDS and AV systems, by exploiting a number of vulnerabilities and techniques.

1) Metasploit

Metasploit is not just a single tool. It is a complete framework. The Metasploit project is used by network administrators to test the security level of their own network. This project's primary function is penetration testing however it can also be used to develop exploits codes and build malware signatures. Moreover, it provides a number of polymorphic shellcode encoders as it also provides a connection with the target host in case of a successful attack.

2) MSFVenom

MSFVenom is an open-source penetration testing tool which combines two other tools. The MSFpayload component of Metasploit that allows to generate shellcode, executables and much more for use in exploits outside the framework and MSFencode that helps avoid bad characters and evade AV and IDS by encoding the original payload generated by the MSFpayload in a way that doesn't contain bad characters.[21]

3) Veil-evasion

Veil-Evasion is a tool designed to generate Metasploit payloads that bypass common anti-virus solutions. It works by creating and transforming the malicious script and hides it within another file then it converts that said file to an encrypted exe file.

9.3. IDS output

When an IDS detects an attack, it sets off an alarm that contains a certain set of information for the operator.

9.3.1. Typical IDS output

The IDS output presents a brief summary of the state of the attack, it provides:

- Time & Date the attack took place,
- Sensor IP address,

- Vendor specific and standard attack name,
- Source and destination IP address and port numbers and
- Network protocol used in the attack.

In order to help the IDS operator correctly assess the situation and the impact of the attack, the IDS also provides a brief description of each type of attack that contains the following:

- severity level.
- The damage caused.
- The type of vulnerability exploited.
- List of software types that are vulnerable.
- Patch information.

10. Existing IDSs

Various types of IDSs were created over the years with each have a certain set of features and capabilities. Here is a list of most commonly used IDS tools:

Tool	Platform	*IDS*	Features
Solarwind [31]	Windows	NIDS	Can log messages generated by Windows PCs and by Mac-OS, Linux, and Unix computers, manages data gathered by snort, it can receive network data in real-time from Snort.
Bro [32]	Unix, Linux, Mac-OS	NIDS	Traffic logging and analysis, provides visibility across packets, event engine, ability to monitor SNMP traffic, ability to track FTP, DNS, and HTTP activity.
OSSEC [33]	Linux, Mac-OS, Windows	HIDS	Free to use open-source HIDS security, ability to detect any alterations to the registry on Windows, ability to monitor any attempts to get to the root account on Mac-OS, log files covered include mail.
Snort [34]	Unix, Linux, Windows	NIDS	Packet sniffer, packet logger, threat intelligence, signature blocking, real-time updates for security signatures, in-depth reporting, ability to detect a variety of events including OS fingerprinting.
Suricata[35]	Linux, Mac-OS, Windows	NIDS	Collects data at the application layer, ability to monitor protocol activity at lower, real-time tracking for network applications, integration with third-party tools, built-in scripting module,
Security Onion[36]	Linux, Mac-OS	NIDS, HIDS	Complete Linux distribution focusing on log management, and intrusion detection, runs on Ubuntu, integrates elements from several front-end analysis tools. It includes HIDS functions as well.

Table 2.3: the most commonly used IDS tools

11. Conclusion

In the multiple technologies of computer security defence, IDS occupy a place of excellence. This is due to the fact that they allow traffic analysis, the only means in the absence of prior knowledge, to detect possible attacks against the systems. We have since the IDS differs according to different criteria of design and use.

Chapter 03:

Deployment of Snort IDS

1. Introduction

In this chapter We are going to present Snort and then, we choose our work environment before we start describing the implementation process.

2. Introduction to Snort

2.1. Snort Modes

Snort operates in two basic modes: packet sniffer mode and NIDS mode. It can function as a packet sniffer, like tcpdump or snoop. Snort can also log these packets to a log file when working under this mode. However, using snort for this purpose is not very useful since there are many other efficient tools for packet logging [37]. When using snort in NIDS mode, it uses its sets of rules to find out if there is any network intrusion detection activity.

2.1.1. Network Sniffer Mode

In the network sniffer mode, snort functions the same as a commonly used program tcpdump. It captures and displays network packets with different levels of detail on the console. Running snort in the sniffing packet mode does not require a configuration file.

2.1.2. Network Intrusion Detection Mode

In the intrusion detection mode snort does not log each captured packet. Rather, it applies a certain set of rules on all captured packets. only after a packet matches a rule it is then logged or an alert is generated. Otherwise, the packet will be dropped and no log entry is created. Snort requires a configuration file that contains snort rules or a reference to other files that contains those said rules. The typical name of a snort configuration file is snort.lua. The following command launches snort in intrusion detection mode: snort -c /usr/local/etc/snort/snort.lua When this command is executed, snort will read the configuration file as well as the files referenced in it. After reading these files, snort will build its internal data structures and rule chains. The captured packets will then be matched against the rules set and the appropriate action will take place based on the configuration file.

2.2. Snort Alert Modes

When snort runs as a Network Intrusion Detection System, it can send alerts in many different modes. These alert modes are configurable through the command line or the

snort.lua file. We are going to use the following rule as an example to differentiate between these modes:

alert icmp any any -> any any (msg: "Ping with TTL=100"; \ ttl:100;)

Whenever an ICMP packet is captured and matched to this rule, snort will send an alert based on the mode that it is configured for.

2.2.1. Fast Mode

The fast alert mode logs the alert with the following information:

- Timestamp
- An alert message (depends on the rule that matches the captured packet)
- Source and destination IP/Ports

To configure snort for fast alert mode, you have to use the "-A fast" command line option. The full command to start snort in fast alert mode is:

/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -A alert_fast

Following the previous rule as an example, the alert message should be similar to the following:

```
05/28-22:16:25.126150  [**] [1:0:0] Ping with TTL=100 [**]
{ICMP} 192.168.1.100 -> 192.168.1.3
```

Figure 3.1: -A alert_ fast output

Where it presents:

- 05/28-22:16:25.126150 as Timestamp.
- "Ping with TTL=100" as the alert message.
- 192.168.1.100 and 192.168.1.3 as the source and destination IP addresses.
- ICMP as the packet type.

2.2.2. Full Mode

Full alert mode is the default alert mode. It logs the alert message with the packet header. To start snort in full alert mode, you have to use the "-A full "command line option. The full command is as follows: /usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -A alert_full

Snort generated alerts in this mode are similar to the following:

```
[**] [1:0:0] Ping with TTL=100 [**]
05/28-22:14:37.766150 192.168.1.100 -> 192.168.1.3
ICMP TTL:100 TOS:0x0 ID:40172 IpLen:20 DgmLen:60
Type:8  Code:0  ID:768   Seq:20224  ECHO
```

Figure 3.2: -A alert_full output

Along with the typical alert message it also provides:

- TTL (time to live) value in the IP packet header.
- TOS (type of service) in the IP packet header.
- Length of IP packet header as IPLen: 20.
- Full length of IP packet as DgmLen: 60.
- ICMP type field and code value.
- IP packet ID.
- Sequence number.

2.2.3. UNIX Socket Mode

Using the command line option "-A unsock" you can send alerts to any other application using UNIX sockets. This mode is mainly used to process alerts using custom applications with snort.

2.2.4. No Alert Mode

This mode is used for high-speed intrusion detection using unified logging. Snort alerts are completely disabled using the command line option "-A none". You can also disable normal logging using this option while using the unified option [38]. The following table summarize Snort modes output:

Option	Description
-A fast	Fast alert mode. Writes the alert in a simple format with a timestamp, alert message, source and destination Ips/ports
-A full	Full alert mode. This is the default alert mode and will be used automatically if you do not spercify a mode
-A unsock	Sends alerts to a UNIX socket that another program can listen on
-A none	Turns off alerting

Table 3.1: Snort modes output

2.3. Snort architecture

Snort is an elegantly designed little beast, made up of several components that each perform specific tasks.

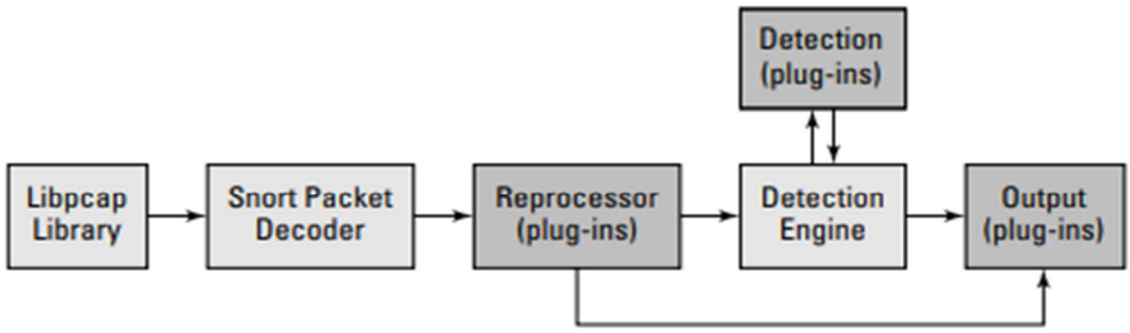


Figure 3.3: Snort architecture

Each packet captured by snort follows this specific path:

1. **Packet capture library:** the packet capture library is a piece of software that sends Snort network packets from network cards. Snort uses libpcap library on Linux as well as Unix system whereas, on windows it uses WinPcap library.
2. **Packet Decoder:** the packet decoder takes packets from different network interfaces and prepares them to be preprocessed or send them to the detection engine.
3. **Preprocessor**: Snorts preprocessor has several plug-ins that can be turned on or off. Preprocessing works on the decoded packets, making several different changes and operations such as packet defragmentation making the data easier to digest for snort. Preprocessors can also alert on, classify, or drop a packet before sending it to the detection engine

4. **Detection Engine:** the detection engine is the most important part of snort. It takes information from the packet decoder and Preprocessors and employs snort rules for the purpose of intrusion detection. The rules are read into internal data structures or chains where they are matched against all packets. When a packet matches a rule, appropriate action is taken based on the rule it matched to. Appropriate action maybe logging the packet or generating alerts.
5. **Output Modules:** output modules can do different operations depending on how you want to save output generated by snort. Basically, these modules control the type of output generated whenever a preprocessor or a rule is triggered. Based on their configuration, these modules can do thing like the following:
   - Simply logging to /var/log/snort/alerts file or some other file.
   - Sending messages to syslog facility.
   - Logging to a database like MySQL or Oracle.
   - Modifying configuration on routers and firewalls.

3. Snort Rules

Snort is an incredible tool, but the engine is only a part of what makes snort a useful tool. The rules are where the true strength of snort lies. This is an example of a snort rule:

alert icmp any any -> any any (msg: "Ping with TTL=100"; \ ttl:100;)

3.1. Structure of a rule

All snort rules are divided to two logical parts rule header and rule option as demonstrated in the figure:

Rule Header	Rule Options

The rule header provides information about the action the rule takes, and the rule option provides an alert message and which part should be used to generate it. Both the header and the options part contain criteria for matching rules against packets. The rules that are capable of detecting more than a single type of intrusion are called intelligent rules. [39]

3.2. Rule Header

All of snort rules contain header information. The rule header provides the action taken by the rules as well as a filter to separate traffic using five key sifting factors: source IP address,

source Port, destination IP address, destination Port and protocol. The following figures presents all the components as well as their order in the structure of the rule header:

Action	Protocol	Address	Port	Direction	Address	Port

3.2.1. Rule Actions

The action is the first part of the rule of a rule. It shows what action will be taken when the rule matches to a packet. The action will not be taken unless all of the rule conditions are met. There are five predefined actions. However, you can also provide your action if needed.

1. Pass: this action makes snort ignore the packet. This action is used to speedup snort in scenarios where you don't need to check on certain packets. For example, vulnerability assessments.
2. Log: this action tells snort to log the packet. Packets can be logged in different ways with different levels of information.
3. Alert: the alert action is used to send alerts when rule conditions are true for a certain packet. Alerts can be sent in multiple ways. For example, you can send an alert to file or to a console.
4. Activate: the activate action is used to generate an alert before activating another rule to check for other conditions. The rules activated by the activate action are called dynamic rules.
5. Dynamic: Dynamic rules can only be invoked by the activate action and they are not usually applied on a packet.

3.2.2. Protocols

Protocol is the second part of a snort rule. Currently snort understands four types of protocols IP, ICMP, TCP, UDP.

3.2.3. Address

There are two address parts in a Snort rule. These addresses are used to check the source from which the packet originated and the destination of the packet. The address can be a single IP address as well as a network address. You can use "any" keyword to apply the rule on all

addresses. Snort also provides a mechanism to exclude addresses by using a negation symbol "! ". you can also specify a full list of addresses in a single rule.

3.2.4. Port Number

There are also two port parts in a snort rule. These port numbers are used to apply a rule on packets that originate from or go to a certain port or a range of ports. When creating a list of ports, you can use only the starting port number or the ending port number in the range by using the symbol ":".

3.2.5. Direction

the direction field determines the source and destination addresses and port numbers in a rule.

- This -> symbol shows the source address and port number are on the left side.
- This <- symbol shows the destination address and port number are on the left side.
- This <> symbol shows the rule will be applied to packets in both directions.

3.3. Rule Options

Rule option is the second part of a snort rule. It follows the rule header and it comes enclosed in a parenthesis. There may be a single option as well as multiple options. If multiple options exist, they are separated with a semicolon";" and treated with the logical AND. The action in the rule header will not be activated unless all the criteria in the rule options are true. In general, an option may have two parts: a keyword and an argument separated with ":". For example, we'll take a rule option from our previous example: msg: "Ping with TTL=100";

3.3.1. General rule options

These options provide information about the rule but do not have any affect during detection:

Keyword	Description
msg	The msg keyword tells the logging and alerting engine the message to print with the packet dump or alert.
reference	This keyword allows rules to include references to external attack identification systems.
gid	The gid keyword (generator id) is used to identify what part of Snort generates the event when a particular rule fires.
sid	The sid keyword is used to uniquely identify Snort rules. This information allows output plugins to identify rules easily. This option should be used with the rev keyword.
rev	The rev keyword is used to uniquely identify revisions of Snort rules.
classtype	The classtype keyword is used to categorize a rule as detecting an attack that is part of a more general type of attack class.
priority	The priority keyword assigns a severity level to rules. Default priority is assigned by classtype rule.
metadata	The metadata keyword allows a rule writer to embed additional information about the rule, typically in a key-value format.

Table 3.2: General rule options

3.3.2. Payload detection rule options

These options all look for data inside the packet payload and can be inter-related:

Keyword	Description
content	The content keyword allows the user to set rules that search for specific content in the packet payload and trigger response based on that data.
rawbytes	The rawbytes keyword allows rules to look at the raw packet data, ignoring any decoding that was done by preprocessors.
depth	The depth keyword allows the rule writer to specify how far into a packet Snort should search for the specified pattern.
offset	The offset keyword allows the rule writer to specify the starting search point for a pattern within a packet. offset modifies the previous 'content' keyword in the rule.
distance	The distance keyword allows the rule writer to specify how far into a packet Snort should ignore before starting to search for the specified pattern relative to the end of the previous pattern match.
within	The within keyword is a content modifier that makes sure that at most N bytes are between pattern matches using the content keyword.
fast_pattern	The default behavior of fast pattern determination is to use the longest HTTP buffer content. If no HTTP buffer is present, then the fast pattern is the longest content
uricontent	The uricontent keyword in the Snort rule language searches the normalized request URI field.

Table 3.3: Payload detection rule options

3.3.3. Non-payload detection rule options

These options look for non-payload data:

Keyword	Description
fragoffset	The fragoffset keyword allows one to compare the IP fragment offset field against a decimal value.
ttl	The ttl keyword is used to check the IP time-to-live value.
tos	The tos keyword is used to check the IP TOS field for a specific value.
id	The id keyword is used to check the IP ID field for a specific value.
ipopts	The ipopts keyword is used to check if a specific IP option is present.
fragbits	The fragbits keyword is used to check if fragmentation and reserved bits are set in the IP header.
dsize	The dsize keyword is used to test the packet payload size
flags	The flags keyword is used to check if specific TCP flag bits are present.
flow	The flow keyword allows rules to only apply to certain directions of the traffic Flow.
flowbits	he flowbits keyword allows rules to track states during a transport protocol session.
seq	The seq keyword is used to check for a specific TCP sequence number.
ack	The ack keyword is used to check for a specific TCP acknowledge number
window	The window keyword is used to check for a specific TCP window size.
itype	The itype keyword is used to check for a specific ICMP type value.
icode	The icode keyword is used to check for a specific ICMP code value.

Table 3.4: Non-payload detection rule options

3.3.4. Post-detection rule options

These options are rule specific triggers that happen after a rule has “fired.”:

Keyword	Description
logto	The logto keyword tells Snort to log all packets that trigger this rule to a special output log file. this option does not work when Snort is running in binary logging mode.
session	The session keyword is built to extract user data from TCP Sessions.
resp	The resp keyword is used attempt to close sessions when an alert is triggered.
react	This keyword implements an ability for users to react to traffic that matches a Snort rule by closing connection and sending a notice.
replace	Replace the prior matching content with the given string of the same length. Available in inline mode only.
detection filter	Track by source or destination IP address and if the rule otherwise matches more than the configured rate it will fire.

Table 3.5: post-detection rule options

4. Work environment

4.1. HOST:

HP EliteBook, processor Intel(R) Core (TM) i7-5600U CPU @ 2.60GHz, 8 GB of RAM and a 230 GB SSD hard drive, running Windows 10.

4.2. Operating System:

Our operating system of choice is Ubuntu 22.04 LTS 64bit running on a VirtualBox. Ubuntu is a free Linux distribution based on Debian and composed mostly of open-source software.

4.3. Intrusion Detection System:

We have chosen SNORT for our Intrusion Detection System in this project for the following reasons:

- Snort is an open-source IDS.
- One of the most used most popular IDS on the market.
- It has a large community that contributes to its development and success.
- An extensive list of signatures provided by the community members of Snort.

4.4. Deploying the IDS:

For deploying Snort, we chose position 3 already explained in chapter 2 section 6.3.1 Deploying Network-based IDS.

5. Snort 3 on Ubuntu 22

In this part we are going to describe the process of installing Snort 3 as a network intrusion detection system with Splunk as a security information and event manager (SIEM) on the Operating System Ubuntu22.04 LTS 64 bit.

Snort 3: is a free and open-source network intrusion prevention system (NIPS) and network intrusion detection system (NIDS), created by Martin Roesch in 1998.

PulledPork

Pulledpork is a tool that detects the latest rule set released by snort and automatically downloads and integrates it to the snort.rules files.

Splunk

Splunk is the software we will be using as our SIEM (Security information and event management) solution, which will display graphically (through a web interface) all the alerts Snort has generated, and will give us some powerful tools to search and understand those alerts, as well as draw deeper information from them.

OpenAppId

OpenAppId allows for the identification of application layer (layer 7) traffic. You can create rules that operate on application-layer traffic (for example to block Facebook or a specific type of VPN), and to log traffic statistics for each type of traffic detected.[40]

5.1. Installing snort

For the installation of the project, we are going to follow the official manual provided by the project's owners on their website snort.org [37].

First, ensure your system is up to date and has the latest list of packages:

sudo apt-get update && sudo apt-get dist-upgrade -y

Make sure your system has the correct time and the correct time zone. This command allows you to choose your time zone:

sudo dpkg-reconfigure tzdata

prepare the installation directory:

mkdir ~/snort_src

cd ~/snort_src

install the snort 3 required dependencies and some optional but highly recommended tools:

sudo apt-get install -y build-essential autotools-dev libdumbnet-dev libluajit-5.1-dev libpcap-dev \ zlib1g-dev pkg-config libhwloc-dev cmake liblzma-dev openssl libssl-dev cpputest libsqlite3-dev \ libtool uuid-dev git autoconf bison flex libcmocka-dev libnetfilter-queue-dev libunwind-dev \ libmnl-dev ethtool

download and install safec. It is optional but good for runtime bounds checks on certain C-library calls:

cd ~/snort_src

wget https://github.com/rurban/safeclib/releases/download/v02092020/libsafec-02092020.tar.gz

tar -xzvf libsafec-02092020.tar.gz

cd libsafec-02092020.0-g6d921f

./configure

make

sudo make install

install PCRE: Perl Compatible Regular Expressions. Pcre is required for regular expression pattern matching. We will not be using the Ubuntu repository because it has an older version:

cd ~/snort_src/

wget https://sourceforge.net/projects/pcre/files/pcre/8.45/pcre-8.45.tar.gz

tar -xzvf pcre-8.45.tar.gz

cd pcre-8.45

./configure

make

sudo make install

The final requirement is to download (but don't install) the Boost C++ Libraries:

cd ~/snort_src

wget https://boostorg.jfrog.io/artifactory/main/release/1.77.0/source/boost_1_77_0.tar.gz

tar -xvzf boost_1_77_0.tar.gz

Next, download and install Data Acquisition library (DAQ) from the Snort website. Note that Snort 3 uses a different DAQ than the Snort 2.9. series. Libdaq 3.0.8 is the latest version at the time of installation.

cd ~/snort_src

wget https://github.com/snort3/libdaq/archive/refs/tags/v3.0.8.tar.gz -O libdaq-3.0.8.tar.gz

tar -xzvf libdaq-3.0.8.tar.gz

cd libdaq-3.0.8

./bootstrap

./configure

make

sudo make install

Update shared libraries:

sudo ldconfig

Now we download, compile, and install Snort 3 from the snort website. If you want to enable additional compile-time functionality, such as the ability to process large (over 2 GB) PCAP files, or the new command line shell: you should run ./configure cmake.sh --help to list all optional features, and append them to the ./configure_cmake.sh command below.

Download and install, with default settings:

cd ~/snort_src

wget https://github.com/snort3/snort3/archive/refs/tags/3.1.31.0.tar.gz -O snort3-3.1.31.0.tar.gz tar -xzvf snort3-3.1.31.0.tar.gz

cd snort3-3.1.31.0

./configure_cmake.sh --prefix=/usr/local --enable-tcmalloc

cd build

make

sudo make install

Snort should be installed un **usr/local/**. to verify that Snort runs correctly along with configuration file run the following command:

/usr/local/bin/snort -V

The following picture presents a standard and successful installation of snort

```
badkarma@badkarma-VirtualBox:~$ /usr/local/bin/snort -V

   ,,_     -*> Snort++ <*-
  o"  )~   Version 3.1.31.0
   ''''    By Martin Roesch & The Snort Team
           http://snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights rese
rved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using DAQ version 3.0.8
           Using LuaJIT version 2.1.0-beta3
           Using OpenSSL 3.0.2 15 Mar 2022
           Using libpcap version 1.10.1 (with TPACKET_V3)
           Using PCRE version 8.45 2021-06-15
           Using ZLIB version 1.2.11
           Using LZMA version 5.2.5
```

Now let's test snort with the default configuration file:

snort -c /usr/local/etc/snort/snort.lua

the output should be as follows:

```
Snort successfully validated the configuration (with 0 warnings).
o")~   Snort exiting
badkarma@badkarma-VirtualBox:~$
```

5.2. Configuring network card

Modern network cards use offloading (LRO for one example) to handle network packet re-assembly in hardware, instead of software.For a NIDS, we want to disable LRO and GRO, since this can truncate longer packets. We need to create a SystemD service to change these settings. First determine the name(s) of the interfaces you will have snort listen on using ifconfig:

```
badkarma@badkarma-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.103  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::3044:fb29:3a71:d81c  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:c6:f1:3a  txqueuelen 1000  (Ethernet)
        RX packets 139250  bytes 148711797 (148.7 MB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 60055  bytes 6809603 (6.8 MB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Once you know the name of the network interface that Snort will listen for traffic on: check the status of large-receive-offload (LRO) and generic-receive-offload (GRO) for those interfaces. In the example below, my interface name is **enp0s4**. We use ethtool to check the status:

```
badkarma@badkarma-VirtualBox:~$ sudo ethtool -k enp0s3 | grep receive-offload
generic-receive-offload: on
large-receive-offload: off [fixed]
```

As you can, the GRO is enabled, and LRO is disabled (the 'fixed' means it can not be changed). We need to ensure that both are set to 'off' (or 'off [fixed]'). We could use the ethtool command to disable LRO and GRO, but the setting would not persist across reboots. The solution is to create a systemD script to set this every time the system boots up, using the following command:

sudo gedit /lib/systemd/system/ethtool.service

[Unit]

Description=Ethtool Configration for Network Interface

[Service]

Requires=network.target

Type=oneshot

ExecStart=/sbin/ethtool -K enp0s3 gro off

ExecStart=/sbin/ethtool -K enp0s3 lro off

[Install]

WantedBy=multi-user.target

the file is NOW created, enable and start the service:

sudo systemctl enable ethtool

sudo service ethtool start

5.3. Installing PulledPork

PulledPork is a tool that we will use to download rulesets, which are the latest rules files that snort/Talos releases to ensure that your system can detect the latest attacks. The original PulledPork is written in Perl, and has worked wonderfully for years supporting Snort 2. With Snort 3, it was decided to re-write PulledPork in Python 3 and not add Snort3 functionality to the original PulledPork. You need to decide if you want to use the original PulledPork or the new PulledPork3 for ruleset management; each have strengths and weaknesses. In the near future you'll only want to use PulledPork3, but it's still in Beta and may not be the correct solution right now, depending on your needs. Since PulledPork 3 is still in beta version we decided to use PulledPork original the stable, but with less features version of PulledPork. Start by registering an account on the Snort website to get a unique your oinkcode before continuing, as the oinkcode is required for the most popular free ruleset.

Our oinkcode for this installation will be: 83e1c6e7392f59e67ea2ff1b7aab255ea17ca8a8

Install the PulledPork pre-requisites:

sudo apt-get install -y libcrypt-ssleay-perl liblwp-useragent-determined-perl

Next, download the latest version of PulledPork and install it by copying the perl file to /usr/local/bin and the needed configuration files to /usr/local/etc/pulledpork:

cd ~/snort_src

wget https://github.com/shirkdog/pulledpork/archive/master.tar.gz -O pulledpork-master.tar.gz

tar xzvf pulledpork-master.tar.gz

cd pulledpork-master/

sudo cp pulledpork.pl /usr/local/bin

sudo chmod +x /usr/local/bin/pulledpork.pl

sudo mkdir /usr/local/etc/pulledpork

sudo cp etc/*.conf /usr/local/etc/pulledpork

Test that PulledPork runs by running it with the -V flag as done below, looking for the output below:

```
badkarma@badkarma-VirtualBox:~/snort_src$ /usr/local/bin/pulledpork.pl -V
PulledPork v0.8.0 - The only positive thing to come out of 2020...well this and
 take-out liquor!
```

Now that we are sure that PulledPork runs, we need to configure it:

sudo gedit /usr/local/etc/pulledpork/pulledpork.conf

line 19, we need to change the URL, and then replace with the oinkcode they assigned you when you registered with the snort.org website. This tells PulledPork where to download the rules from. However, since the md5 for our snort version hasn't been uploaded at the time of writing this we are going to use an MD5 for a previous version 31210

```
18 # i.e. url|tarball|123456789,
19 rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot-31210.tar.gz|
   83e1c6e7392f59e67ea2ff1b7aab255ea17ca8a8
```

line 21: Comment out the community rules. These are not needed since they are included in the registered ruleset we included above:

```
20 # NEW Community ruleset:
21 #rule_url=https://snort.org/downloads/community/|community-rules.tar.gz|
   Community
```

line 72: We need to point to the correct snort.rules file, where PulledPork will save all the rules it downloads and includes from the local.rules file:

```
72 rule_path=/usr/local/etc/rules/snort.rules
73
```

line 87: We need to tell PulledPork where the local.rules file is to copy rules from (and into our snort.rules):

```
87 local_rules=/usr/local/etc/rules/local.rules
88
```

line 94: This tells PulledPork to output metadata about the rules in the newer sid_msg format:

```
94 sid_msg_version=2
95
```

line 134: change the distro to Ubuntu-18-4 (even if you're running Ubuntu 22):

```
134 distro=Ubuntu-18-4
135
```

line 142: tell PulledPork where to save the blocklist (IP Addresses that are known to be malicious and should be blocked):

```
142 block_list=/usr/local/etc/lists/default.blocklist
143
```

line 151: Tell PulledPork the default location for block and allow lists:

```
151 IPRVersion=/usr/local/etc/lists
152
```

line 209: uncomment this line to enable all rules in the downloaded rule file. The rules are split into different rulesets, depending on how aggressive you want to detect traffic. as the "security" ruleset is more aggressive about detecting traffic that might be malicious, or might be normal, we are going for "security":

```
209   ips_policy=security
210
```

Run PulledPork, passing it our configuration file and do extra logging. This will download the latest rulesets, combine them with any rules in our local.rules file and save all the rules into snort.rules, as well as save blacklist entries in our default.blocklist file:

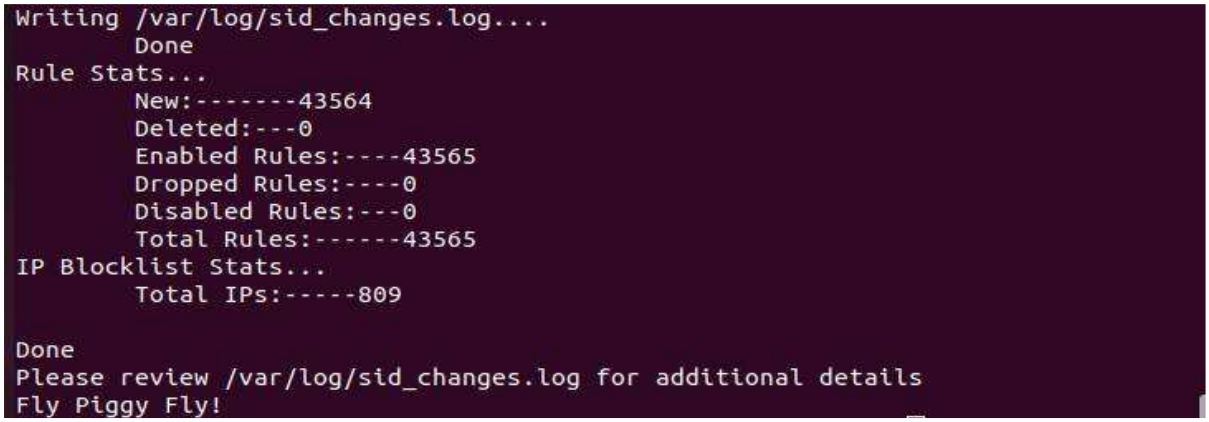
sudo /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf -l -P -E -T

we are using the following flags:

Flag	description
-c /usr/local/etc/pulledpork/pulledpork.conf	The PulledPork configuration file
-l	log important info to syslog
-P	Process rules even if no new rules downloaded
-E	only write enabled rules out
-T	Do not use .so rules (they don't work with original PuledPork)

Table 3.6: pulledpork command flags

you should see output similar to the following:



If this works, the next step is to turn this command into a scheduled task, so that we can update our rulesets daily: **sudo crontab -e**

The Snort team recommends that you randomize when PulledPork connects to their server to help with load balancing. In the example below, we have PulledPork checking at 13:44 every day.

44 13 * * * /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf -l -P -E -T

Modify snort.lua to load the snort.rules rather than the local.rules file (the rules in our local.rules are automatically added to the snort.rules file automatically by PulledPork along with all the downloaded rules, under section 5 configure detection:

```
ips =
{
    -- use this to enable decoder and inspector alerts
  enable_builtin_rules = true,

  include = RULE_PATH .. "/snort.rules",

   variables = default_variables
}
```

Test Snort to see if those rules load correctly:

```
-------------------------------------------------
rule counts
       total rules loaded: 44168
              text rules: 43565
            builtin rules: 603
            option chains: 44168
            chain headers: 1666
                 flowbits: 700
     flowbits not checked: 67
-------------------------------------------------
```

5.4. Configuring Snort plugins

We are going to activate some features in our snort.lua file, so we start with the command:

sudo gedit /usr/local/etc/snort/snort.lua

first we configure our home net variable:

HOME_NET = '192.168.1.0/24'

Enable the reputation blocklist. the Snort.lua file uses the older, unsupported “blacklist”, which has been replaced with blocklist:

Reputation = {blocklist = BLACK_LIST_PATH .. "/default.blocklist",}

we validate the file:

snort -c /usr/local/etc/snort/snort.lua

5.4.1. JSON Alerts Output Plugin

the alert_json output plugin is used to ease the process of importing Snort 3 log files to the SIEM of your choice. To enable alert_json output, we modify Snort.lua file in section 7 Configure Output:

sudo gedit /usr/local/etc/snort/snort.lua

then we add the following:

alert_json = {

file = true,

limit = 100,

**fields = 'seconds action class b64_data dir dst_addr dst_ap dst_port eth_dst eth_len **

**eth_src eth_type gid icmp_code icmp_id icmp_seq icmp_type iface ip_id ip_len msg mpls **

**pkt_gen pkt_len pkt_num priority proto rev rule service sid src_addr src_ap src_port **

target tcp_ack tcp_flags tcp_len tcp_seq tcp_win tos ttl udp_len vlan timestamp',}

in the alert_json plugin we specify 3 options

- File: enable outputting alerts the json file
- Limit: the limit in which snort changes to a new file
- Fields: identify which specific fields from the alert should be included in the json file

Now to test our plugin, the alerts will be saved to var/log/snort. Run the below commands and ping the host again:

sudo /usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \ -k none -l /var/log/snort -i enp0s3-m 0x1b

we have used two new flags:

-l var/log/snort: the directory where log files should be written

-m 0x1b: Umask of 033 for file permissions (rw-r–r–)

you won't see anything output to the screen after snort starts, since we've enabled the alert_json output module. Use ctrl-c to stop snort then check /var/log/snort:

5.4.2. Snort start-up script

We create a systemD script to run snort automatically on start-up. We will also have snort run as a regular (non-root) user after start-up for security reasons.

Create the snort user and group:

sudo groupadd snort

sudo useradd snort -r -s /sbin/nologin -c SNORT_IDS -g snort

Remove old log files:

sudo rm /var/log/snort/*

Grant the 'snort' user rights to the log directory:

sudo chmod -R 5775 /var/log/snort

sudo chown -R snort:snort /var/log/snort

create the systemD service file:

sudo gedit /lib/systemd/system/snort3.service

enter the following information:

[Unit]

Description=Snort3 NIDS Daemon After=syslog.target network.target

[Service]

Type=simple

ExecStart=/usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s 65535 \ -k none -l /var/log/snort -D -u snort -g snort -i enp0s3 -m 0x1b --create-pidfile

[Install]

WantedBy=multi-user.target

The following flags has been used in our SystemD file:

Flag	Description
/usr/local/bin/snort	This is the path to the snort binary. We don't use sudo here since the script will be started with elevated (root) privileges.
-c /usr/local/etc/snort/snort.lua	The snort.lua configuration file.
-s 65535	Set the snaplen so Snort doesn't truncate and drop oversized packets.
-k none	Ignore bad checksums, otherwise snort will drop packets with bad checksums, and they won't be evaluated.
-l /var/log/snort	The path to the folder where Snort will store all the log files it outputs.
-D	Run snort as a Daemon.
-u snort	After startup (and after doing anything that requires elevated privileges), switch to run as the "snort" user.
-g snort	After startup, run as the "snort" group.
--create-pidfile	Create a PID file in the log directory (so pulledpork can restart snort after loading new rules)

Table 3.7: systemd flags

Enable and launch the Snort systemD service:

sudo systemctl enable snort3

sudo service snort3 start

check the status of the service:

service snort3 status

your output should be similar to the following, showing 'active (running)':

```
badkarma@badkarma-VirtualBox:~/snort_src$ sudo service snort3 start
[sudo] password for badkarma:
badkarma@badkarma-VirtualBox:~/snort_src$ service snort3 status
● snort3.service - Snort3 NIDS Daemon
     Loaded: loaded (/lib/systemd/system/snort3.service; enabled; vendor prese>
     Active: active (running) since Wed 2022-06-15 02:32:32 CET; 2s ago
   Main PID: 20483 (snort)
      Tasks: 1 (limit: 5895)
     Memory: 56.4M
        CPU: 2.016s
     CGroup: /system.slice/snort3.service
             └─20483 /usr/local/bin/snort -c /usr/local/etc/snort/snort.lua -s>
```

you can check the full output of the service with the following command if there are any problems:

sudo journalctl -u snort3.service

5.5. Splunk

Splunk is our SIEM of choice. He will graphically display all of snort alerts, and will give us tools that will allow us to properly search and understand as well as draw some deeper information related to those alerts.

5.5.1. Installing splunk

In order to download the software, you are going to have to sign up on their website first. Go to splunk.com then hit free trial on the top right corner and fill in the required fields.

After signing up and logging in, navigate to the download page, where you will click the link under “Splunk Enterprise” titled Download Free 60-day trial.

Select .deb file since ubuntu is a Debian based. You can cancel this download and click Download via command line **wget** if you want to download the installer instead.

Once the download is completed run the following commands to install it to your system:

sudo dpkg -i splunk-9.*.deb

sudo chown -R splunk:splunk /opt/splunk

splunk will be installed to opt/splunk.

Now we run splunk, and for the first time you will be asked to choose an admin user and a password for splunk. The user and password will later be used to access the web interface:

sudo /opt/splunk/bin/splunk start --answer-yes --accept-license

now we are going to configure splunk to start at boot time, and we will also enable system and start the service for splunk:

sudo /opt/splunk/bin/splunk stop

sudo /opt/splunk/bin/splunk enable boot-start -systemd-managed 1

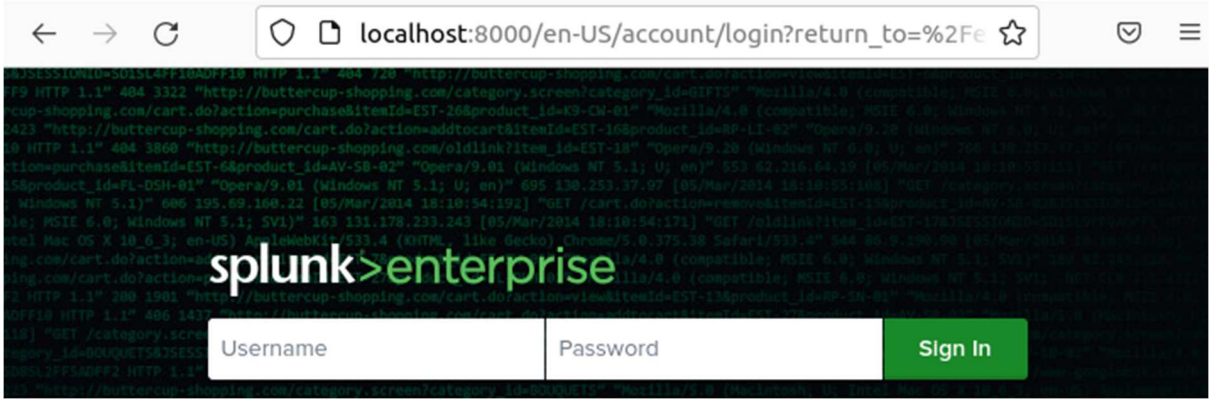
sudo chown -R splunk:splunk /opt/splunk

sudo service Splunkd start

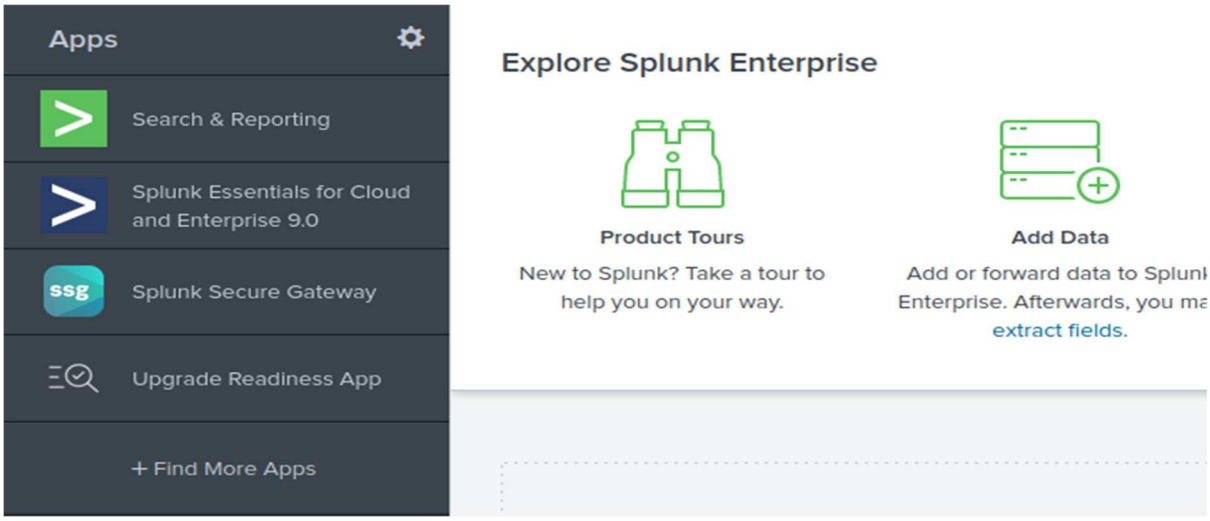
The Splunk server is now listening on port 8000 of this server (http://localhost:8000).

5.5.2. Configuring Splunk:

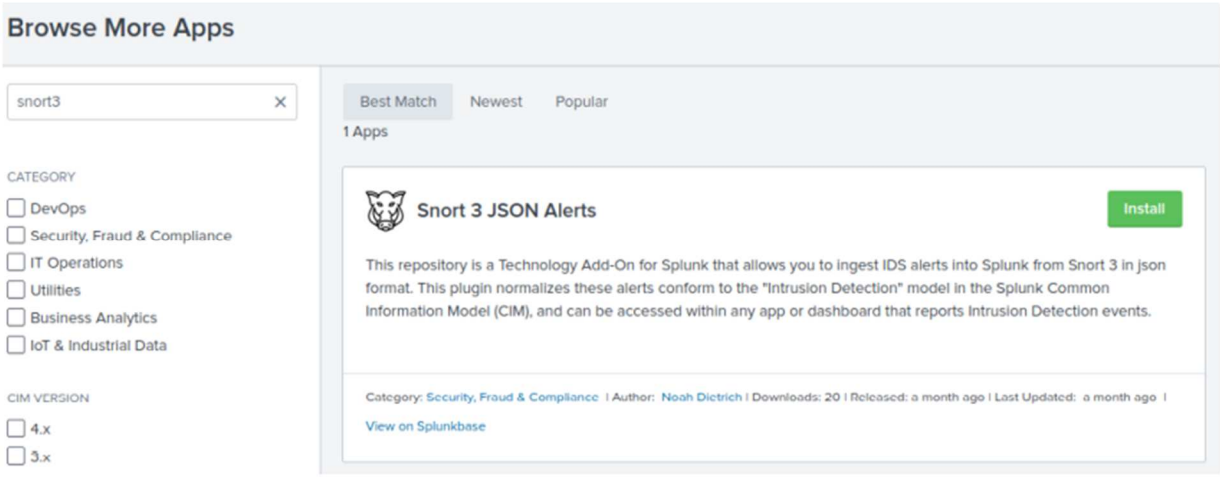
Using the username and the password log onto your instance.



We are going to install an add-on that will help us simplify and collect logs created by snort 3 easily. To install this app, from the main web page of your Splunk instance, click the link titled **+Find More Apps** on the left side of the Splunk Web Interface:

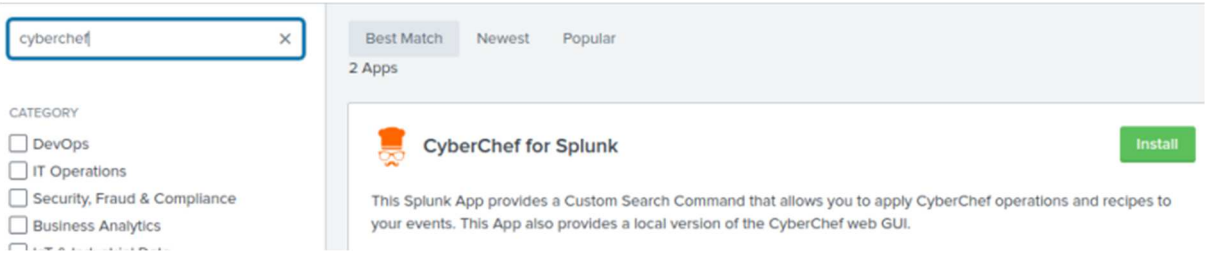


This will take you to Splunkbase. Search Splunkbase for Snort and you'll be presented with multiple results choose: Snort 3 JSON Alerts. Click the install button, next to this Add-on:



enter the username and password your created with Splunk when you registered to download Splunk. Accept the terms & conditions, and click Login and Install. click done once the install is completed.

Next, we are going to install the CyberChef for Splunk plugin that will allow us to covert the b64_data fields into readable text:



Next, we configure the Snort 3 JSON Alerts add-on to tell Splunk where the log files are stored that Snort 3 generated so Splunk can ingest them. We use the following commands:

sudo mkdir /opt/splunk/etc/apps/TA_Snort3_json/local

sudo touch /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf

sudo gedit /opt/splunk/etc/apps/TA_Snort3_json/local/inputs.conf

then we input the following text into the inputs.conf file:

[monitor:///var/log/snort/*alert_json.txt*]

sourcetype = snort3:alert:json

finally, we restart Splunk:

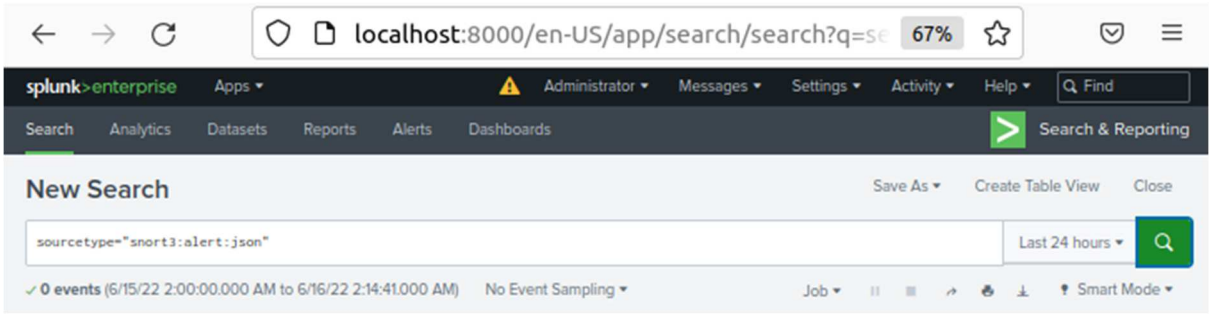
sudo service Splunkd restart

now when Splunk starts, it will scan the **/var/log/snort** directory for json files, assign them sourcetype of **snort3:alert:json**, and ingest them so we can search them.

After the restart you login again and go to **Search and Reporting** on the left side and type in the search field:

sourcetype="snort3:alert:json"

then hit the search button, this will show all events that the server is collecting.



5.5.3. Using splunk

show all events in a table with the time, source, destination, and message, run the following search:

sourcetype="snort3:alert:json"

| table _time src_ap dst_ap msg

To show the count of all events by destination:

sourcetype="snort3:alert:json"

| stats count by dest

to show all events sources on a map (you may have to click on the “Visualization” tab, and then “line chart” and change it to Choropleth Map):

sourcetype="snort3:alert:json"

| iplocation src_addr

| stats count by Country

| geom geo_countries featureIdField="Country"

For many of your events, there will payload data (the b64_data) field that's base64 encoded (http and SMTP are a good example of this). To convert this data so we can read it, we use the "cyberchef" function to convert it for each event (on the fly), and add a new field to each event called "decrypted":

sourcetype="snort3:alert:json" dest_port=80

| cyberchef infield='b64_data' outfield=decrypted operation="FromBase64"

| table src_addr, dst_addr, rule, msg, decrypted

5.6. Installing OpenAppId

OpenAppId is a dynamic preprocessor created by cisco.[41] it is a tool that allows the identification of application layer. Openappid is used to create rules that operate on application-layer traffic, and to log traffic statistics for each type of traffic detected.

First go to https://snort.org/downloads#openappid, look for ***snort-openappid.tar.gz*** and download it, then run the following commands to finish the installation:

tar -xzvf snort-openapid.tar.gz

sudo cp -R odp /usr/local/lib/

Next, we need to edit our Snort configuration file to point to this odp directory:

sudo gedit /usr/local/etc/snort/snort.lua

Locate the following sections and configure as follows in the Configure Inspection section (around line 90 or so in your snort.lua:

appid = { app_detector_dir = '/usr/local/lib', }

Validate your snort.lua file as above since you've made changes.

snort -c /usr/local/etc/snort/snort.lua --warn-all

next we modify local.rules and create some rules to test if OpenAppId is functioning correctly:

alert tcp any any -> any any (msg:"Facebook Detected"; appids:"Facebook"; sid:10000002; metadata:policy security-ips alert;)

since we are using PulledPork, we need to run it first to re-build pulledpork.rules file and include the new rules

sudo /usr/local/bin/pulledpork.pl -c /usr/local/etc/pulledpork/pulledpork.conf -l -P -E -T

generate some Facebook traffic using **wget facebook.com**, and you'll see the alerts written to splunk and to the json file:

sourcetype="snort3:alert:json" msg="Facebook Detected"

Snort is ready to go.

6. Testing Snort

for the testing phase, we made some preparations. We acquired files.pcap (.cap and .pcapng), they are packet captures that contain malicious as well as normal packets. We also setup kali linux on a virtual machine to generate some live attacks. For this testing phase Snort is running in IDS mode using the rules provided by TALOS that were automatically download by pulledpork and displays its findings on the terminal.

6.1. General test

First, we will test snort with a variety of attacks by using pcaps files provided by MACCDC for National CyberWatch Mid-Atlantic Collegiate Cyber Defense Competition.

Run the command: snort -c $my_path/etc/snort/snort.lua --pcap-dir /path/to/pcap/dir \ --pcap-filter '*.pcap' --max-packet-threads 8 -A alert_fast

The process of generating alerts:

```
03/17-19:23:58.560000 [**] [116:434:1] "(icmp4) ICMP ping Nmap" [**] [Priority:
 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.27.105
03/17-19:23:58.630000 [**] [116:434:1] "(icmp4) ICMP ping Nmap" [**] [Priority:
 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.27.106
03/17-16:15:55.579999 [**] [122:5:1] "(port_scan) TCP filtered portscan" [**] [
Priority: 3] {TCP} 192.168.202.140:42475 -> 192.168.26.25:1027
03/17-16:15:55.970000 [**] [122:5:1] "(port_scan) TCP filtered portscan" [**] [
Priority: 3] {TCP} 192.168.202.140:42475 -> 192.168.27.25:1433
03/17-16:15:56.710000 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
03/17-19:24:00.940000 [**] [129:2:1] "(stream_tcp) data on SYN packet" [**] [Pr
iority: 3] {TCP} 192.168.202.138:63000 -> 192.168.27.103:60000
03/17-19:24:00.960000 [**] [129:2:1] "(stream_tcp) data on SYN packet" [**] [Pr
iority: 3] {TCP} 192.168.202.138:63000 -> 192.168.27.101:60000
```

```
03/17-19:16:35.190000 [**] [116:434:1] "(icmp4) ICMP ping Nmap" [**] [Priority:
 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.21.220
03/17-19:16:35.550000 [**] [116:418:1] "(icmp4) ICMP4 type other" [**] [Priorit
y: 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.21.1
03/17-19:16:35.580000 [**] [119:8:1] "(http_inspect) URI path contains consecut
ive slash characters" [**] [Priority: 3] [AppID: Internet Explorer] {TCP} 192.1
68.202.138:54487 -> 192.168.21.102:80
03/17-19:16:36.650000 [**] [122:26:1] "(port_scan) ICMP filtered sweep" [**] [P
riority: 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.21.1
03/17-19:16:36.950000 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
03/17-19:16:39.270000 [**] [122:17:1] "(port_scan) UDP portscan" [**] [Priority
: 3] [AppID: ICMP] {ICMP} 192.168.21.102 -> 192.168.202.138
03/17-19:16:39.320000 [**] [116:434:1] "(icmp4) ICMP ping Nmap" [**] [Priority:
 3] [AppID: ICMP] {ICMP} 192.168.202.138 -> 192.168.21.221
03/17-19:16:39.580000 [**] [122:1:1] "(port_scan) TCP portscan" [**] [Priority:
 3] {TCP} 192.168.21.103:9618 -> 192.168.202.138:60181
```

```
03/17-19:19:19.290000 [**] [116:150:1] "(decode) loopback IP" [**] [Priority: 3
] {UDP} 127.0.0.1:1000 -> 192.168.21.253:111
03/17-19:19:19.430000 [**] [116:150:1] "(decode) loopback IP" [**] [Priority: 3
] {UDP} 127.0.0.1:1000 -> 192.168.21.253:111
03/17-19:19:22.110000 [**] [119:228:1] "(http_inspect) server response before c
lient request" [**] [Priority: 3] {TCP} 192.168.21.203:80 -> 192.168.202.138:50
754
03/17-19:19:22.110000 [**] [119:224:1] "(http_inspect) misformatted HTTP traffi
c" [**] [Priority: 3] {TCP} 192.168.202.138:50754 -> 192.168.21.203:80
03/17-19:19:22.170000 [**] [119:31:1] "(http_inspect) HTTP request method is no
t known to Snort" [**] [Priority: 3] [AppID: Internet Explorer] {TCP} 192.168.2
02.138:50770 -> 192.168.21.203:80
03/17-19:19:22.220000 [**] [119:228:1] "(http_inspect) server response before c
lient request" [**] [Priority: 3] {TCP} 192.168.21.203:80 -> 192.168.202.138:50
777
03/17-19:19:22.220000 [**] [119:201:1] "(http_inspect) not HTTP traffic or unre
coverable HTTP protocol error" [**] [Priority: 3] {TCP} 192.168.202.138:50777 -
> 192.168.21.203:80
03/17-19:19:24.510000 [**] [116:446:1] "(tcp) TCP port 0 traffic" [**] [Priorit
y: 3] {TCP} 192.168.202.138:51295 -> 192.168.21.25:0
```

The detection and codec output are as follows:

```
-------------------------------------------------
detection
                analyzed: 14374046
              hard_evals: 13894825
            raw_searches: 1058
         cooked_searches: 1163
            pkt_searches: 2221
                  alerts: 113735
            total_alerts: 113735
                  logged: 113735
-------------------------------------------------
```

```
-------------------------------------------------
codec
                   total: 14374046      (100.000%)
                   other: 125445        (  0.873%)
                discards: 6             (  0.000%)
                     arp: 32929         (  0.229%)
                     eth: 14374046      (100.000%)
                     gtp: 3             (  0.000%)
                   icmp4: 207274        (  1.442%)
                icmp4_ip: 34749         (  0.242%)
                   icmp6: 5902          (  0.041%)
                icmp6_ip: 41            (  0.000%)
                    igmp: 753           (  0.005%)
                    ipv4: 13147574      ( 91.467%)
                    ipv6: 898255        (  6.249%)
           ipv6_hop_opts: 688           (  0.005%)
                     llc: 287739        (  2.002%)
                     tcp: 13533091      ( 94.149%)
                  teredo: 74            (  0.001%)
                     udp: 180986        (  1.259%)
                    vlan: 14352660      ( 99.851%)
                   vxlan: 2             (  0.000%)
-------------------------------------------------
```

Snort generated 113735 alerts from 14347046 analyzed packets.

6.2. Targeted tests

6.2.1. Brute-Force

For the brute-force test, we have set up an ftp server on our ubuntu machine with the username: ftpuser and password: kabaneri. We attack with hydra a brute-force tool in kali linux.

The command used for the attack is: hydra -L file/dir/users.txt -P file/dir/passwords.txt ftp://192.168.1.104 -V -t 4

```
┌──(kali㉿kali)-[~]
└─$ hydra -L /home/kali/Documents/users.txt -P /home/kali/Documents/passwords.txt ftp:/
/192.168.1.104 -V -t 1
Hydra v9.3 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military
or secret service organizations, or for illegal purposes (this is non-binding, these **
* ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2022-06-24 07:57:19
[DATA] max 1 task per 1 server, overall 1 task, 404 login tries (l:4/p:101), ~404 tries
 per task
[DATA] attacking ftp://192.168.1.104:21/
[ATTEMPT] target 192.168.1.104 - login "admin" - pass "123456" - 1 of 404 [child 0] (0/
0)
[ATTEMPT] target 192.168.1.104 - login "admin" - pass "12345" - 2 of 404 [child 0] (0/0
)
[ATTEMPT] target 192.168.1.104 - login "admin" - pass "123456789" - 3 of 404 [child 0]
(0/0)
```

```
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "yellow" - 197 of 404 [child 1]
 (0/0)
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "daniela" - 198 of 404 [child 2
] (0/0)
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "lauren" - 199 of 404 [child 0]
 (0/0)
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "mickey" - 200 of 404 [child 3]
 (0/0)
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "princesa" - 201 of 404 [child
1] (0/0)
[ATTEMPT] target 192.168.1.104 - login "ftpuser" - pass "kabaneri" - 202 of 404 [child
2] (0/0)
[21][ftp] host: 192.168.1.104   login: ftpuser   password: kabaneri
[ATTEMPT] target 192.168.1.104 - login "kastro" - pass "123456" - 203 of 404 [child 2]
(0/0)
[ATTEMPT] target 192.168.1.104 - login "kastro" - pass "12345" - 204 of 404 [child 0] (
0/0)
```

```
[ATTEMPT] target 192.168.1.104 - login "kabay" - pass "lauren" - 401 of 404 [child 2] (
0/0)
[ATTEMPT] target 192.168.1.104 - login "kabay" - pass "mickey" - 402 of 404 [child 1] (
0/0)
[ATTEMPT] target 192.168.1.104 - login "kabay" - pass "princesa" - 403 of 404 [child 2]
 (0/0)
[ATTEMPT] target 192.168.1.104 - login "kabay" - pass "kabaneri" - 404 of 404 [child 0]
 (0/0)
1 of 1 target successfully completed, 1 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2022-06-23 23:59:31
```

Meanwhile, snort was active on the network interface and running in IDS mode. The command used for snort since it's a live test is: snort -c /usr/local/etc/snort/snort.lua -i enp0s3 -A alert_full

The process of generating alerts:

```
[**] [1:10000004:1] "FTP connection attempt" [**]
[Priority: 0]
06/24-06:01:50.014807 192.168.1.107:60103 -> 192.168.1.104:21
TCP TTL:64 TOS:0x0 ID:64283 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xF9EE0  Ack: 0x54FA9230  Win: 0x804  TcpLen: 20

[**] [1:10000004:1] "FTP connection attempt" [**]
[Priority: 0]
06/24-06:01:50.076275 192.168.1.107:60103 -> 192.168.1.104:21
TCP TTL:64 TOS:0x0 ID:64284 IpLen:20 DgmLen:53 DF
***AP*** Seq: 0xF9EE0  Ack: 0x54FA9230  Win: 0x804  TcpLen: 20

[**] [1:10000004:1] "FTP connection attempt" [**]
[Priority: 0]
06/24-06:01:53.381575 192.168.1.107:60103 -> 192.168.1.104:21
TCP TTL:64 TOS:0x0 ID:64285 IpLen:20 DgmLen:40 DF
***A**** Seq: 0xF9EED  Ack: 0x54FA9246  Win: 0x804  TcpLen: 20
```

The detection and codec output are as follows:

```
-------------------------------------------------
detection
                analyzed: 6495
              hard_evals: 2299
                  alerts: 2335
            total_alerts: 2335
                  logged: 2335
             alert_limit: 1
-------------------------------------------------
```

```
-------------------------------------------------
codec
                   total: 6495         (100.000%)
                 discards: 2360         ( 36.336%)
                     arp: 123          (  1.894%)
                     eth: 6495         (100.000%)
                   icmp4: 7            (  0.108%)
                 icmp4_ip: 7           (  0.108%)
                   icmp6: 156          (  2.402%)
                    igmp: 25           (  0.385%)
                    ipv4: 6141         ( 94.550%)
                    ipv6: 231          (  3.557%)
           ipv6_hop_opts: 6            (  0.092%)
                     tcp: 5805         ( 89.376%)
                     udp: 379          (  5.835%)
-------------------------------------------------
```

6.2.2. Denial of service (Dos)

For DOS tests we used 18 different .pcap file.[42]

The process of generating alerts:

```
03/27-14:49:51.880357 [**] [116:447:1] "(udp) UDP port 0 traffic" [**] [Priorit
y: 3] [AppID: MDNS] {UDP} 16.228.107.44:0 -> 10.10.10.10:5353
03/27-14:49:52.110657 [**] [116:182:1] "(geneve) invalid header" [**] [Priority
: 3] {UDP} 82.179.14.240:6081 -> 10.10.10.10:623
03/27-14:50:11.402306 [**] [116:272:1] "(ipv6) IPv6 truncated extension header"
 [**] [Priority: 3] {UDP} 197.35.31.222:6081 -> 10.10.10.10:427
03/27-14:50:16.747922 [**] [116:447:1] "(udp) UDP port 0 traffic" [**] [Priorit
y: 3] {UDP} 164.111.12.115:0 -> 10.10.10.10:138
```

```
01/06-11:10:34.205348 [**] [123:8:1] "(stream_ip) fragmentation overlap" [**] [
Priority: 3] {IP} 203.45.216.97 -> 10.10.10.10
01/06-11:10:34.205444 [**] [116:475:1] "(ipv6) IPv6 mobility header includes an
 invalid value for the 'payload protocol' field" [**] [Priority: 3] {IP} 23.35.
53.240 -> 10.10.10.10
01/06-11:10:34.205444 [**] [116:296:1] "(ipv6) IPv6 packet includes out-of-orde
r extension headers" [**] [Priority: 3] {IP} 23.35.53.240 -> 10.10.10.10
01/06-11:10:34.205444 [**] [116:272:1] "(ipv6) IPv6 truncated extension header"
 [**] [Priority: 3] {IP} 23.35.53.240 -> 10.10.10.10
01/06-11:10:34.205486 [**] [116:170:1] "(mpls) bad MPLS frame" [**] [Priority:
3] [AppID: MPLS] {IP} 139.208.9.170 -> 10.10.10.10
01/06-11:10:34.205492 [**] [116:450:1] "(decode) bad IP protocol" [**] [Priorit
y: 3] [AppID: Swipe] {IP} 200.5.241.70 -> 10.10.10.10
01/06-11:10:34.205560 [**] [116:418:1] "(icmp4) ICMP4 type other" [**] [Priorit
y: 3] [AppID: ICMP] {ICMP} 5.154.243.206 -> 10.10.10.10
```

The detection and codec output are as follows:

```
--------------------------------------------------
detection
                analyzed: 774673
              hard_evals: 549
                  alerts: 14422
            total_alerts: 14423
                  logged: 14423
--------------------------------------------------
```

```
-------------------------------------------------
codec
                  total: 779387       (100.000%)
                  other: 67135        (  8.614%)
               discards: 9600         (  1.232%)
                    arp: 13           (  0.002%)
                   auth: 470          (  0.060%)
              bad_proto: 1063         (  0.136%)
                    esp: 589          (  0.076%)
                    eth: 779387       (100.000%)
                 geneve: 6            (  0.001%)
                    gre: 1            (  0.000%)
                    gtp: 7            (  0.001%)
                  icmp4: 11911        (  1.528%)
               icmp4_ip: 1467         (  0.188%)
                  icmp6: 554          (  0.071%)
                   ipv4: 779372       ( 99.998%)
                   ipv6: 27           (  0.003%)
           ipv6_dst_opts: 480         (  0.062%)
               ipv6_frag: 521         (  0.067%)
           ipv6_hop_opts: 8154        (  1.046%)
           ipv6_mobility: 571         (  0.073%)
            ipv6_no_next: 525         (  0.067%)
            ipv6_routing: 482         (  0.062%)
                     llc: 2           (  0.000%)
                    mpls: 538         (  0.069%)
                     pgm: 531         (  0.068%)
                     tcp: 59387       (  7.620%)
                  teredo: 10          (  0.001%)
            trans_bridge: 236         (  0.030%)
                     udp: 601383      ( 77.161%)
                   vxlan: 10          (  0.001%)
```

6.2.3. Malware

For the malware test we used 4 .pcap files in addition to teardrop.cap :

Matanbuchus-with-Cobalt-Strike.pcap

IcedID-infection-with-DarkVNC.pcap

Contact-Forms-BazarLoader-with-Cobalt-Strike.pcap

obama186-Qakbot-infection-with-DarkVNC-and-spambot-traffic. pcap

The process of generating alerts:

```
06/17-21:02:15.711196 [**] [122:3:1] "(port_scan) TCP portsweep" [**] [Priority
: 3] {TCP} 10.6.17.101:51045 -> 23.82.141.136:443
06/17-21:04:03.461453 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
06/17-21:06:03.449626 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
06/17-21:07:08.460830 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
06/17-21:07:08.466541 [**] [112:1:1] "(arp_spoof) unicast ARP request" [**] [Pr
iority: 3] {ARP}  ->
06/17-21:07:56.494038 [**] [116:444:1] "(ipv4) IPv4 option set" [**] [Priority:
 3] [AppID: IGMP] {IP} 10.6.17.101 -> 224.0.0.22
06/17-21:07:56.499028 [**] [116:444:1] "(ipv4) IPv4 option set" [**] [Priority:
 3] [AppID: IGMP] {IP} 10.6.17.101 -> 224.0.0.22
06/17-21:07:56.499549 [**] [116:444:1] "(ipv4) IPv4 option set" [**] [Priority:
 3] [AppID: IGMP] {IP} 10.6.17.101 -> 224.0.0.22
06/17-21:07:56.499670 [**] [116:444:1] "(ipv4) IPv4 option set" [**] [Priority:
 3] [AppID: IGMP] {IP} 10.6.17.101 -> 224.0.0.22
06/17-21:07:56.938241 [**] [116:444:1] "(ipv4) IPv4 option set" [**] [Priority:
```

The detection and codec output are as follows:

```
-------------------------------------------------
detection
                analyzed: 273521
            hard_evals: 217991
          raw_searches: 81690
       cooked_searches: 152432
          pkt_searches: 234122
          pdu_searches: 3650
        file_searches: 789
                alerts: 2197
          total_alerts: 2197
                logged: 2197
           alert_limit: 27
-------------------------------------------------
```

```
-------------------------------------------------
codec
                  total: 273521       (100.000%)
                  other: 5            (  0.002%)
                    arp: 5453         (  1.994%)
                    eth: 273521       (100.000%)
                  icmp4: 12           (  0.004%)
                   igmp: 60           (  0.022%)
                   ipv4: 268062       ( 98.004%)
                    llc: 1            (  0.000%)
                    tcp: 263510       ( 96.340%)
                    udp: 4478         (  1.637%)
-------------------------------------------------
```

6.2.4. Packet Injection Attack/ Man-in-the-middle

To test packet injection attacks, we have used 13 .pcap files that contain malicious packets recorded during real PIA attacks:

handshake_lost_hijack_netcat_loopback1.pcap

id1-cn_packet-injection.pcap

linkedin_FIN.pcap

man-in-the-middle-egypt-packetlogic-ttl-localization.pcap and man-in-the-middle-turkey-malware-injection.pcap

mots-with-fin.pcap and mots-with-fin_no_3whs.pcap

ordered_coalesce_netcat1.pcap and ordered_coalesce_netcat2.pcap

PIA.pcap (Packet injection against www.02995.com doing a redirect to www.hao123.com)

qi_local_GET_slashdot_redirect.pcap and qi_local_SYNACK_linkedin_redirect.pcap

sloppy_spray_injection1.pcap

The process of generating alerts:

```
03/01-09:03:49.208326 [**] [119:13:1] "(http_inspect) HTTP start line or header
 line terminated by LF without a CR" [**] [Priority: 3] [AppID: Firefox] {TCP}
192.168.1.254:59360 -> 122.225.98.197:80
03/01-09:03:50.739822 [**] [116:6:1] "(ipv4) IPv4 datagram length > captured le
ngth" [**] [Priority: 3] {unknown}  ->
03/01-09:03:50.743326 [**] [119:201:1] "(http_inspect) not HTTP traffic or unre
coverable HTTP protocol error" [**] [Priority: 3] [AppID: Hao123.com] {TCP} 192
.168.1.254:59362 -> 103.235.46.234:80
03/01-09:03:50.743374 [**] [116:6:1] "(ipv4) IPv4 datagram length > captured le
ngth" [**] [Priority: 3] {unknown}  ->
03/01-09:03:50.750962 [**] [116:6:1] "(ipv4) IPv4 datagram length > captured le
ngth" [**] [Priority: 3] {unknown}  ->
```

The detection and codec output are as follows:

```
-------------------------------------------------
detection
                analyzed: 495
              hard_evals: 445
            raw_searches: 395
         cooked_searches: 99
            pkt_searches: 494
            pdu_searches: 138
           file_searches: 14
                  alerts: 33
            total_alerts: 33
                  logged: 33
-------------------------------------------------
```

```
-------------------------------------------------
codec
                  total: 495          (100.000%)
                discards: 18          (  3.636%)
                     eth: 138         ( 27.879%)
                    ipv4: 495         (100.000%)
                     raw: 357         ( 72.121%)
                     tcp: 477         ( 96.364%)
-------------------------------------------------
```

7. Conclusion

Snort has been successfully installed and appropriately configured. The tests prove that Snort is fully operational. At the end of this chapter, we have a functional intrusion detection system that is configured to run when the system boot. The results of the tests were tremendous, where in the general test, Snort generated 113735 alerts. In the targeted tests Snort never failed to generate alerts with each type of attack.

General conclusion

Today, computer systems will continue to grow in importance and become more and more integrated in our daily life. The evolution of these systems involves numerous technological challenges such as mobility, scalability and autonomy, and the responsiveness, etc. Among these challenges, the fundamental issue of security remains an unsolved because of the increasing dematerialization of information and the risks related to the complexity of these systems.

This thesis deals with the security of computer systems. The contribution that we have carried out focuses mainly on the protection of these systems by the detection of computer attacks, which today provokes growing interest as much in the academic, professional and above all the industrial world because of the dissemination of these computer threats (intrusions), which are currently increasingly more aggressive. The implementation of a good security strategy makes intrusions involves improving IDS detection techniques. For this purpose, it is necessary to improve detection techniques, to process data efficiently and reliably.

We began our contribution by proposing the installation of an intrusion detection system into the information system to increase its security. Then we set out describing and simplifying the process of implementing the IDS of our choice Snort. For the evaluation phase and to concrete our work, Snort was tested with an extensive set of malicious packets in which it presented tremendous detection capabilities.

Reference list:

[1] Mason, A. M. (2004). *Cisco secure virtual private networks* (2nd ed.). cisco presse.

[2] Cisco Secure Intrusion Detection System (CSIDS) v4.1. Cisco Systems, Inc.2004.

[3] Overview of How Cyber Resiliency Affects the Cyber Attack Lifecycle. the Mitre Corporation. 2015.

[4] Y, Diogenes. E, Ozkaya. Cybersecurity - Attack and Defense Strategies. Packt.

[5] Kutub, T. Al Sakib Khan, P., 2020. Cybersecurity Fundamentals A Real-World Perspective. CRC Presse.

[6] Ray Hunt. 1998. Internet/Intranet firewall security-policy, architecture and transaction services. Computer Communications 21, 13 (1998), 1107–1123

[7] Huashan Chen, Jin-Hee Cho, and Shouhuai Xu. 2018. Quantifying the security effectiveness of firewalls and DMZs. In Proceedings of the 5th Annual Symposium and Bootcamp on Hot Topics in the Science of Security (HoTSoS '18). Association for Computing Machinery, New York, NY, USA, Article 9, 1–11

[8] Scarfone, K. A., & Hoffman, P. (2009). SP 800-41 Rev. 1. Guidelines on Firewalls and Firewall Policy. Gaithersburg, MD, USA: National Institute of Standards & Technology.

[9] HOWTO- Design and Configure a DMZ Network. (n.d.). Www.skullbox.net. Retrieved June 21, 2022, from http://www.skullbox.net/configureDMZnetwork.php

[10] Webb, J. (2014). Network Demilitarized Zone (DMZ). ICTN.

[11] Chhabra, Y. (2015). A Study of Recent Research Trends of Proxy Server. International Journal of Advanced Technology in Engineering and Science, 3(01), 159-164.

[12] Sysel, M., & Doležal, O. (2014). An Educational HTTP Proxy Server. Procedia Engineering, 69, 128–132.

[13] Tayal, S., Gupta, N., Gupta, P., Goyal, D., & Goyal, M. (2017). A review paper on network security and cryptography. Advances in Computational Sciences and Technology, 10(5), 763-770.

[14] G. Karatas¸, “Genetic algorithm for intrusion detection system,” in Signal Processing and Communication Application Conference (SIU), 2016 24th. IEEE, 2016, pp. 1341–1344.

[15] Bace, R. and Mell, P., 2001. *Intrusion detection systems*. [Gaithersburg, MD: U.S. Dept. of Commerce, Technology Administration, National Institute of Standards and Technology.

[16] Rajaallah, M., Chamkar, S. and El Hayat, S., 2019. *Intrusion Detection Systems: To an Optimal Hybrid Intrusion Detection System*. Springer, Cham.

[17] Kim, K., Aminanto, M. and Tanuwidjaja, H., n.d. *Network Intrusion Detection using Deep Learning*.

[18] Kent, K. and Mell, P., 2007. *Guide to intrusion detection and prevention systems (IDPS)*. Gaithersburg: National Institute of Standards and Technology.

[19] Burton, J., Dubrawsky, I., Osipov, V. and Baumrucker, C., 2003. *Cisco Security Professional's Guide to Secure Intrusion Detection Systems*. syngress.

[20] Anley, C., Heasman, J., Linder, F. and Richarte, G., 2007. *The Shellcoder's Handbook*. 2st ed. Indianapolis, Ind.: Wiley, pub., ch.3.

[21] Kennedy, D., O'Gorman, J., Kearns, D. and Aharoni, M., 2012. *Metasploit*. San Francisco: no starch press, pp.12-13.

[22] Horng, S.-J., Su, M.-Y., Chen, Y.-H., Kao, T.-W., Chen, R.-J., Lai, J.-L., & Perkasa, C. D. (2011). A novel intrusion detection system based on hierarchical clustering and support vector machines. *Expert Systems with Applications*, *38*(1), 306–313.

[23] Mukkamala, S., Janoski, G., & Sung, A. (2002, May). Intrusion detection using neural networks and support vector machines. In *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)* (Vol. 2, pp. 1702-1707).

[24] Ahmad, I., Basheri, M., Iqbal, M. J., & Rahim, A. (2018). Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection.

[25] Modi AS. "Review article on deep learning approaches," in 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS).

[26] Chen M, Challita U, Saad W, Yin C, Debbah M. "Artificial neural networks based machine learning for wireless networks: A tutorial,".

[27] Smith RE, Forrest S and Perelson AS (1993) Searching for diverse, cooperative population with genetic algorithms. Evolutionary Computation 1(2): 127–149

[28] L de Castro and J Timmis. Artificial Immune Systems: A New Computational Intelligence Approach. Springer, 2002.

[29] Kim, J., & Bentley, P. (1999, September). The human immune system and network intrusion detection. In *7th European Conference on Intelligent Techniques and Soft Computing (EUFIT'99), Aachen, Germany* (pp. 1244-1252).

[30] BazaraI.A. Barry and H. Anthony Chan, “,” in Peter Stravoulakis, Mark Stamp (Eds), Handbook of Information and Communication Security, 2010.

[31] IT Management Software and Observability Platform | SolarWinds. (n.d.). Solarwind. Retrieved June 13, 2022, from https://www.solarwinds.com/

[32] Bricata | Network Detection & Response | Visibility & Analytics | Threat Hunting. 2022. *Bro IDS*. [online] Available at: <https://bricata.com/blog/what-is-bro-ids/> [Accessed 3 June 2022].

[33] OSSEC. 2022. *OSSEC - World's Most Widely Used Host Intrusion Detection System - HIDS*. [online] Available at: <https://www.ossec.net/> [Accessed 3 June 2022].

[34] Snort. 2022. Snort IDS. [online] Available at: <https://www.snort.org/> [Accessed 3 June 2022].

[35] Suricata. 2022. Available at: <https://suricata.io//> [Accessed 3 June 2022].

[36] Security Onion. 2022. Available at: <https://securityonionsolutions.com/> [Accessed 3 June 2022].

[37] Snort User’s Manual 3. The Snort Project November 11, 2021.

[38] C. Scott, P. Wolfe, and B. Hayes, Snort for Dummies.

[39] SNORTOLOGY 101 THE ANATOMY OF A SNORT RULE, Talos.

[40] N. Dietrich, Snort 3.1.17.0 on Ubuntu 18 & 20 Configuring a Full NIDS & SIEM.

[41] Open-Source Detectors Developers Guide. Cisco.

[42] L.F. Haaijer, DDoS Packet Capture Collection, (2022).