INDIVIDUAL PROJECT

FINAL REPORT

Deploying And Examining Security Onion For Network Threat Detection In A Simulated Network Environment

Author
Silvia Aminul

Supervisor
Dr. Kamran Ali

M00702000

May 2, 2023

1 Abstract

The objective of this project revolves around determining the effectiveness of Security Onion as an operating system specialised in threat detection. To address this, this project involves carrying out the necessary research and evaluating the deployment of Security Onion by injecting low-risk threats in a controlled environment. The results of this deployment are examined though observing the system's health performance and security logs. The insight gained from evaluating the findings indicates that Security Onion's default configurations fulfill the need for an efficient system, and although this may not cater to everyone it is an effective configuration for first-time users to gain familiarity with the system's functionality. A further note to be made based on the previous statement is that the documentation for Security Onion has a broad guide and therefore the system configurations can be manipulated to accommodate various needs.

Contents

Declaration of Authenticity

I, Silvia Aminul, declare that this dissertation is solely my work,
and affirm that all sources used in this report have been duly acknowledged
through citation.

Signature *Date*
Silvia Aminul May 2, 2023

Acknowledgment

I would like to express my gratitude to my loving family and friends for their unwavering support and encouragement throughout completing this dissertation. Their constant motivation and belief in me have been of immense support.
I am grateful to my supervisor, Dr. Kamran Ali, for his guidance, expertise, and continuous support. His insightful feedback played an important role in shaping this dissertation.

2 Introduction

This project revolves around Security Onion, a Linux-based distribution designed to detect all types of threats including network intrusions. This distribution is based on the operating system Ubuntu, comprising multiple open-source security tools including Suricata, which is the primary Network Intrusion Detection system (NIDS).

This project focuses on the NIDS performance commonly used to monitor network traffic and identify potential security threats such as malicious activity using signature and behavior-based detections.

2.1 Problem Definition

The scope of this report transitions from the research aspect to implementation to determine whether Security Onion is effective through evaluating the systems detection performance when threats are sent to the targetted system.

In this report, different studies are reviewed to understand network threats, Security Onion's behaviour in detecting these threats, and strategies used to prevent threats from infiltrating the system. This report also involves analysing different topologies, doing so allows a more effective topology to be designed. Furthermore, two NIDS infrastructures are examined with the addition of analysing studies comprised of the comparison between inbuilt NIDS. Halfway through this report, the project's research transition towards implementing the initialisation steps to sending attacks, this allows recording results and evaluation of whether the system is effective in its default settings.

2.2 Aim

2.3 Objectives

- To study different forms of network threats.
- To design and analyse the implementation of a simulated network.
- To implement Security Onion as a proxy.
- To examine Security Onion implemented in the simulated network.
- To evaluate how Suricata recognises network threats.
- To conclude on the effectiveness of Security Network to detect network intrusions.

2.4 Findings

In the research side of this project, it was found that sources on Security Onion's NIDS are scarce as not many studies have been done on comparing the inbuilt network intrusion detection system. A general observation of the comparison studies points to Suricata being a better performer than other inbuilt network intrusion detection systems. Given an updated version of Security Onion has been posted on the website, the results from this project are going to be potentially better and might be considered a new benchmark for researchers comparing NIDS in Security Onion. Moving onto the implementation aspect, many issues had been encountered and resolved while implementing, however, whether Security Onion is an effective system is later found to be debatable, as although it prevents the attacks, its flagging mechanism in the default setting may not be suitable for everyone.

3 Literature Review

As the reliance on technology surges, network security becomes a high-priority aspect due to the significant amount of personal and financial information travelling through public networks. This makes day-to-day users prone to network integrity breaches which have also been a growing threat amongst large organisations.
To further emphasise the importance of having a secure network, the most common form of attack in the UK amongst medium-sized businesses around 83% is phishing, and around one in five (21%) can result in experiencing Denial Of Service (DOS), malware, or ransomware [1]. Security breaches occur every day amongst businesses including monetary losses in very large sums amongst organisations; there are many cases of cyber-theft whereby attackers leave no trace of attacks, Bangladesh Bank was no exception to be a victim of losing nearly 81 million dollars, this occurred by disabling the ink cartridge of a printer to prevent alerting staff of any transaction during the intrusion [2]. One potential reason for malicious network interceptions may be the lack of tools that are easy to implement.
This research of this report will evaluate whether this framework is powerful enough to detect all forms of network intrusions.

This report involves researching prominent ways of successful network intrusions and methods in which NIDS such as Suricata can detect suspicious requests or unknown fingerprints that fall into the category of malicious data and alert through Network Intrusion Detection Systems (NIDS).

Researching this would provide sufficient knowledge and insight into stimulating a networking environment with Security Onion implemented as the network host and Kali Linux as the source environment for the attack. The data gathered can then be monitored with the help of further research with tools such as Elastic Search which can perform optimised searches through the database of logged request and Kibana which retrieves this data and present it visually or simply log into the Security Onion Console.
To execute this project, further research is required on the following objectives:

- To review a variety of studies on different forms of network threats, this allows understanding of how sending bad requests functions and observe as well as record Security Onion's behaviour.
- To analyse the implementation of a simulated network. Doing this minimises the risk of network mishaps. Regardless of the environment, it is always good practice to design a network topology before implementing it.
- To review different implementations of NID systems and compare their effectiveness.

Reviewing the above aids in gaining further clarity, the knowledge gained from the literature review will carry over to the initial stages of the project.

3.1 Network Threats

The constant increase in new techniques developed and used by attackers is currently a growing theme whose severity in the future is unpredictable [3]. One potential reason for malicious network interceptions may be the lack of tools that are easy to implement. Understanding different network threats is crucial to improve the security of the cloud computing environment.
As part of this study, multiple implementations of Intrusion Detection Systems (IDS) are explored to understand how their speed and automation in detecting malicious traffic can be a key factor in cloud computing environments with high-speed networks.

Although this project does not involve cloud computing, Figure 1 extracted from a journal by Mikail et al. is an example of an organisation with a large network that may consider deploying various implementations of IDS to have secure network infrastructure.

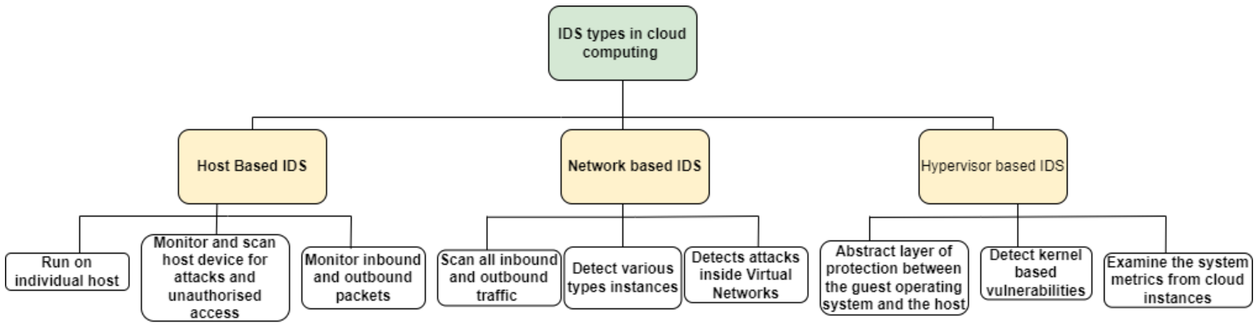

IDS types in cloud computing
Host Based IDS
Network based IDS
Hypervisor based IDS
Run on individual host
Monitor and scan host device for attacks and unauthorised access
Monitor inbound and outbound packets
Scan all inbound and outbound traffic
Detect various types instances
Detects attacks inside Virtual Networks
Abstract layer of protection between the guest operating system and the host
Detect kernel based vulnerabilities
Examine the system metrics from cloud instances

Figure 1: three types of IDS that can be combined or separately used in cloud computing. Systems.[4]

In Figure 1, the diagram conveys that in a secure large environment, a network may comprise many IDS to prevent a majority of threats. One may prefer to have many IDS deployed to have a secure environment and therefore pointing to a very important note; solely deploying a network-based IDS cannot guarantee a near-to-secure environment.

Various industry standard implementations of IDS are available for use in large network infrastructures. However, in this project, only threats captured by Suricata are studied for this project. As Security Onion is using a state-of-the-art intrusion detection technique by leveraging Suricata, the performance and accuracy of the system are proven to be high under a high network load [4].

Numerous methods have been suggested in the literature to implement Security Onion for different purposes. However, this project will focus solely on implementing them in a general simulated network and leave the flexibility to be extended further for a specific use case.

According to Hindy et al., the most common type of threats are Denial Of Service (DoS) and Distributed Denial Of Service(DDoS), this is whereby a malicious user occupies the network by flooding it with resources, causing the system to become unresponsive, an example of this daily would be a big event or breaking news [5]. Dos and DDoS can fall into the following categories:

- Flood Attack - refers to flooding the system with high volumes of requests exhausting the system's bandwidth making the network unavailable for other users to use [6].
- Amplification Attack - also known as Network Time Protocol (NTP) amplification attack, causes the network to become unavailable through the use of one or more

compromised applications sending more UDP traffic than the targeted network can handle [6].

- Protocol Exploit - is whereby an attacker alters internal content by bugging the network's protocol to exhaust the operating system's resources [6].
- Malformed Packets - causes the network service to be overloaded with packets that may seem clean from threats, but indeed are malformed, bugged to fail the service [7].

Several other attacks are in common with the public cloud in terms of network threats that impact users' data and the integrity of the organisation. These attacks include:

- **Smurf Attack** also known as an ICMP DoS is an example of a Reflection-Based flooding attack, in which the attacker creates a fake echo containing the target IP addresses which is then broadcasted to all hosts in that network. The hosts then reply to the broadcast accumulating into volumes of Internet Control Message Protocol (ICMP) packets making the network inoperable. ICMP is used by routers to test the reachability of a destination [8].

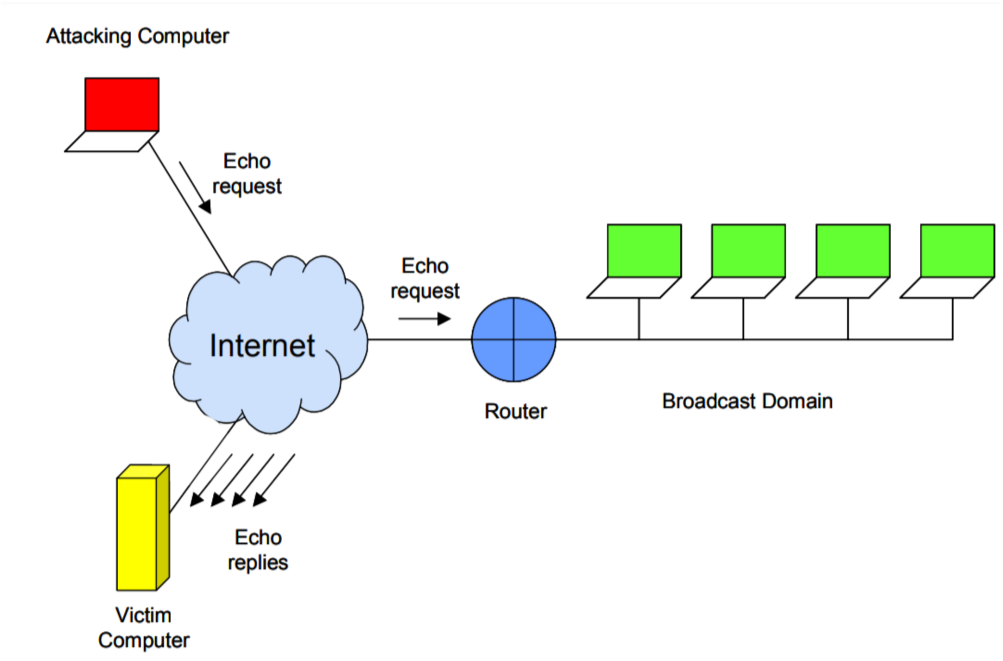


Figure 2: Smurf attack's Infrastructure [8]

```
hping3 ####.####.####.####-q -ICMP -C 3 -K 4 [1]
```

In this command, the hashes represent the IP address of the victim and must be swapped for the hashes [9].

- **SYN flood** This type of attack occurs by looking for a known vulnerability in the Transmission Control Protocol (TCP) connection which validates authentication using a sequence of communication. The sequence is initiated when the client sends a SYN message to the server, the server then replies with the SYN message and attaches its message. If the client can decrypt the message and send it back to the server with the attached encrypted message (in this case "ACK"), the authentication is deemed successful.
  An SYN attack occurs when an attacker floods the server with multiple SYN requests, the server in turn, replies and waits for the client to respond this uses up the resource while preventing the target network to be accessed.

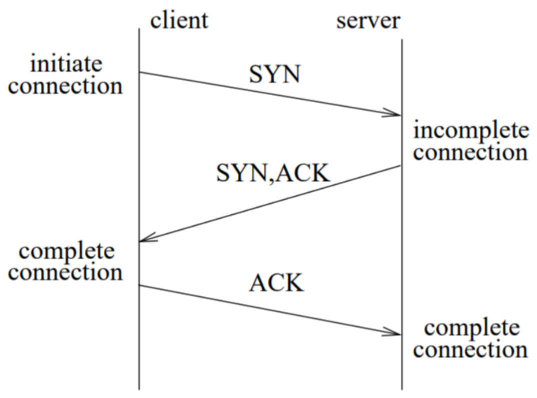


Figure 3: SYN flood's Infrastructure [10]

In this project, this attack can be implemented by setting up Kali Linux and using the following command: [9]

```
hping3 ####.####.####.#### -q -n -d 90 -S -p xx --flood -rand-
source [2]
```

[1]this was extracted from a thesis on Testing the Security Onion by Kayla Jansen. [9]
[2]this was extracted from a thesis on Testing the Security Onion by Kayla Jansen. [9]

To implement this command successfully the IP address of the victim must be swapped for the hashes, then -d 90 can be changed to the size of packets the attacker would like to send, and the x needs to be changed for the port number [9].

- **UDP Flood** this is a User Datagram Protocol (UDP) whereby the attacker issues an attack on the Master Control Program (MCP) upon receiving the relevant information the MCP issues instructions to compromised systems to mask themselves using different IP addresses and repeatedly sending UDP packets directing to a random port. This causes the system to listen to the port, causing the system to wait for a response while more UDP packets are sent, eventually, this causes the system to crash. The figure 4 below is an illustration made to understand this concept better.

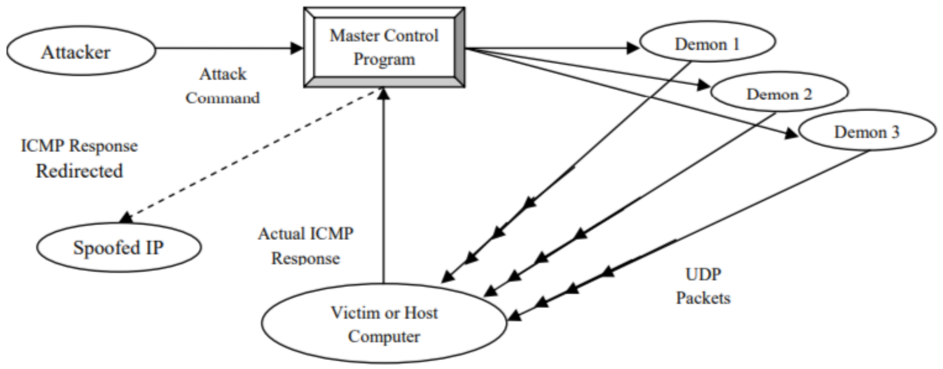


Figure 4: UDP flood's Infrastructure [11]

To carry out a similar attack on Security Onion the following command comes in to use:

```
    sudo hping3 --flood -a \#.\#.\#.\# -2 -p x \#.\#.\#.\# [3]
```

As mentioned before the hashes represent IP addresses, the first set of hashes is the IP address the attacker masks themselves as. The last set of hashes the victim's IP addresses, x represents the port the victim's system is going to listen to.

Other types of attack include:

- **The ping of Death** is another type of attack that occurs when the size of the packet attached to the ping sent exceeds 65,535 bytes which are significantly large for an Internet Protocol Version 4 (IPv4) router to process, therefore causing the packet to

[3]this was extracted from a thesis on Testing the Security Onion by Kayla Jansen [9].

split. Attempting to reassemble the packets causes the system to overflow and the system goes into DoS [6].

- **XSS** also known as Cross-Site Scripting is an attack that occurs around sending a bugged client-side web application to a user. The attacker initially alters the website by injecting a bad script and defacing it or redirecting the user to a fake website. The attacker takes advantage of this by having access to sensitive data such as cookies.
- **Packet forging** also known as an injection is another type of attack where the packets sent can perform malicious activities such as accessing sensitive information [5].
- **IP Spoofing** is used to gain unauthorized access, firstly the attackers must mask themselves by using a credible IP address that the target IP recognises as credible and trusts, the attacker would then disable the trusted IP's ability to access the network resources to prevent any unwanted intervention, the authentication process would then start, if successful, the attacker may now able to send a packet that has been tampered to have access in the future [12].
- **DNS spoofing** In this network threat the attacker redirects the user by inserting or altering the IP address to what seems like a legitimate website or server that is under the attacker's control. The following is a sequence in a normal scenario:
  - The user sends a query to the DNS server.
  - The DNS server replies to the user with the IP address.
  - The user then connects to the IP address given by the server; redirecting them to the destination the user initially made a query on.

  The following is a sequence of scenarios where an attacker is involved:
  - The attacker identifies the server used by the user and tampers the IP address of a query to an IP the attacker can control without seeming suspicious.
  - The user sends a query to the DNS server.
  - The DNS server replies to the user with the IP address modified by the attacker.
  - The user then connects to the IP address given by the attacker masked as the server; redirecting them to a fake page [12].
- **ARP spoofing (ARP poisoning)** ARP refers to Address Resolution Protocol, the attacker may use spoofing tools such as Parasiten, Ettercap, or ARPoison to look for

a MAC and IP address to send a tampered ARP request to convince the target network to reply allowing the tools mentioned above to tamper, redirect another type of attacks such intercepting communication channels.

- **Cloning** is an attack commonly used by attackers to impersonate a user to gain information [5], there are real cases such as the NATO'S senior commander Admiral Jones identity being cloned, this was done to gain private information on government officials receiving and accepting Facebook friendship requests from a cloned profile of Admiral Jones [13].

Common consensus by other researchers suggests that Security Onion is an off-the-shelf framework that provides detection and prevention of attacks on public and private clouds, as well as the simulated network this project will use.

This report will aim to explore these attacks and implement Security Onion to detect and prevent these attacks in the project.

3.2 Different implementation of Network Intrusion Detection systems

Network Intrusion System (NIDS) refers to tools implemented in a framework to prevent incoming threats in a network to alter internal settings or retrieve personal information. NIDS is especially important due to the rise in cybercrime and the surging demand for a larger bandwidth to process packets at a higher speed. There are several approaches in which way users or large organisations decide to protect their environment. In this report NIDS' will be analysed and compared; this is done to extract information on how each infrastructure processes packets and how they single malicious packets.

Due to the rampantly increased rates in cybercrime, it has become significantly difficult to process vast amounts of network packets using IDS such as Snort due to single threading, single threading can be exemplified by Figure 5 below; a diagram representing Snort's infrastructure originally published by (Gómez et al., 2009)[14].

3.2.1 Snort

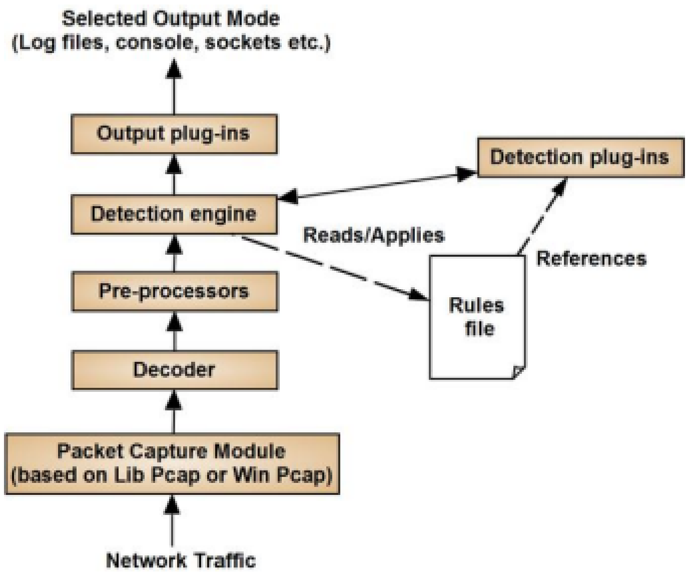


Figure 5: Snort's Infrastructure [15]

Initially, starting from the bottom of the infrastructure, network traffic, in essence, the packets, are fed into a network capture module, whereby an external library such as Lib Pcap is used as required by Snort to sniff packets, the decoder then processes these raw packets through a sequence of decoders each designated to decode specific components of the protocol such as their settings, packet size, and other hidden abnormalities. In the case a suspicious packet is detected, the decoder would then generate an alert [15]. Next, the purpose of a pre-processor is to improve overall performance by preventing any form of rouge data from going into the next level, in essence, the preprocessor must ensure it combs through the traffic of data before sending it to the detection engine. To ensure this process is successful, the preprocessors must be configured appropriately. As further explained in the book Snort Intrusion Detection and Prevention Toolkit published by Beale, Baker, and Esler, many types of preprocessors can perform useful activities before packets reach the detection engine also known as the rule engine [16], the table below is a few commonly known preprocessor names and their function:

Preprocessor	Function
Frag3	This preprocessor consists of a module that can handle fragmentation-based attacks on a user-specified target by re-assembling IP fragments. It uses the target's TCP/IP stack to model its fragmentation behaviour. This preprocessor follows two steps • Global initialization phase • Definition of defragmentation engines This preprocessor is useful due to its extensive abilities, for instance, it may be required to apply specific policies to certain IP groups whilst having a default policy for all other traffic, this preprocessor can be configured to run multiple engines at the same time.
Stream4	This preprocessor is constructed of two modules: • Stateful analysis is a module that is used for detecting TCP state issues and port scans, and it requires careful configuration to optimize its performance • Stream reassembly is known to perform complete stream reassembly for TCP and can manage both client and server streams, with options for specifying which ports to perform reassembly on and other directives.
sfPortscan	A preprocessor designed that can detect scan techniques, it allows users to specify which protocols to monitor the type of scan and sensitivity level. To assign the flow of direction to connectionless protocols like UDP and ICMP flow processor can be used with these preprocessors, to avoid generating multiple alerts for the same scan packets it is recommended to disable evasion alerts in the stream4 preprocessor.

Going into the detection engine, a set of predefined rules are applied to the incoming data, these rules are specially structured to analyse these packets for:

- A known signature of the attack to be detected [14].

- The detection plug-ins permit the configuration of the engine [14].

The output plug-ins would clarify where and how different types of alerts should be logged. Suricata is considered to be a more computationally advanced NIDS as its ability to overcome the limitations of single threading NIDS surpasses the current concerns with the demand of larger bandwidth by processing large amounts of packets while maintaining a good speed; this is done through a process called multithreading whereas Snort would ignore packets that it gets flooded by due to not receiving enough bandwidth [17], Suricatas infrastructure on how it deals with managing multiple packets in the same environment is displayed in Figure 6, handling many packets at the same time, has become especially important with malicious penetration methods constantly evolving and the necessity of having a rigorous process to ensure a secure environment is at hand.

3.2.2 Suricata

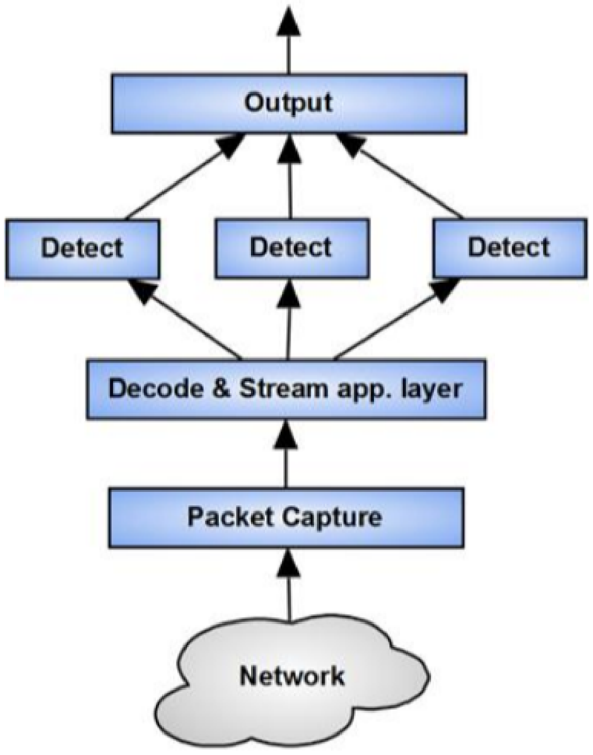


Figure 6: Suricata's Infrastructure [15]

To begin, the traffic goes into the Packet capture to process the packets for analysis, then, just like Snort the packets go into the decoding layer with an additional feature called the stream app layer which essentially is a reassembly engine, this helps to assemble packets from incoming traffic to be sorted in a form of queue ready to go into the thread engine. Before going into the next layer, here, a Queue Handler function is invoked where packets are either being fetched into a thread or ditched, going through this layer consists of multiple threads detectors that fetch packets from the thread and detect malicious threats by putting the signature against a database consisting predefined set of signatures known to be threats to base the decision whether the packet can enter the network or be dropped [15].

3.3 Analysing Network Simulation

To execute the implementation of this project, a network topology must be first designed and analysed for faults or further adjustments, doing so may allow a more successful deployment.

Simulations are considered a necessity to help in the process of building new technologies as this gives developers an idea of what is to be expected had their prototype has been developed in a real environment at a larger scale where the impact is real. A network simulator refers to an interactive software tool that allows network designers/engineers to design a topology at a smaller case with little to no impact, this allows designers to observe and calculate the network's behaviour ensuring it is suitable for real-life implementation. Implementing a network in a real environment can be very difficult and costly, thus observing how a network behaves in a simulated environment when a new protocol is implemented may aid the process of ensuring the network's topology is safe to deploy in a real environment with little to no errors which are important to prevent accidents to occur.

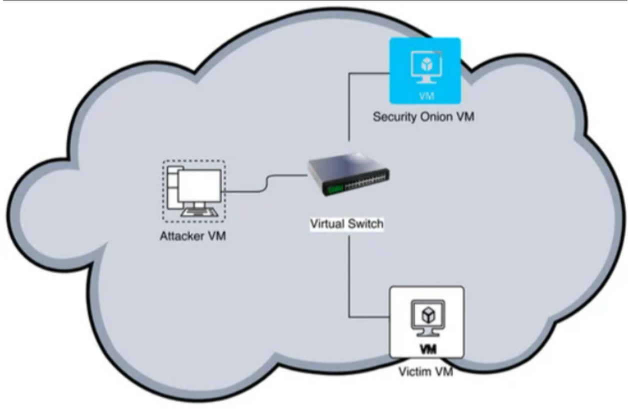


Figure 7: Network Topology [4]

The experiment carried out by Mikail and Pranggono revolved around Snort and testing the security of cloud computing. To experiment with integrated tools Security Onion comes with, Figure 7 used by Mikail and Pranggono, which revolves around three VMs:

- Security Onion generates alerts through the intrusion detection system.
- Kali Linux acts as the attacker's system.
- Metasploitable acts as the victim's system.

All three VMs in this configuration are connected through a switch. Although Figure 7 is a single realistic configuration, a concern this project can alter is releasing more than one attack. Back to this project, in this experiment one NIDS is being tested whose results will be then put against existing results.

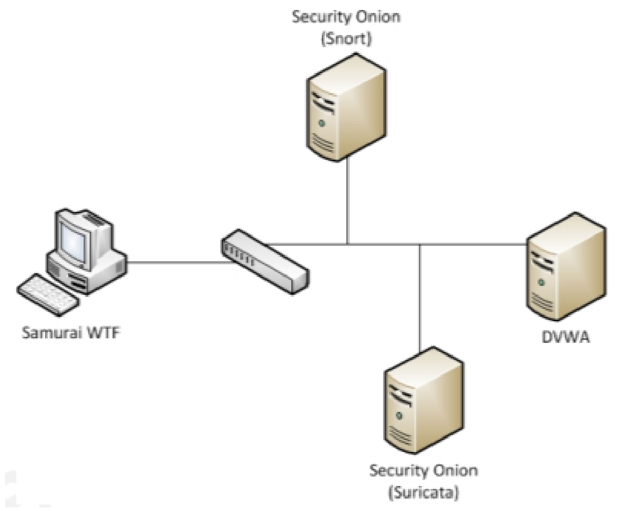


Figure 8: Network Topology [18]

Another network simulation experiment executed by Deuble, A. Security involved testing both Suricata and Snort on detecting web application-based attacks. the topology is similar to the previous Figure 8; however this topology consist of four components: [18]

- Security Onion alerts and prevents intrusion detection using Suricata.
- Security Onion alerts and prevents intrusion detection using Snort.
- Samurai distribution acts as the attacker's system.

- Damn Vulnerable Web Application acts as the victim's system.

All 4 VMs in this configuration are connected through a switch. In this Figure 8 all four components are interconnected, and the topology gives more accurate results as both Snort and Suricata generate alerts upon detection.

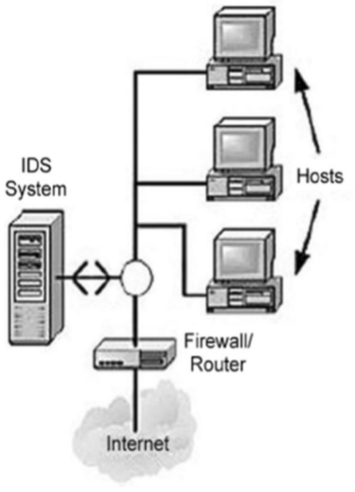


Figure 9: Network Topology [17]

One last example of topology is in Figure 9 where there is no experiment involved. Although this is what typical settings look like it is not applicable as in this project a controlled environment is required to send threats that can be damaging to the system.

4 Methodology

An ample amount of research has been done to find an IDS that is the most fitting to cover all ranges of threats and all sizes of packets, Suricata is one of many NIDS that is considered early in its life cycle and has a few scholarly activities but not many extensive documentations were found, this might convey why there is a lack of organisation and day to users implementing this NIDS in their system. This section of the literature review covers the research done on different NIDS, methodologies findings conclusions based on the findings.

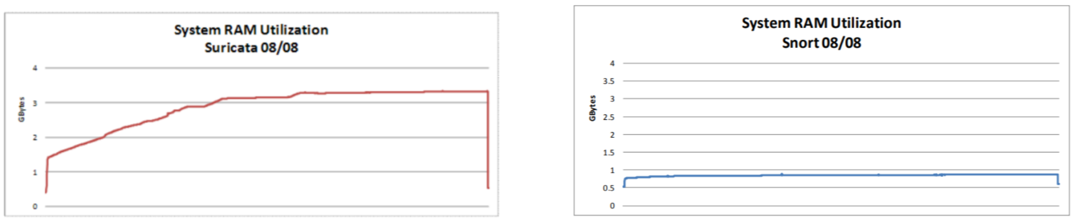
One evident difference pointed out in the article by Fekolkin mentions a key factor that differentiates Suricata and Snort methods of threading packets, Snort was designed to run on single-core processors, this can become costly in the future and disadvantageous to users who value performance, and as the network grows. However, Snort can be seen progressing to tackle these issues in the near future, through a fairly new Hybrid IDS called H-Snort which is a redesign of the preprocessor to essentially detect anomalies and many more extensive features [14]. On the other hand, Suricata's multithreading engine makes use of the computational power by having multi-core processing units allowing more efficiency in processing multiple large packets while maintaining speed, the opportunity to have a scalable IDS as users can configure the number of thread detectors to run in the same environment can be advantageous from a financial perspective [15].

An article published by Albin and Rowe carried out a comparison experiment between the two NIDS by sending threats in a controlled environment. Albin measured detection performance and accuracy in an active virtual environment by measuring the following:

- CPU percentage.
- Memory usage.
- Network usage.

To further note, the network tap had been used to compare the two NIDS had a network bandwidth of 20 Gbps which is a considerably large pipe where traffic has an average of 200Mbps [19], the signatures came from two IDS rule maintainers, SourceFire Vulnerability Research Team (VRT) and Emerging Threats (ET). In total three experiments were carried out. Experiment one was carried out to compare the monitoring of the traffic, Albin noted the results conveyed that although Suricata used more computational resources Suricata still managed to use less CPU (50-60% range) than Snort (60-70% range), suggesting implementing Suricata might not have

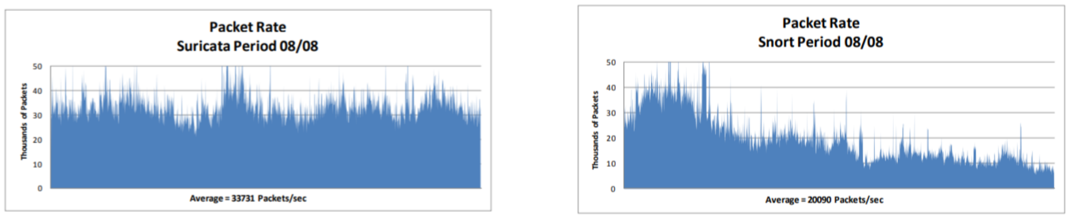
In the first experiment, Albin et al. also noted that Suricata is "more memory-intensive", the memory usage started from 1.5 GB to 3.3Gb whereas for Snort it seems to be a bit more consistent from 0.8Gb to 1Gb [19]. Below are two side-by-side Figures 10 Suricata and Snort representing RAM usage:



(a) Suricata (b) Snort

Figure 10: RAM usage by the two NIDS [19]

Continuing, due to technical issues although the network traffic rate could not be recorded, the averages are in the Figures 11 below.



(a) Suricata 33,731/s packets (b) 20,090/s packets

Figure 11: Network packets processed per second [19]

Given the findings, a similar methodology to this section will be used as a guide when evaluating the findings of this project.

5 Analysis and Design

From this section onwards the report will be a record of the transition from research to implementation.

5.1 Designing The Network Topology

The figure below is a diagram that has been designed for this project:

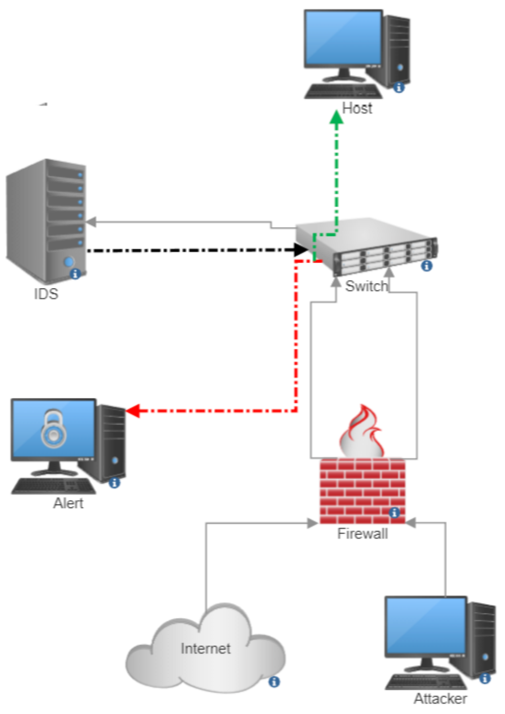
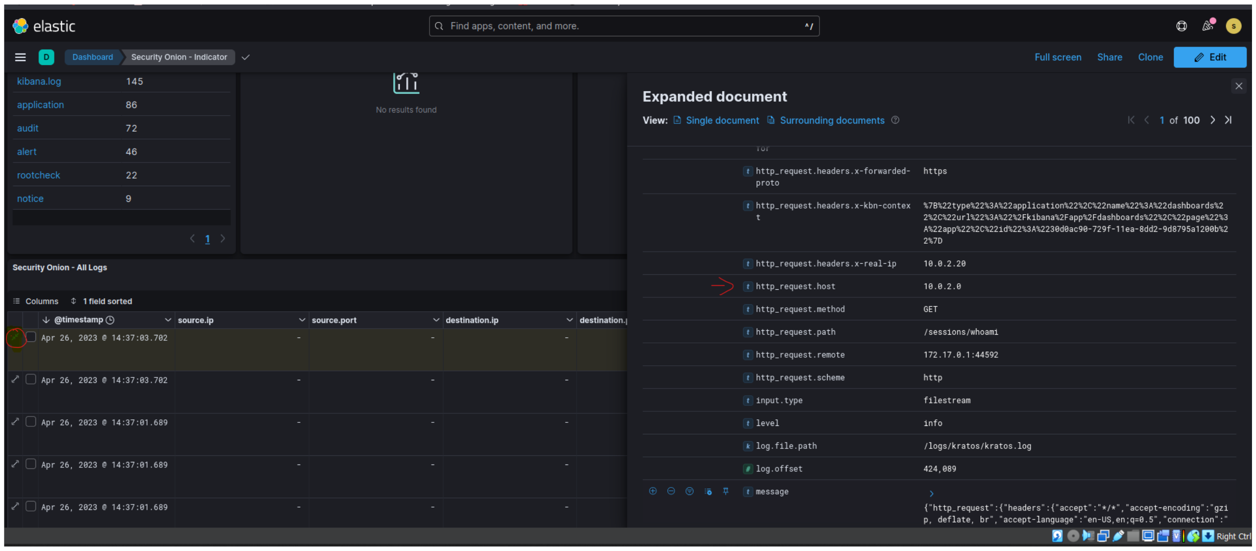


Figure 12: A design of this project's topology

The topology mapped in figure 12 displays how this project is going to set the environment for network threats in the Security onion, an attacker is required to ensure safety is not compromised, and the threats would travel from the source IP address of the Kali Linux environment to the target address on the Security Onions environment. The attacker's device Kali in this case, can for instance flood the host's device with pings, overflowing the system enough to create a DOS attack, this would be detected by the Intrusion Detection System (IDS) which would be flag alerts on the dashboard and log it into the system to be viewed. One thing to be noted in this project is that a different route has been taken than what has been proposed in the proposal. Security Onion as a proxy in-between the network and the outside network layer in a simulated network is longer being implemented as it's not as realistic as it would be if real virtual environments are used.

5.2 Data Collection

To have a look at the alert system, as mentioned previously the user would be required to access the Security Onion's dashboard, testing around the filters and adding tools gives a good understanding of each threat that is sent, for instance, in an attack where the IP source IP is not specified, the user would need to expand on the unspecified attack the user can access the dashboard to this link which is a good site to start with for users with all the necessary filters they should look for when testing attacks. As expected in the table there's no IP address pointing to where this threat is coming from. In this case, adding in the following column would be required.



kibana.log	145
application	86
audit	72
alert	46
rootcheck	22
notice	9

Security Onion - All Logs

Columns 1 field sorted

@timestamp	source.ip	source.port	destination.ip	destination.p
Apr 26, 2023 @ 14:37:03.702	-	-	-	
Apr 26, 2023 @ 14:37:03.702	-	-	-	
Apr 26, 2023 @ 14:37:01.689	-	-	-	
Apr 26, 2023 @ 14:37:01.689	-	-	-	
Apr 26, 2023 @ 14:37:01.689	-	-	-	

Expanded document

View: Single document Surrounding documents

1 of 100

http_request.headers.x-forwarded-proto	https
http_request.headers.x-kbn-context	%7B%22type%22%3A%22application%22%2C%22name%22%3A%22dashboards%22%2C%22url%22%3A%22%2Fkibana%2Fapp%2Fdashboards%22%2C%22page%22%3A%22app%22%2C%22id%22%3A%2230d0ac90-729f-11ea-8dd2-9d8795a1200b%22%7D
http_request.headers.x-real-ip	10.0.2.20
http_request.host	10.0.2.0
http_request.method	GET
http_request.path	/sessions/whoami
http_request.remote	172.17.0.1:44592
http_request.scheme	http
input.type	filestream
level	info
log.file.path	/logs/kratos/kratos.log
log.offset	424,089
message	{"http_request":{"headers":{"accept":"*/*","accept-encoding":"gzip, deflate, br","accept-language":"en-US,en;q=0.5","connection":"

Figure 13: Security Logs from the SO's dashboard.

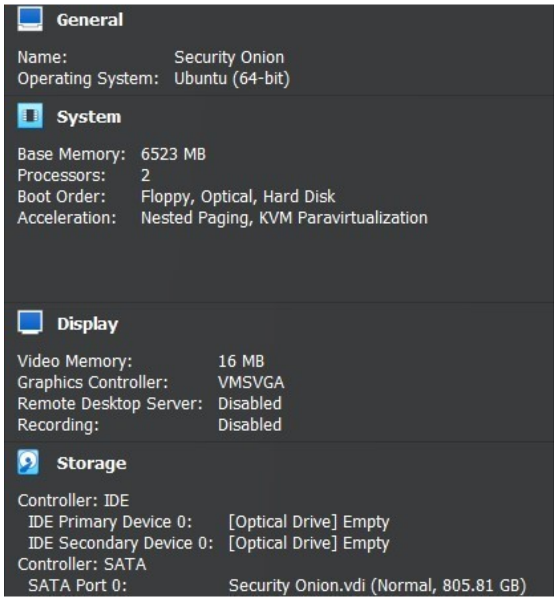
This data can further be accessed in JSON format under "message", in the case the user would like more detail in a compact version. The flooding threats to be sent will be sent for about 45 seconds.

Overall implementing the architecture in 12 has been a straightforward task due to decision to design the topology first. Because of this, it is much more convenient to collect data in this way.

6 Implementation

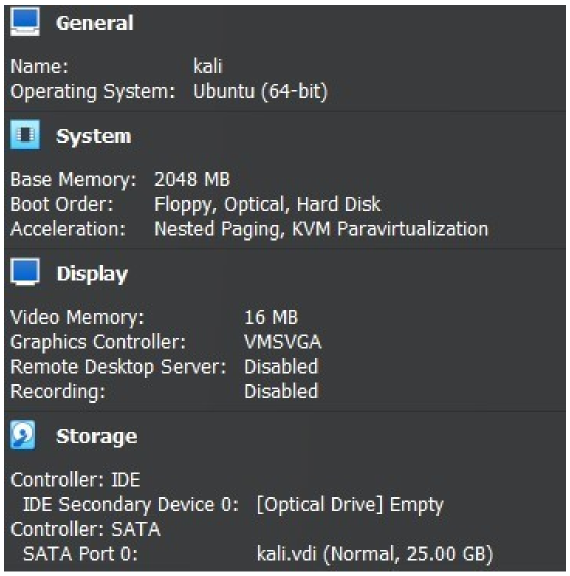
6.1 Installing Security Onion and Kali Linux

To initiate the installing program a virtual box is required, this allows virtual machines to be installed. This section will cover the installation process of Kali and Security Onion. The basic configuration for Security Onion and Kali are as follows:



```
General
Name:               Security Onion
Operating System:   Ubuntu (64-bit)

System
Base Memory:   6523 MB
Processors:    2
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  Nested Paging, KVM Paravirtualization

Display
Video Memory:            16 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: IDE
  IDE Primary Device 0:    [Optical Drive] Empty
  IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:             Security Onion.vdi (Normal, 805.81 GB)
```

(a) Security Onion's settings



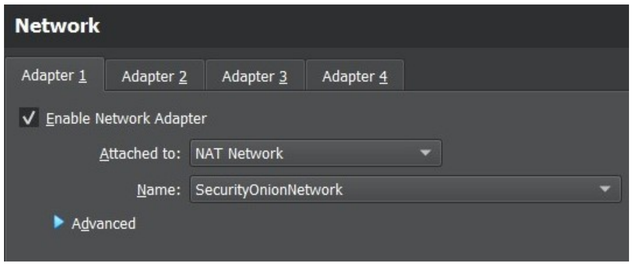
```
General
Name:               kali
Operating System:   Ubuntu (64-bit)

System
Base Memory:   2048 MB
Boot Order:    Floppy, Optical, Hard Disk
Acceleration:  Nested Paging, KVM Paravirtualization

Display
Video Memory:            16 MB
Graphics Controller:     VMSVGA
Remote Desktop Server:   Disabled
Recording:               Disabled

Storage
Controller: IDE
  IDE Secondary Device 0:  [Optical Drive] Empty
Controller: SATA
  SATA Port 0:             kali.vdi (Normal, 25.00 GB)
```

(b) Kali 's settings

Figure 14: Network packets processed per second.

Before proceeding to start an optical disk with the drive of the machine installed should be added, the iso file can be downloaded from the documentation website. Although this can be done at the start of the configuration, it can also be added later by going to the setting, then storage then adding the file through the plus button of the controller IDE.

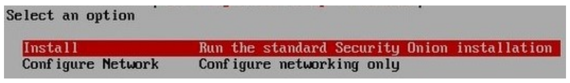
To ensure the virtual machines can communicate in the settings under the network the following should be the configuration:



```
Network
Adapter 1   Adapter 2   Adapter 3   Adapter 4
[✓] Enable Network Adapter
    Attached to: NAT Network
           Name: SecurityOnionNetwork
  ▶ Advanced
```

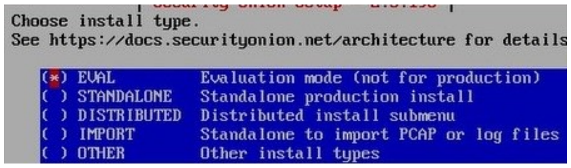
Figure 15: Network settings for both Security Onion and Kali

Starting with Security Onion a prompt message will appear requesting for a username and password.



```
Select an option

  Install           Run the standard Security Onion installation
  Configure Network Configure networking only
```

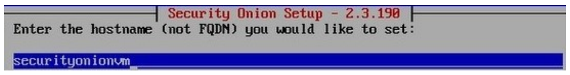
(a) Username and password needs to be set



```
Choose install type.
See https://docs.securityonion.net/architecture for details

(*) EVAL          Evaluation mode (not for production)
( ) STANDALONE    Standalone production install
( ) DISTRIBUTED   Distributed install submenu
( ) IMPORT        Standalone to import PCAP or log files
( ) OTHER         Other install types
```

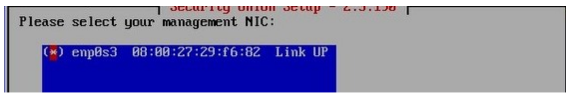
(b) Depending on the type of installation the setup will give different configuration outputs.

Figure 16: The beginning of the configuration for Security Onion.



```
Security Onion Setup - 2.3.190
Enter the hostname (not FQDN) you would like to set:

securityonionvm
```

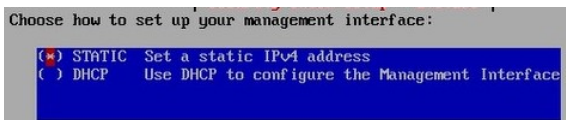
(a) The hostname can be changed or left at the default name.



```
Please select your management NIC:

(*) enp0s3  08:00:27:29:f6:82  Link UP
```

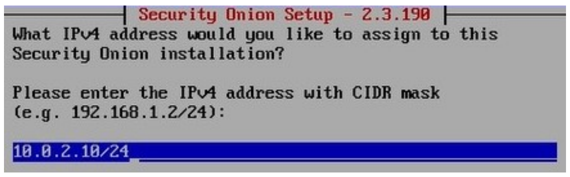
(b) Management NIC

Figure 17: Configuring hostname and Management NIC



```
Choose how to set up your management interface:

(*) STATIC  Set a static IPv4 address
( ) DHCP    Use DHCP to configure the Management Interface
```

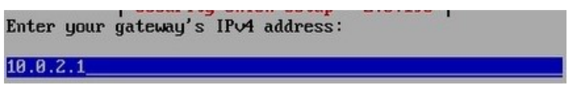
(a) In order to establish communications, IP was chosen



```
Security Onion Setup - 2.3.190
What IPv4 address would you like to assign to this
Security Onion installation?

Please enter the IPv4 address with CIDR mask
(e.g. 192.168.1.2/24):

10.0.2.10/24
```

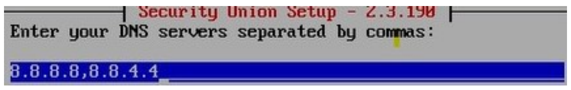
(b) Setting Security Onion's IP address

Figure 18: Setting up the network.



```
Enter your gateway's IPv4 address:

10.0.2.1
```

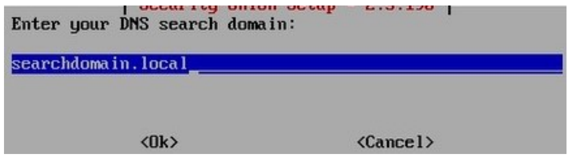
(a) Setting the gateway IP address



```
Security Onion Setup - 2.3.190
Enter your DNS servers separated by commas:

8.8.8.8,8.8.4.4
```

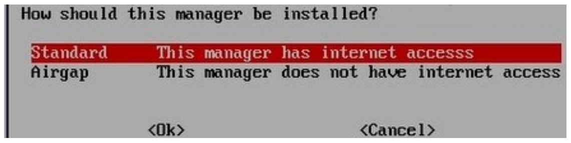
(b) The DNS server address has been kept to default.

Figure 19: Setting SO's gateway IP and DNS server.



```
Enter your DNS search domain:

searchdomain.local

        <Ok>                <Cancel>
```

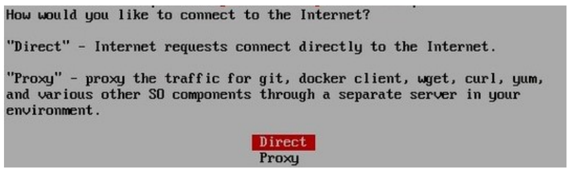
(a) The DNS search domain is kept to default.



```
How should this manager be installed?

Standard     This manager has internet access
Airgap       This manager does not have internet access

        <Ok>                <Cancel>
```

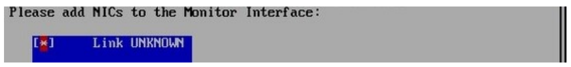
(b) Internet access is required in this project .

Figure 20: Setting up the DNS search domain and installing the manager.



```
How would you like to connect to the Internet?

"Direct" - Internet requests connect directly to the Internet.

"Proxy" - proxy the traffic for git, docker client, wget, curl, yum,
and various other SO components through a separate server in your
environment.

                        Direct
                        Proxy
```

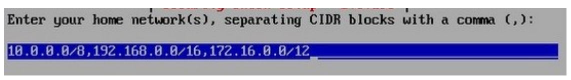
(a) Username and password needs to be set



```
Please add NICs to the Monitor Interface:
  [*]    Link UNKNOWN
```

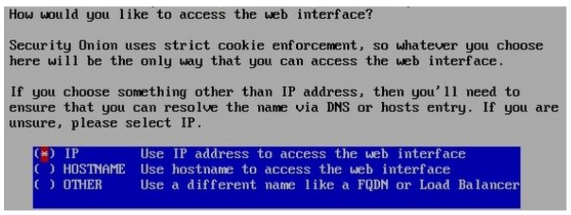
(b) Only one option was given as initially only one network adapter was configured; see Figure 15

Figure 21: Configuring access to the internet and choosing NICs.



```
Enter your home network(s), separating CIDR blocks with a comma (,):
10.0.0.0/8,192.168.0.0/16,172.16.0.0/12
```

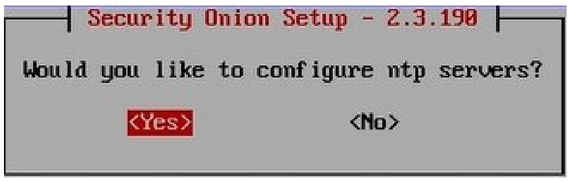
(a) This part is kept to default.



```
How would you like to access the web interface?

Security Onion uses strict cookie enforcement, so whatever you choose
here will be the only way that you can access the web interface.

If you choose something other than IP address, then you'll need to
ensure that you can resolve the name via DNS or hosts entry. If you are
unsure, please select IP.

(*) IP        Use IP address to access the web interface
( ) HOSTNAME  Use hostname to access the web interface
( ) OTHER     Use a different name like a FQDN or Load Balancer
```

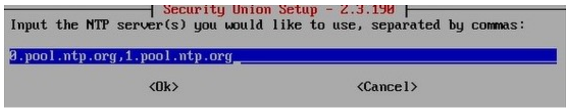
(b) The Virtual machine's communication will be established through configuring the IPs.

Figure 22: Setting up the network.



```
Security Onion Setup - 2.3.190

Would you like to configure ntp servers?

<Yes>              <No>
```

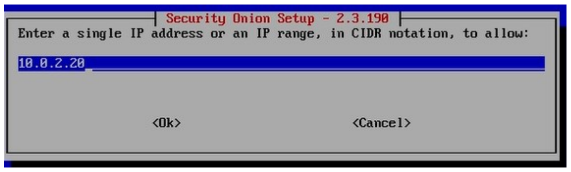
(a) Ntp needs to be configured therefore yes.



```
Security Onion Setup - 2.3.190
Input the NTP server(s) you would like to use, separated by commas:
0.pool.ntp.org,1.pool.ntp.org

<Ok>              <Cancel>
```

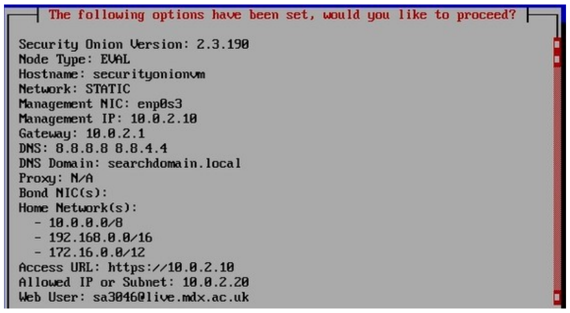
(b) The ntp server has been kept to default.

Figure 23: The beginning of the configuration for Security Onion.



```
Security Onion Setup - 2.3.190
Enter a single IP address or an IP range, in CIDR notation, to allow:
10.0.2.20

<Ok>              <Cancel>
```

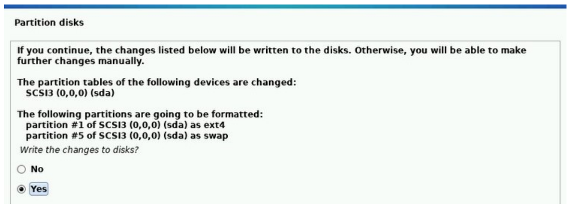
(a) Another IP address later to be configured to the Kali Virtual Machine's (VM) address



```
The following options have been set, would you like to proceed?

Security Onion Version: 2.3.190
Node Type: EVAL
Hostname: securityonionvm
Network: STATIC
Management NIC: enp0s3
Management IP: 10.0.2.10
Gateway: 10.0.2.1
DNS: 8.8.8.8 8.8.4.4
DNS Domain: searchdomain.local
Proxy: N/A
Bond NIC(s):
Home Network(s):
  - 10.0.0.0/8
  - 192.168.0.0/16
  - 172.16.0.0/12
Access URL: https://10.0.2.10
Allowed IP or Subnet: 10.0.2.20
Web User: sa3046@live.mdx.ac.uk
```

(b) A summary of the configuration.

Figure 24: Setting an IP later to be Kali's VM and a summary of SO'S configuration



Partition disks

If you continue, the changes listed below will be written to the disks. Otherwise, you will be able to make further changes manually.

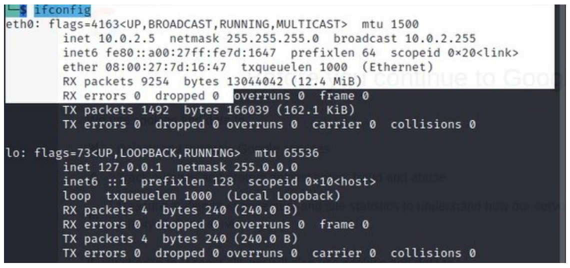
The partition tables of the following devices are changed:
SCSI3 (0,0,0) (sda)

The following partitions are going to be formatted:
partition #1 of SCSI3 (0,0,0) (sda) as ext4
partition #5 of SCSI3 (0,0,0) (sda) as swap

Write the changes to disks?

No [✓ Yes]

(a) All configurations have been left to default in Kali apart from the figure above

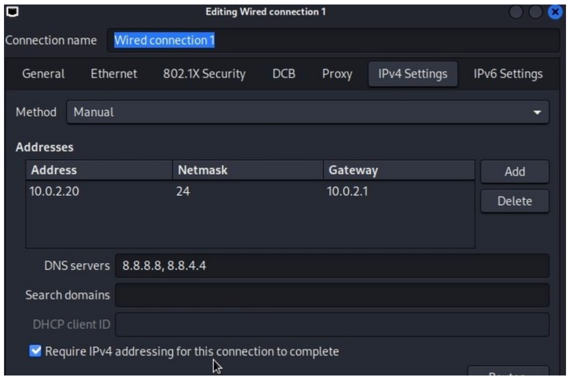


```
└─$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 10.0.2.5  netmask 255.255.255.0  broadcast 10.0.2.255
        inet6 fe80::a00:27ff:fe7d:1647  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:7d:16:47  txqueuelen 1000  (Ethernet)
        RX packets 9254  bytes 13044042 (12.4 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 1492  bytes 166039 (162.1 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 4  bytes 240 (240.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 4  bytes 240 (240.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

(b) An unsuccessful ping suggests there is no connection to the internet and Security Onion.

Figure 25: Configuring the Kali Linux machine and later pinging Security Onion.

6.2 Basic configuration



Connection name: Wired connection 1

Method: Manual

Addresses

Address	Netmask	Gateway
10.0.2.20	24	10.0.2.1

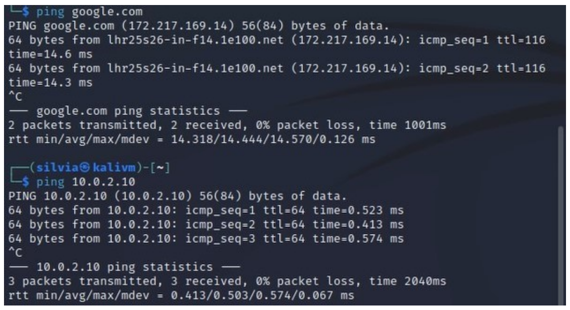
DNS servers: 8.8.8.8, 8.8.4.4

Search domains:

DHCP client ID:

☑ Require IPv4 addressing for this connection to complete

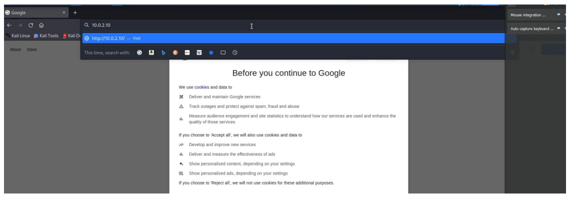
(a) In the settings under network advanced configurations the Security Onion's IP address needs to be entered.



```
└─$ ping google.com
PING google.com (172.217.169.14) 56(84) bytes of data.
64 bytes from lhr25s26-in-f14.1e100.net (172.217.169.14): icmp_seq=1 ttl=116
time=14.6 ms
64 bytes from lhr25s26-in-f14.1e100.net (172.217.169.14): icmp_seq=2 ttl=116
time=14.3 ms
^C
── google.com ping statistics ──
2 packets transmitted, 2 received, 0% packet loss, time 1001ms
rtt min/avg/max/mdev = 14.318/14.444/14.570/0.126 ms

┌──(silvia㉿kalivm)-[~]
└─$ ping 10.0.2.10
PING 10.0.2.10 (10.0.2.10) 56(84) bytes of data.
64 bytes from 10.0.2.10: icmp_seq=1 ttl=64 time=0.523 ms
64 bytes from 10.0.2.10: icmp_seq=2 ttl=64 time=0.413 ms
64 bytes from 10.0.2.10: icmp_seq=3 ttl=64 time=0.574 ms
^C
── 10.0.2.10 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2040ms
rtt min/avg/max/mdev = 0.413/0.503/0.574/0.067 ms
```

(b) If the ping is successful the configuration is also successful

Figure 26: Configuring the Security Onion's IP address in Kali Linux



10.0.2.10

http://10.0.2.10/ — Visit

Before you continue to Google

We use cookies and data to

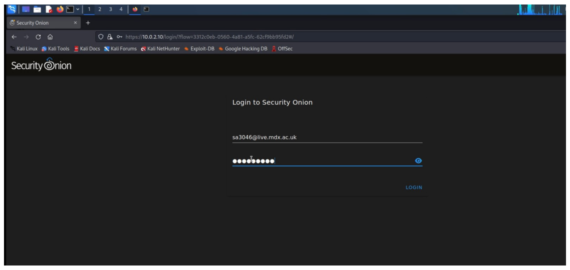
- Deliver and maintain Google services
- Track outages and protect against spam, fraud and abuse
- Measure audience engagement and site statistics to understand how our services are used and enhance the quality of those services

If you choose to 'Accept all', we will also use cookies and data to

- Develop and improve new services
- Deliver and measure the effectiveness of ads
- Show personalised content, depending on your settings
- Show personalised ads, depending on your settings

If you choose to 'Reject all', we will not use cookies for these additional purposes.

(a) Enter the IP Security Onion has been configured with.



https://10.0.2.10/login/?flow=3312c0eb-0560-4a81-a5fc-62cf9bb95fd2#/

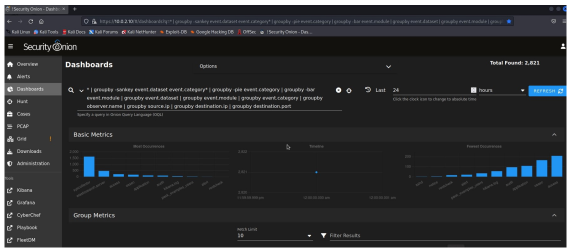
Login to Security Onion

sa3046@live.mdx.ac.uk

●●●●●●●●●●

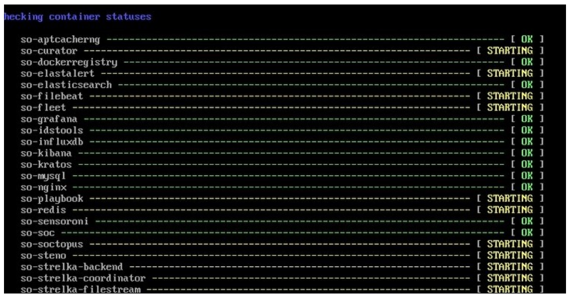
(b) Here the email address entered while configuring Security Onion must be entered to access the dashboard.

Figure 27: Accessing the Security Onion dashboard through Linux



```
* | groupby -sankey event.dataset event.category* | groupby -pie event.category | groupby -bar
event.module | groupby event.dataset | groupby event.module | groupby event.category | groupby
observer.name | groupby source.ip | groupby destination.ip | groupby destination.port
```

(a) The above displays dashboards and all the tools Security Onion comes in-built this may take a while to load.



```
hecking container statuses

  so-aptcacherng -------------------------------------------------------- [ OK ]
  so-curator ---------------------------------------------------------- [ STARTING ]
  so-dockerregistry ----------------------------------------------------- [ OK ]
  so-elastalert ------------------------------------------------------- [ STARTING ]
  so-elasticsearch ------------------------------------------------------ [ OK ]
  so-filebeat --------------------------------------------------------- [ STARTING ]
  so-fleet ------------------------------------------------------------ [ STARTING ]
  so-grafana ------------------------------------------------------------ [ OK ]
  so-idstools ----------------------------------------------------------- [ OK ]
  so-influxdb ----------------------------------------------------------- [ OK ]
  so-kibana ------------------------------------------------------------- [ OK ]
  so-kratos ------------------------------------------------------------- [ OK ]
  so-mysql -------------------------------------------------------------- [ OK ]
  so-nginx -------------------------------------------------------------- [ OK ]
  so-playbook --------------------------------------------------------- [ STARTING ]
  so-redis ------------------------------------------------------------ [ STARTING ]
  so-sensoroni ---------------------------------------------------------- [ OK ]
  so-soc ---------------------------------------------------------------- [ OK ]
  so-soctopus --------------------------------------------------------- [ STARTING ]
  so-steno ------------------------------------------------------------ [ STARTING ]
  so-strelka-backend -------------------------------------------------- [ STARTING ]
  so-strelka-coordinator ---------------------------------------------- [ STARTING ]
  so-strelka-filestream ----------------------------------------------- [ STARTING ]
```

(b) With the command sudo so-status the status of each tool can be seen to make sure all tools can be accessed.

Figure 28: Display the dashboard and check the status of the tools.

To ensure NIDS Suricata is working in the following directory:

```
cd /etc/nsm/rules/local.rules

```

the following command is entered:

```
Alert icmp 10.1.1.1 any -> any (msg:"ICMP";
sid:100002;)

```

To ensure a thorough experiment Suricata must be carried out in a controlled environment. This project aims to focus examine the overall performance of CPU, memory usage, and packet processing rates.

7 Testing and Results

To ensure Kali can reach the target device a ping test should be done to ensure, Kali's attacks are being received in the first place.

```
ping 10.0.2.20
ping google.com

```

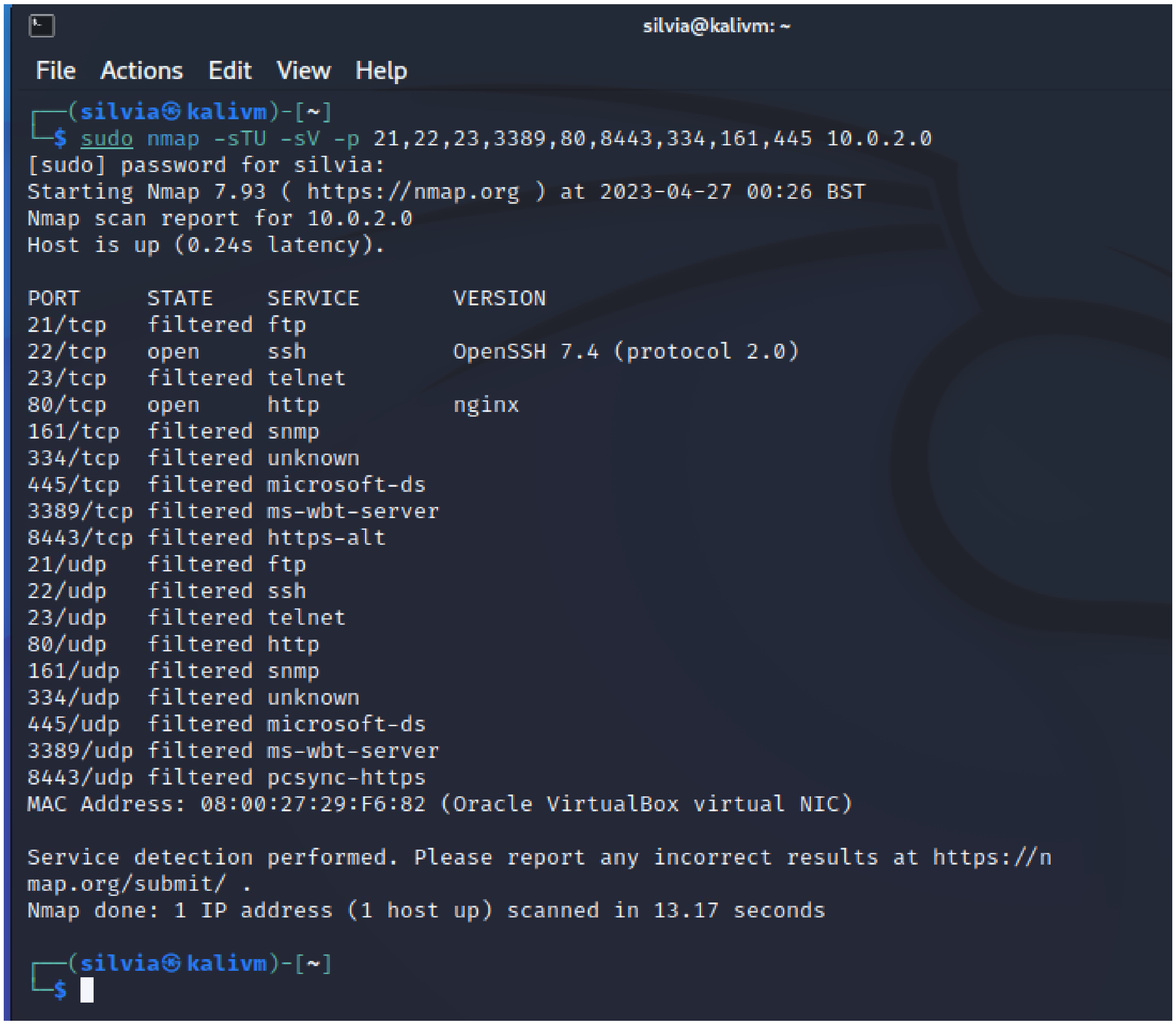
The understanding behind a ping test is that an ICMP request is sent from the Kali virtual machine to check if a connection exists, if this is the case the Security Onion will echo a reply back. Figure 28b shows Kali has a connection established to both Security Onion and Google.com

7.1 Sending attacks

Starting with Nmap Port-scanning, the purpose of this type of attack is to attain real-time information about the target IP address from whether the host is Up (or DOWN) to information on which ports are open at the current time, as it can be seen from the figure below this valuable information for a malicious user to send tailored attacks.

```
    sudo nmap -sTU -sV -p x #.#.#.#

```

The format of this threat is as follows: To understand this low-level threat, "sudo" is necessary as root permission is required to run the command, moving on, the "nmap" command is used to scan whether the host is up and which ports are open for the IP provided. As for "-sTU" the s stands for scanning both "T" and "U" where "T" refers to the TCP scan type. This is where Nmap will either do a TCP Sync scan or a TCP connect scan. The type of scan that will be done will be dependent on the user's privileges however by default it is usually a TCP SYN scan. The following switch -sV determines the service version running on that port. The -p refers to the x as the port numbers being scanned. And lastly, the #'s represent the target IP addresses.



```
┌──(silvia㉿kalivm)-[~]
└─$ sudo nmap -sTU -sV -p 21,22,23,3389,80,8443,334,161,445 10.0.2.0
[sudo] password for silvia:
Starting Nmap 7.93 ( https://nmap.org ) at 2023-04-27 00:26 BST
Nmap scan report for 10.0.2.0
Host is up (0.24s latency).

PORT      STATE    SERVICE        VERSION
21/tcp    filtered ftp
22/tcp    open     ssh            OpenSSH 7.4 (protocol 2.0)
23/tcp    filtered telnet
80/tcp    open     http           nginx
161/tcp   filtered snmp
334/tcp   filtered unknown
445/tcp   filtered microsoft-ds
3389/tcp filtered ms-wbt-server
8443/tcp filtered https-alt
21/udp    filtered ftp
22/udp    filtered ssh
23/udp    filtered telnet
80/udp    filtered http
161/udp   filtered snmp
334/udp   filtered unknown
445/udp   filtered microsoft-ds
3389/udp filtered ms-wbt-server
8443/udp filtered pcsync-https
MAC Address: 08:00:27:29:F6:82 (Oracle VirtualBox virtual NIC)

Service detection performed. Please report any incorrect results at https://n
map.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 13.17 seconds

┌──(silvia㉿kalivm)-[~]
└─$ 
```

Figure 29: Portscanninng Security Onion take at 00:27. Results can be viewed in section 30

Figure 29 can be used to conclude that Kali's system was able to successfully establish communication with the target device. This is inferred from no error messages being displayed and that the host is up and is scanned in two ports through TCP (22/TCP and 80/tcp).

To view the impact on the overall environment Grafana has been used. The guide to using Grafana will be given in the User Guide section. 9An important note be made here is that Nmap can be used for both ethical and malicious reasons, where one can use it to find vulnerabilities to prevent and protect within their systems, and others may misuse Nmap to find weaknesses including but not restricted to identifying weaknesses such as open ports and vulnerable hosts.

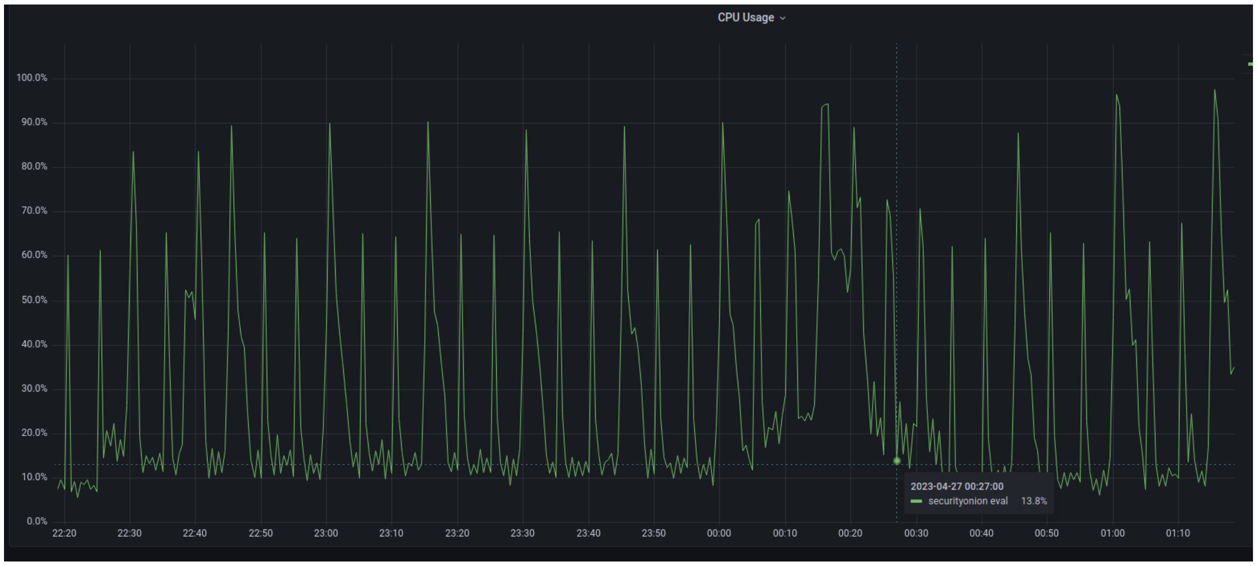


Figure 30: CPU usage when Port-scanning Security Onion take at 00:27 is at 13.8%

It is evident that the system is not using up all of its resources in this "attack" as this is not a DoS or a DDoS attack, low CPU could be due to point towards many reasons. The most likely reasoning is that Kali's system in this case is only scanning 9 specified ports therefore not much computer resource is used causing the overall workload to be lighter. Other possible reasons include the scanning technique being efficient enough to minimise the overall CPU load through distributing the workload to multiple threads which is a similar concept to Suricata infrastructure in figure 6, to further understand this Nmap scan types can be researched into.

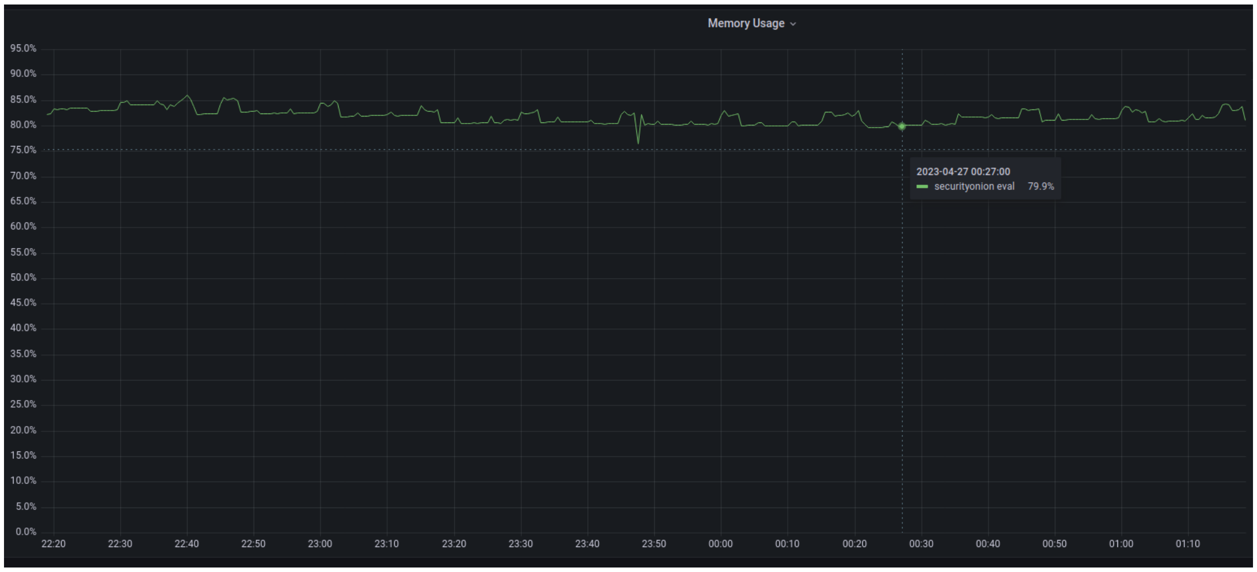


Figure 31: Memory usage when Portscanninng Security Onion take at 00:27 is at 79.9%

Memory usage being 79.9% is quite high, this could be due to the type of scanning the Nmap is carrying out (TCP and UDP scans) but this cannot be the only contributing factor, it could be Security Onion's role in using the systems resources could be impacting the overall memory usage health.

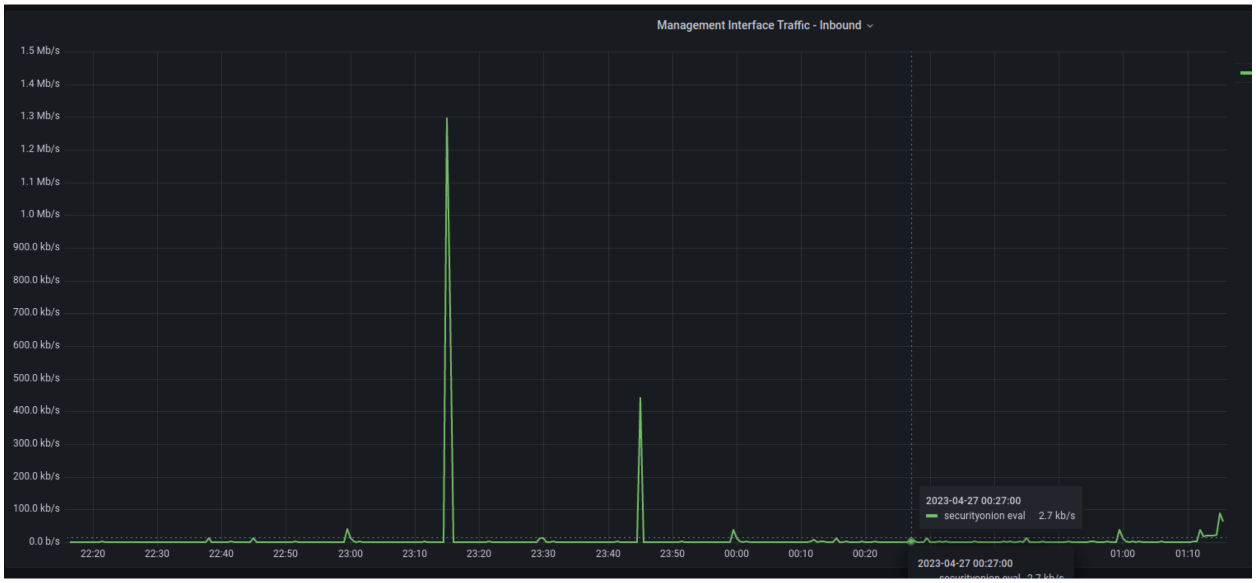
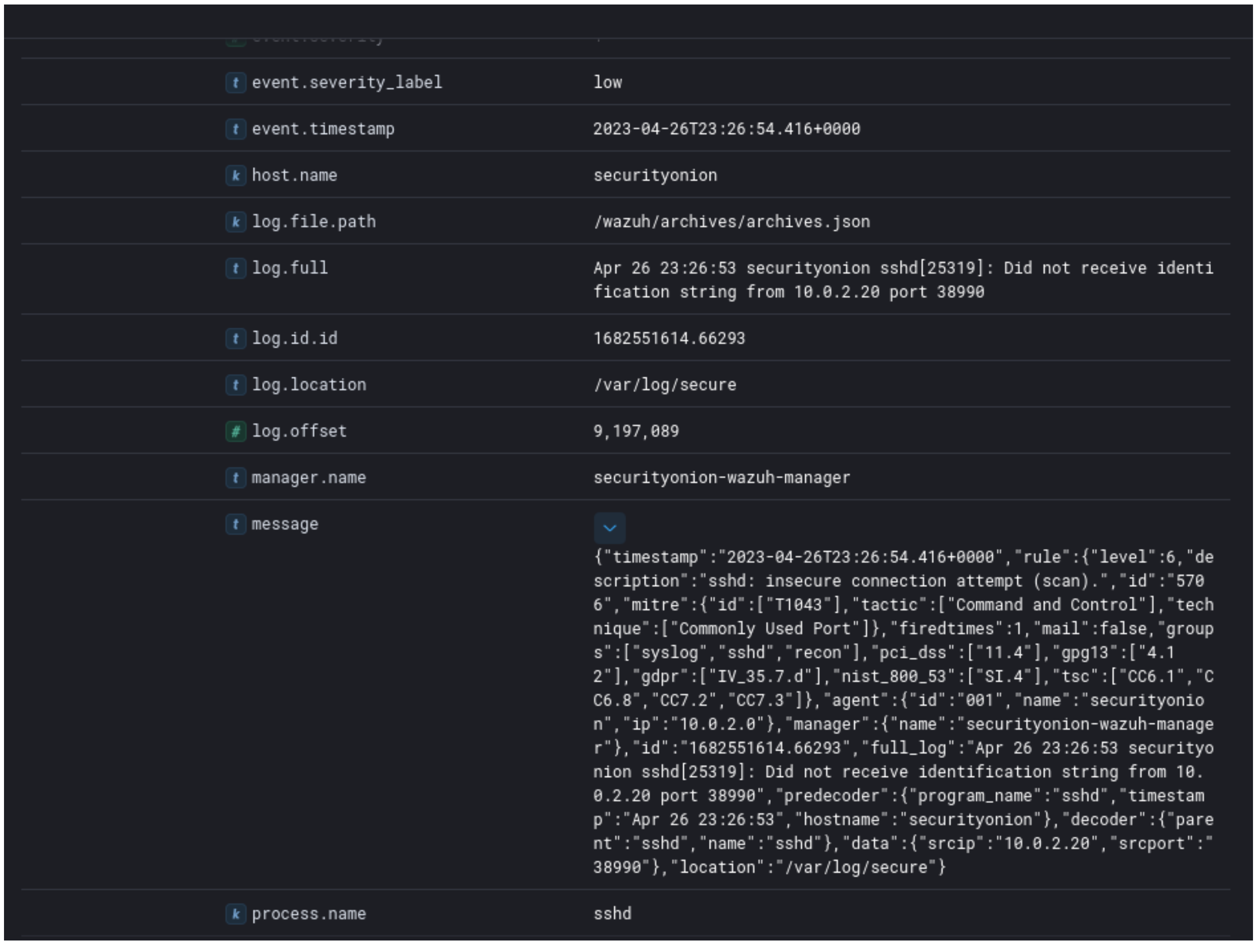


Figure 32: The inbound management interface of the network traffic when Portscanninng Security Onion take at 00:27 is at around 2.7 kb/s

When it comes to Nmap scanning the packet receiving rate seems to be at 2.7 kb/s, the inbound rate being so low can as stated earlier due to a small range of ports being scanned therefore not impacting network traffic as much.



```
event.severity_label     low
event.timestamp          2023-04-26T23:26:54.416+0000
host.name                securityonion
log.file.path            /wazuh/archives/archives.json
log.full                 Apr 26 23:26:53 securityonion sshd[25319]: Did not receive identi
                         fication string from 10.0.2.20 port 38990
log.id.id                1682551614.66293
log.location             /var/log/secure
log.offset               9,197,089
manager.name             securityonion-wazuh-manager
message
```

```
{"timestamp":"2023-04-26T23:26:54.416+0000","rule":{"level":6,"de
scription":"sshd: insecure connection attempt (scan).","id":"570
6","mitre":{"id":["T1043"],"tactic":["Command and Control"],"tech
nique":["Commonly Used Port"]},"firedtimes":1,"mail":false,"group
s":["syslog","sshd","recon"],"pci_dss":["11.4"],"gpg13":["4.1
2"],"gdpr":["IV_35.7.d"],"nist_800_53":["SI.4"],"tsc":["CC6.1","C
C6.8","CC7.2","CC7.3"]},"agent":{"id":"001","name":"securityonio
n","ip":"10.0.2.0"},"manager":{"name":"securityonion-wazuh-manage
r"},"id":"1682551614.66293","full_log":"Apr 26 23:26:53 securityo
nion sshd[25319]: Did not receive identification string from 10.
0.2.20 port 38990","predecoder":{"program_name":"sshd","timestam
p":"Apr 26 23:26:53","hostname":"securityonion"},"decoder":{"pare
nt":"sshd","name":"sshd"},"data":{"srcip":"10.0.2.20","srcport":"
38990"},"location":"/var/log/secure"}
```

```
process.name             sshd
```

Figure 33: Portscanninng Security Onion take at 00:27

Once the threat is successfully sent from Kali's environment, on the Security Onions console (otherwise known as dashboard) an alert flags up the above message in JSON format. This JSON format attack can be as evidence for the threat in the Security Onion's console. To understand the message from the target address perspective, who has no knowledge of where this attack is coming from only key details will be extracted:

- Timestamp - This conveys when the event of attacks occurred however there is a slight time discrepancy but to briefly speak Kali's environment timing does not match with of Security Onion.

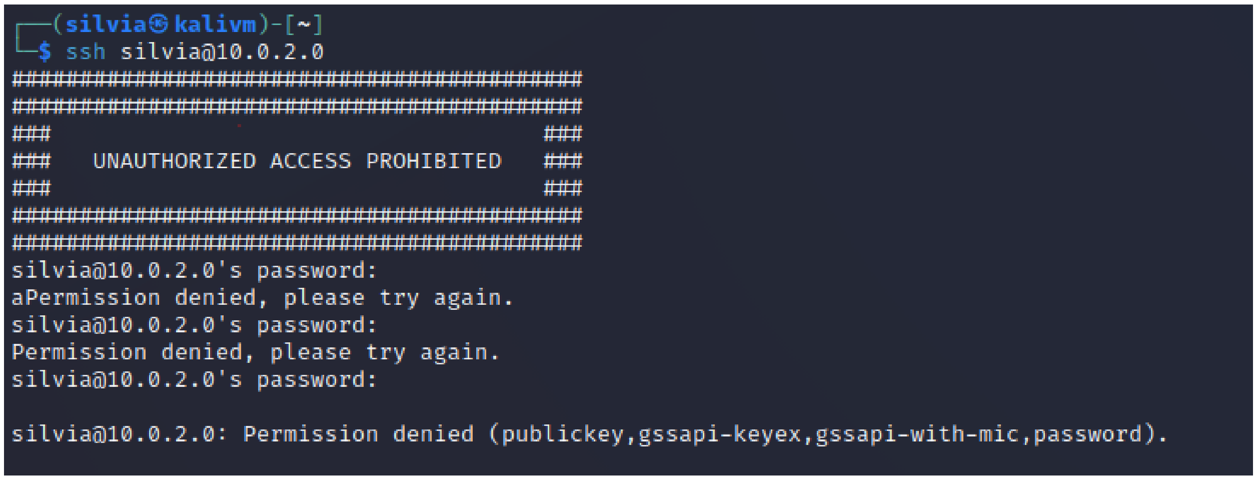
- Description - which is a part of the rule: "6" the number is an indicator of the category the rule falls under in this case the 6 represents the category which this detection fall under, as for the description "sshd - insecure connection attempt (scan)" suggests what kind of detected activity caused the alert to be generated.
- MITRE Techniques - which falls under ID "T1043" Points at the technique that is being used by the source IP address.
- Compliance Frameworks - "pci_dss", "gpg13", "gdpr" and "nist_800_53" These frameworks are all relevant to the threat received as they fall under different recognised standards and regulations.
- Fired Times- refers to the number of times the dashboard flagged this threat.
- Source IP and Port - The Source IP and port detected (10.0.2.20 and 38990).
- Full Log - refers to additional details the target IP address might find useful.

These details can give the user of the targetted IP address a basic understanding of the attack, specifically the activity that has been detected the attack for, techniques used, the compliance frameworks involved, and lastly any additional details on the source IP and port.

7.1.1 Brute Forcing SSH login

To access a secure communication remotely a malicious user might attempt to infiltrate the system through brute forcing SHH login. This is whereby the attacker tries different combinations of usernames and passwords the victim uses to log into their device. The following command shows how to do so:

```
    ssh userid@IPaddress

```



```
┌──(silvia㉿kalivm)-[~]
└─$ ssh silvia@10.0.2.0
#########################################
#########################################
###                                   ###
###   UNAUTHORIZED ACCESS PROHIBITED   ###
###                                   ###
#########################################
#########################################
silvia@10.0.2.0's password:
aPermission denied, please try again.
silvia@10.0.2.0's password:
Permission denied, please try again.
silvia@10.0.2.0's password:

silvia@10.0.2.0: Permission denied (publickey,gssapi-keyex,gssapi-with-mic,password).
```

Figure 34: Trying multiple passwords to ssh login

In figure 34 the denied permission after every few attempts, has been put in place to delay the brute forcing so that the host system can process the detected activity.

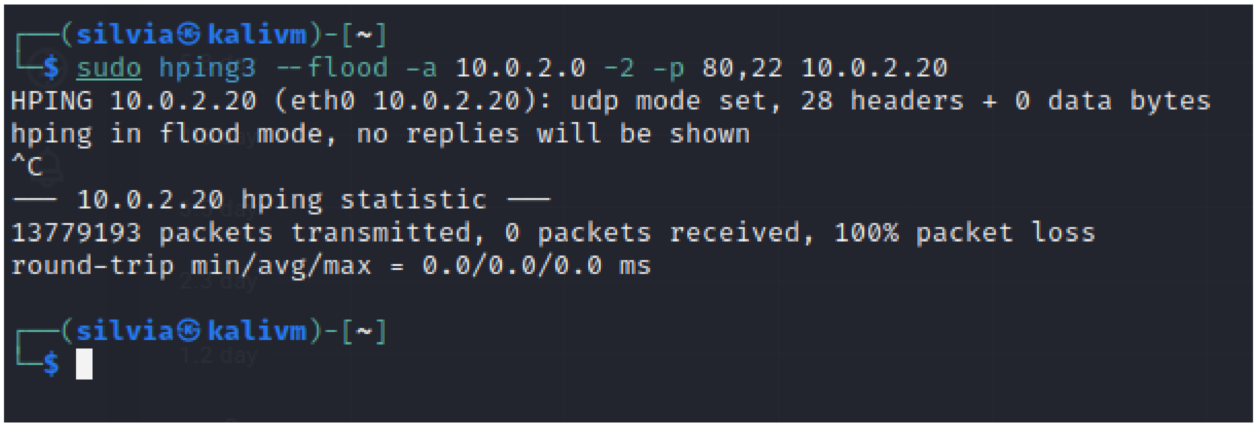
In this scenario, the malicious user might use port scanning to extract details about the target IP. Combining port scanning with a brute-forcing tool to guess the password in the case that attacker already knows the username used, gives the attacker a higher chance of gaining unauthorised access. Brute forcing refers to running a script that runs all possible combinations of for example digits or special characters to enter for a password, this is done instead of having to enter manually one by one, this also reduces human error specifically the likelihood of a password combination being missed by chance. To prevent this a user can enforce a strong password policy and use a combination of digits and special characters that makes a password that cannot be linked back to the user behind the target IP address. In addition to this, the Security Onions network intrusion detection system would log this as an alert, as well as deny more than a few attempts each time to log in to limit and delay the multiple attempts to a single target IP.

7.1.2 UDP attack

The concept behind UDP flood attacks has been explained in section 3.1. But in brief, the attacks overwhelm the system by using up its resources.

```
    sudo hping3 --floood -a #.#.#.# -2 -p x #.#.#.#

```

The format for UDP attacks is as follows: To understand this low-level threat, "sudo" is necessary as root permission is required to run the command, moving on,hping3 refers to the command tool being used to test the network. The next part of the threat, –flood is an option that comes with hpin3 that causes it to send an immense quantity of packets very fast, moving on to the source IP address the "-a" point at the IP address Kali will flood the Security Onion's system with. The "-2" refers to the UDP option being specified as the transport protocol, the -p is indicating the port that is being entered so in this case the two ports being used are HTTP and SSH. The last set of IP addresses is the target IP address. Figure 35 below shows a UDP attack being sent successfully.



```
┌──(silvia㉿kalivm)-[~]
└─$ sudo hping3 --flood -a 10.0.2.0 -2 -p 80,22 10.0.2.20
HPING 10.0.2.20 (eth0 10.0.2.20): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.2.20 hping statistic ---
13779193 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(silvia㉿kalivm)-[~]
└─$ 
```

Figure 35: Portscanninng Security Onion take at 18.34

Figure 35 below shows a UDP attack being sent successfully, where a total of 13779193 packets have been sent within the span of approximately 45 seconds.

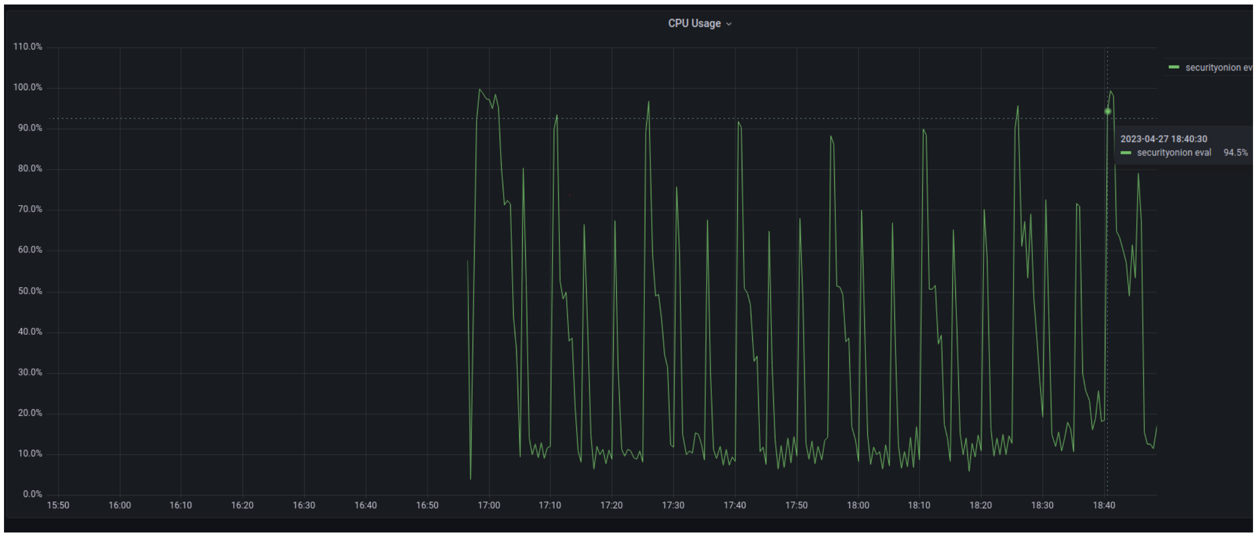


Figure 36: Security Onion take at 18.40

UDP is a transport layer protocol that requires reliable and fast transmissions of packets; hence why it is used for video and voice transmission where all packets are guaranteed to be received. As can be seen by the figure above the CPU is at its highest, the most likely reasoning is due to the amount of packets sent for approximately 45 seconds. As this is an amplification attack it is very CPU intensive.

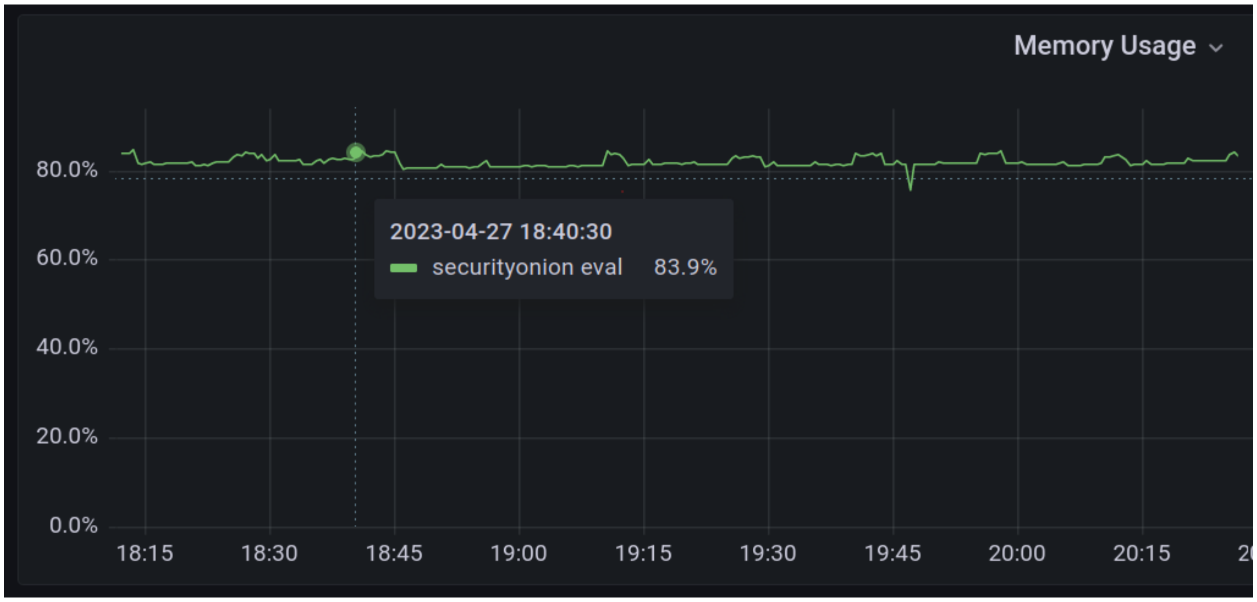
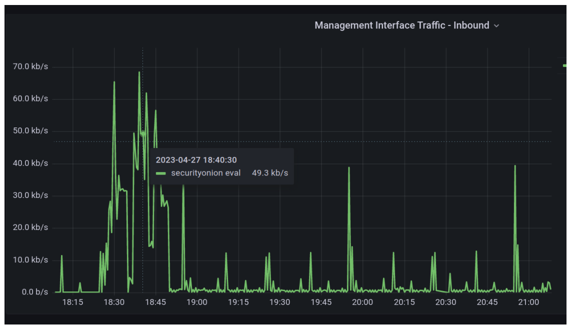
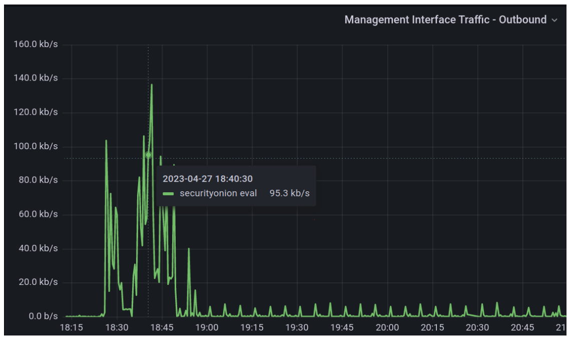


Figure 37: UDP flooding impact on Security Onion's memory usage at around 89.3% at 18:40:30

Memory usage is quite high when UDP attacks are sent to the Security Onion. Troubleshooting this accurately requires heavy monitoring of a combination of resources and expertise. High memory usage could be due to memory leaks that have not been released from the memory it has been allocated to, therefore using up more memory than expected. To understand the cause of this better, tools such as SharkTap can be used to assess what is using up memory resources and therefore causing the percentage to be high. It should be noted that the memory usage when port-scanning was also, this may indicate a pattern of tasks taking up memory resources unnoticed rather than a being cause of the attacks.



(a) UDP flooding impact on Security Onion's network inbound usage at around 49.3 kb/s at 18:40:30



(b) UDP flooding impact on Security Onion's network outbound at around 95.3 kb/s at 18:40:30

Figure 38: UDP attack's impact on the inbound and outbound network traffic.

The inbound traffic graph refers to the amount of data that is required to reply to the attacker's environment (Kali Linux), as for the outbound network traffic graph displays how much data is being sent to the attacker's IP source.

The inbound network traffic being 49.3 kb/s suggests a substantial amount of data is required to be processed, but when compared with the outbound network traffic of 95.3 kb/s, it is a quite good indication that the system handling responding to UDP packets quite well, as there is nearly half of the total packet that requires to be dealt, to put it more simply, Security Onion can send more processed data out than it is receiving.

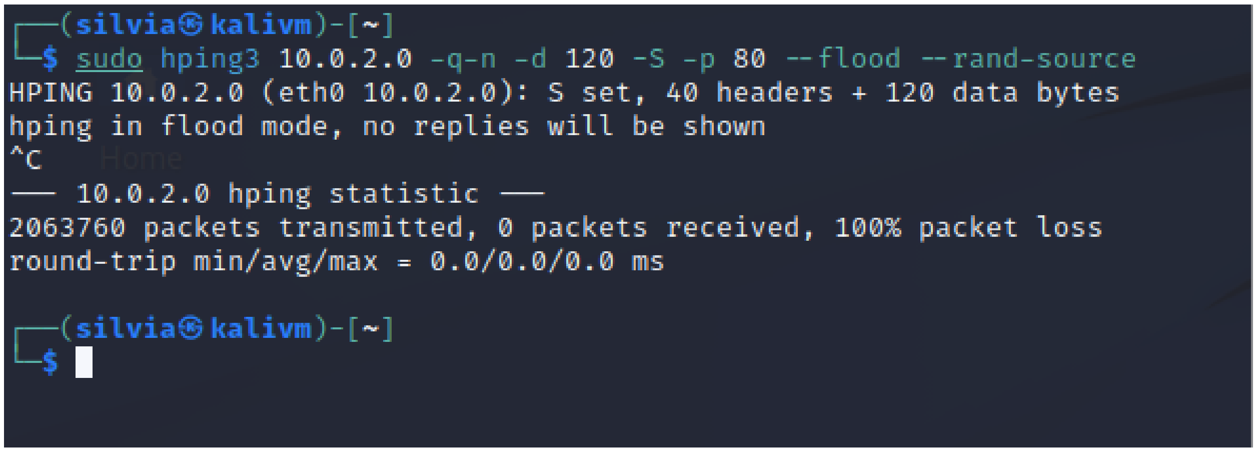
Taking preventative measures is still necessary to protect the system's integrity, a solution to this can be to immediately terminate the connection between the system and network.

7.1.3 TCP SYN Flood Attack

The penultimate attack that will be tested on Security, occurs when the attacker initiates a connection but never responds, creating a backlog of unanswered messages in Security Onion's system therefore using up its resources.SYN packets are what are used to establish connections.

```
    sudo hping3 #.#.#.# -q -n -d 120 -S -p 80 --flood --rand-source

```

The format for TCP SYN flood attacks is as follows: To understand this low-level threat, "sudo" is necessary as root permission is required to run the command, moving on, hping3 refers to the command tool being used to test the network. The hashes represent the target IP address, and "-q" stands for quiet which suppresses the amount being displayed. The -n is a numeric value used to disable the DNS server, by doing so it is ensure that the hping3 command uses the target IP without trying to mitigate unresolved hostnames. The next part involves "-d 120" which refers to the data size per each packet to be a total of 120 bytes filled with randomise data. Moving on to the "-S" set the packets to indicate that they TCP SYN packets, the "-p" is indicating the port that is being entered so in this case the port being used is HTTP. The –flood is an option that comes with hpin3 that causes to send packets of data continuously, and lastly –rand-source refers to spoofing the source IP address to make it harder to identify the attacker's address.



```
┌──(silvia㉿kalivm)-[~]
└─$ sudo hping3 10.0.2.0 -q-n -d 120 -S -p 80 --flood --rand-source
HPING 10.0.2.0 (eth0 10.0.2.0): S set, 40 headers + 120 data bytes
hping in flood mode, no replies will be shown
^C
--- 10.0.2.0 hping statistic ---
2063760 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(silvia㉿kalivm)-[~]
└─$ 
```

Figure 39: TCP SYN flooding impact on Security Onion's target IP address 21:09

After successfully sending this attack it should be noted that there was a two-minute delay right after in Kali's system this could be due to Kali not finishing sending the attack, and it further skewed the results as the JSON logs do not specify the activity detected are related to the attack. To identify this attack approximate timestamps were used as the attack was not flagged in the alert section.

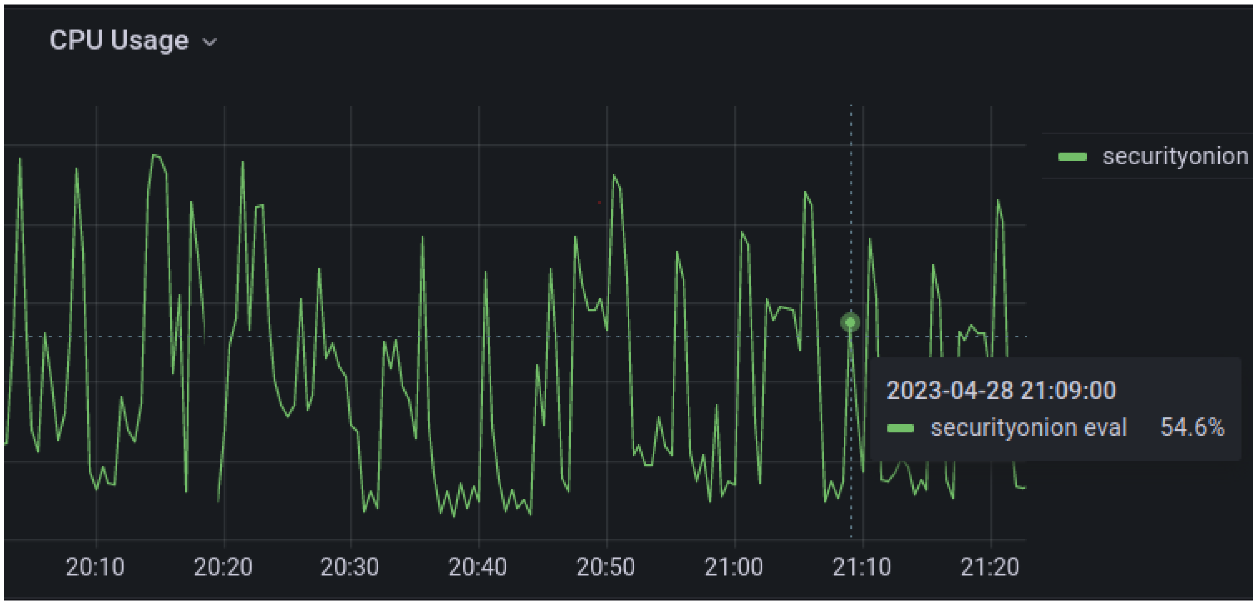
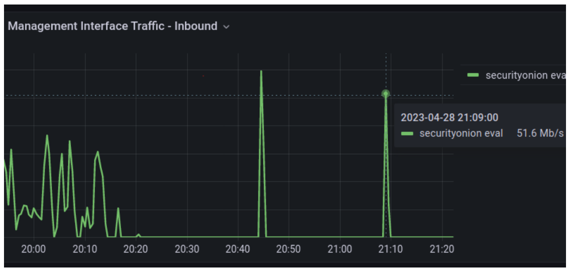
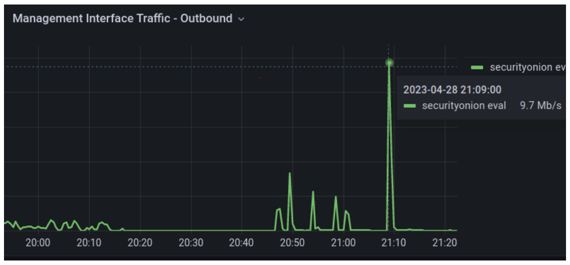


Figure 40: TCP SYN flooding impact on Security Onion's CPU usage at around 54.6% at 21:09

As it can be seen the attack seems to have little impact on CPU usage, there could be many reasoning however it should be noted moments after this the next peak in the graph is much higher, one possible reason for this is network latency, as although a substantial amount of packets are being sent they are being sent a few minutes later than they should. The cause for this may be the packet size of 120 bytes impacting the transportation protocol layer.



(a) TCP SYN flooding impact on Security Onion's network inbound usage at around 51.6 Mb/s at 21:09



(b) TCP SYN flooding impact on Security Onion's network inbound usage at around 9.7 Mb/s at 21:09

Figure 41: UDP attack's impact on the inbound and outbound network traffic.

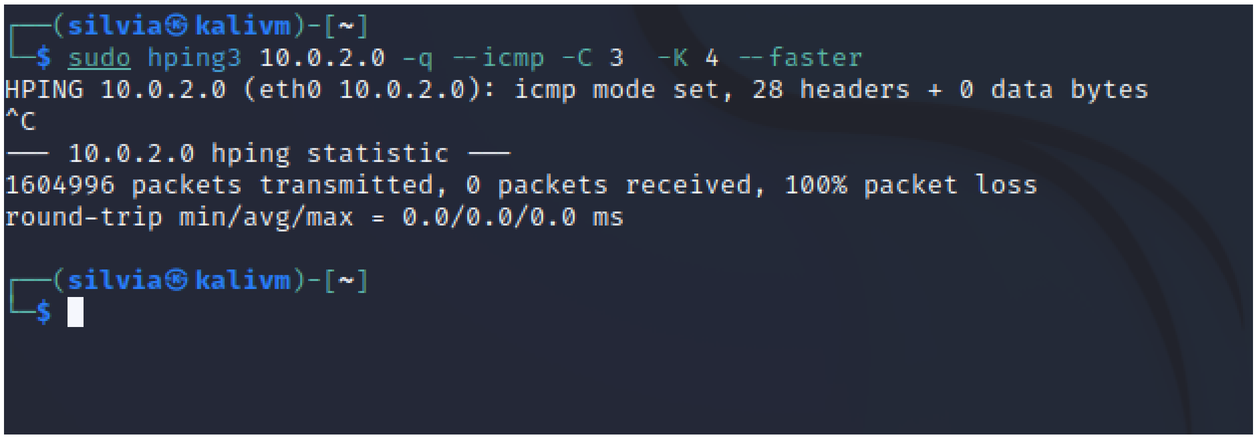
This attack processed greater quantities of both inbound and outbound network traffic compared to the previous attack which can be referred back to 3.1. Comparing inbound and outbound suggests that more data is being received to be processed than packets of data that have already been processed and are ready to be sent back. If this situation escalates Security Onion's system can potentially crash. A way to prevent this would be to implement Intrusion Prevention System (IPS), which is used to not only detect malicious activity but also prevent attacks from occurring.

7.1.4 ICMP attack

The last attack involves masking as the target address to echo to a network of IP addresses which echo back to the real target IP address flooding the system with replies, this attack has been covered in more depth in section 3.1

```
    sudo hping3 #.#.#.# -q --ICMP -C 3 -K 4 --faster

```

The format for UDP attacks is as follows: To understand this low-level threat, "sudo" is necessary as root permission is required to run the command, moving on, hping3 refers to the command tool being used to test the network. The next part of the threat, "-q" stands for quiet which suppresses the output that is being displayed. The "–ICMP" is an option that comes with hping3 that causes specifically ICMP packets to be sent. "C 3" refers to the number of echo request packets that are to be sent which means 3 echoed replies will be sent to the target IP. Moving onto "-K 4 ", this simply acts as an identifier. And lastly, –faster is a command for a faster process preventing any possibilities of delay.



```
┌──(silvia㉿kalivm)-[~]
└─$ sudo hping3 10.0.2.0 -q --icmp -C 3  -K 4 --faster
HPING 10.0.2.0 (eth0 10.0.2.0): icmp mode set, 28 headers + 0 data bytes
^C
--- 10.0.2.0 hping statistic ---
1604996 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──(silvia㉿kalivm)-[~]
└─$ 
```

Figure 42: ICMP attack's impact on Security Onion at 00:45:33 lasting around 2 minutes.

For this attack, the focus has been on the dashboard's result. Whilst understanding the health of the system in the precious attacks, the detection performance on the dashboard had been slightly overlooked. The result of this attack did not display any indication of the source address, the only way to identify the alert from the attack was by matching the timestamps, on the JSON log system the following two messages were displayed:

```
  "note":"CaptureLoss::Too_Little_Traffic",
  "msg":"Only observed 0 TCP ACKs and was expecting at least
1."
  4
```

Although other attacks were timed to be at 45 seconds, due to the message suggesting the threat did not occur for long enough, this attack was carried out a second time for a lengthy period of two minutes, the log for this was the same, which based on the following log :

```
  "actions":["Notice::ACTION\_LOG"],"email\_dest":[],"
suppress\_for":3600.0}
```

suggested the engine detecting this threat Zeek might have been configured to suppress this form of alert in its rule file.

[4]The note and the message

8 Discussion and Evaluation

8.1 Objectives

The following objectives were proposed at the start of the project:

- To study different forms of network threats. This has been done in the literature review section of this report beginning from section 3.1.
- To design and analyse the implementation of a simulated network. This objective has been changed for a more realistic experiment, the initial idea entailed using software to send attacks to a cluster of devices. Instead, virtual machines are being used where configurations can be manipulated better.
- To implement Security Onion as a proxy, this objective has also been changed to fit a more realistic experience whereby Security Onion's virtual machine is used as a target address and there was no need to fit it as a proxy due to the environment already having multiple NIDS embedded in the system.
- To examine Security Onion implemented in the simulated network. This objective has been completed, it starting from section 7.
- To evaluate how Suricata recognises network threats, although the understanding of Suricata's role has been refined and explored throughout the report, it is a combination of Suricata Snort and Zeek that are embedded within the system however only one can be used at a time, according to the documentation one NIDS has to be disabled before another can be enabled. The focus of this objective has shifted towards evaluating how effective Security Onion is at detecting unusual activities.
- To conclude on the effectiveness of Security Network to detect network intrusions. This objective has been achieved successfully.

8.2 Amended issues

This section focuses on issues that have been encountered during the completion of this project and have been rationalised.

8.2.1 Time discrepancy

An important note to be made in this report is that there is a one-hour time discrepancy. Where the Kali Linux is set to Europe/London time, the Security Onion is set one hour back from that. This can be easily fixed through the date command by doing sudo timedatectl set-timezone YOUR_COUNTRY. The Security Onion Documentation, however, advises not to change this one-hour setback as this (UTC/GMT)timing can be used to correlate different cybersecurity events within systems across the world and therefore not changing this time zone is considered as best practice within the industry.

8.2.2 Sudo Crash

Users who may be using the file attached to this document must be aware of sudo crashing, this can be due to a long period of inactivity creating a backlog of necessary updates, while executing this project, this issue was faced and the only viable solution was to re-install the Security Onion's environment as going into recovery mode to repair the sudo command was not an option. Therefore consistently updating both environments is necessary to avoid such problems.

8.3 False Positives

A lot of alerts in the dashboard consisted of false positives such as NAT detections, which can lead to unwanted skews in graphs causing other attacks to go unnoticed, to suppress false positives according to the Security Onions's documentation the rules can be tuned. Numbers are used as an indication of the severity level used by the engines Suricata, Wazuh, and Playbook. The severity is categorised as follows:

- event.severity: 4 ==> event.severity_label: critical
- event.severity: 3 ==> event.severity_label: high
- event.severity: 2 ==> event.severity_label: medium
- event.severity: 1 ==> event.severity_label: low

For instance, Wazuh, a host-based intrusion detection system responsible for detecting unusual activities giving a lot of false positives can be modified by including a noalert="1" in the rule section.

8.4 Further Development

There are alternative methods that are far more ideal compared to portscan, instead of using Nmap, using the GUI version called Zenmap would mean that a wider range of ports would be port-scanned instead of a few randomly selected port numbers.

To further understand the very powerful engines embedded within Security Onion one can compare their detection algorithms in a controlled environment, this could have been done in this project however due to time constraints it is not possible, this is because Security Onion requires enabling and disabling each time the preferred engine is changed and for multiple attacks with multiple trials this was not an option.

To further explore this in this project only low-level attacks have been carried out due to insufficient expertise in this field, this research can be further pushed by carrying out higher-level attacks to imitate real threats better.

8.5 Conclusion

To conclude this report, Security Onion is an effective and easy-to-use NIDS solution that offers several unique advantages over other systems. Its simplicity and ease of use make it an attractive option for organisations looking to improve their network security. Its scalability and its powerful security tools make it an effective tool for detecting and responding to network intrusions. However, this framework requires much more research and testing to ensure it is as mentioned in the initially proposed concern; whether Security Onion is an effective distribution.

All research done so far on Security Onion are experiments implemented using an older version of Security Onion. The experiments have carried an updated version of Security Onion, the latest version being 2.3.200, however to ensure a stable version has been used in this project. The version suggested at the front of the Security Onion's website has been used instead. Using an updated distribution makes this a fresh research with potentially newer and higher results in terms of accuracy. More research involving a broader spectrum of attacks is required to conclude, as this project mainly focused on DoS/DDos-based attacks.

An overall theme to be noted is where a low-level flooding-based attack gets little relevance in the dashboard alert, in the same manner, short burst low-level threats get flagged under alerts suggesting that Security Onion's default configuration focuses on efficiency as logging packets sent every millisecond would greatly impact the system's memory instead it records how many times the attack has been fired within that short window of time. To better this every user deems efficiency differently, not much reconfiguration is required to use SO, therefore, it may be suitable for users who are not experienced in using Security Onion as their Operating system. The next steps in the research implementation of this project involve using real severe attacks in controlled settings to understand where

Security Onion's limitations fall.

9 User Guide

This section is going to be a user-friendly guide to allow individuals with access to the .vdi files to carry out the same attacks to test security onion in their environment.

To commence, there will be two .vdi files that will be attached along with this documentation. One of the fie is to run the security onion environment that has been ready set up and the latter is for Kali. If Virtual Box has not already been installed, the user would simply need to:

- Set the name and folder path for the virtual environment.
- Choose the type of Linux distribution version that is going to be used.
- Make decisions on storage and CPU based on their machine's capability.

After doing so, within the VirtualBox (VBox) manager interface such as in figure 14a which displays all the machines running, under storage there should be an option to create or upload personal disk files. Here the user should choose their own.

This should enable them to run both environments without setting up any of the initial steps in deploying virtual machines.

A few important things required to be noted when running these machines are listed below:

- Security Onion's IP address (Target IP)- 10.0.2.0
- Kali Linux's IP address (Source IP)- 10.0.2.20
- The username for both environments- silvia
- The password for both environments silvia123
- The username for Security Onions Console that is accessed through Kali Linux's browser- SA3046@live.mdx.ac.uk
- The password for security onion's console- silvia123
- The username to Grafana should be as stated by the documentation- admin
- The password for Grafana is- 4bav4b5thBn4Qk2kJiWm

To access Security Onion's dashboard within the Kali Linux environment, a command would be required to be run if it's the first time and will be required every once in a while, the user would be required to allow the IP range to be allowed to view the analytics. To do this the user would run the following command as the root.

```
sudo so-allow
```

the user then should select the purpose, in this project a) analytics was chosen, and sequentially the user will be prompted the IP address to allow viewing the dashboard (In this case it should be Kali's). If sudo does not work it might be that Security Onion's configurations are not updated. There are cases in which going into recovery mode does not resolve the issue, if this is the case the user needs to redeploy the security onion from the start. Therefore it is suggested the user looks out for necessary updates. The Security Onion .vdi file attached is just a command prompt hence why it is necessary to be able to have access to the dashboard and this can be done so by accessing Linux's environment. This step is done to ensure the dashboard is ready to be viewed, the following command should display an [ok] on each container similar to figure 28b :

```
sudo so-status
```

When all containers are "OK", the user can then proceed to open any browser in Kali and enter Security Onion's IP address. If prompted to enter login details the user should enter the email and password entered when configuring Security Onion, an example of doing this can be viewed in figure 27b.

To make a health check on the Security Onion's environment, the user can Grafana which has been made available on the dashboard however the username and password are different. According to the documentation the username is set to "admin", and as for the password can be accessed through the following command:

```
sudo salt-call pillar.get secrets
```

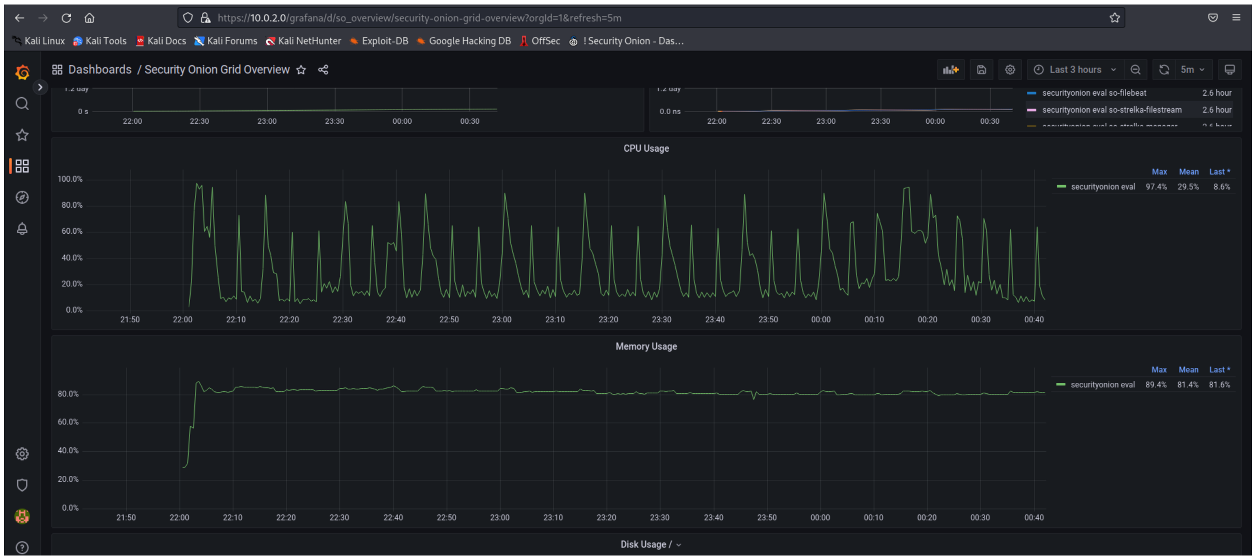

https://10.0.2.0/grafana/d/so_overview/security-onion-grid-overview?orgId=1&refresh=5m
Dashboards / Security Onion Grid Overview
Last 3 hours
5m
CPU Usage
Max Mean Last *
securityonion eval 97.4% 29.5% 8.6%
Memory Usage
securityonion eval 89.4% 81.4% 81.6%
securityonion eval so-filebeat 2.6 hour
securityonion eval so-strelka-filestream 2.6 hour
Disk Usage /

Figure 43: Security Onion's health overview viewed through Grafana.

If for any reason the command above cannot be used to retrieve the password for Grafana's health overview dashboard, the user can use htop to view the CPU from the command line in the case that the command line Security Onion is being used rather than GUI.

10 References

[1] Maddy Ell and Robbie Gallucci. Cyber security breaches survey 2022, Jul 2022.

[2] Mohammed Mazumder and Abdus Sobhan. The spillover effect of the bangladesh bank cyber heist on banks' cyber risk disclosures in bangladesh. *Journal of Operational Risk*, 15(4), 2020.

[3] Brian Cashell, William D. Jackson, Mark Jickling, and Baird Webel. Infosec isr congress, 04 2001.

[4] Abdullahi Mikail and Bernardi Pranggono. Securing infrastructure-as-a-service public clouds using security onion. *Applied System Innovation*, 2:6, 03 2019.

[5] Hanan Hindy, David Brosset, Ethan Bayne, Amar Kumar Seeam, Christos Tachtatzis, Robert Atkinson, and Xavier Bellekens. A taxonomy of network threats and the effect of current datasets on intrusion detection systems. *IEEE Access*, 8:104650–104675, 2020.

[6] Krushang Sonar and Hardik Upadhyay. A survey: Ddos attack on internet of things. *International Journal of Engineering Research and Development*, 10(11):58–63, 2014.

[7] Mohammed Alenezi and Martin J Reed. Methodologies for detecting dos/ddos attacks against network servers. In *The Seventh International Conference on Systems and Networks Communications ICSNC*, pages 92–98, 2012.

[8] Sanjeev Kumar. Smurf-based distributed denial of service (ddos) attack amplification in internet. In *Second International Conference on Internet Monitoring and Protection (ICIMP 2007)*, pages 25–25. IEEE, 2007.

[9] Kayla Jansen. Testing the security onion. 2018.

[10] Jonathan Lemon. Resisting {SYN} flood {DoS} attacks with a {SYN} cache. In *BSDCon 2002 (BSDCon 2002)*, 2002.

[11] Aarti Singh and Dimple Juneja. Agent based preventive measure for udp flood attack in ddos attacks. *International Journal of Engineering Science and Technology*, 2(8):3405–3411, 2010.

[12] P Ramesh Babu, D Lalitha Bhaskari, and CH Satyanarayana. A comprehensive analysis of spoofing. *International Journal of Advanced Computer Science and Applications*, 1(6), 2010.

[13] Michael Fire, Roy Goldschmidt, and Yuval Elovici. Online social networks: threats and solutions. *IEEE Communications Surveys & Tutorials*, 16(4):2019–2036, 2014.

[14] J. Gómez, C. Gil, N. Padilla, R. Baños, and C. Jiménez. Design of a snort-based hybrid intrusion detection system. *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living*, pages 515–522, 2009.

[15] Roman Fekolkin. Intrusion detection and prevention systems: Overview of snort and suricata, 01 2015.

[16] Jay Beale, Andrew R Baker, and Joel Esler. *Snort : IDS and IPS toolkit*. Syngress, 2007.

[17] Wonhyung Park and Seongjin Ahn. Performance comparison and detection analysis in snort and suricata environment. *Wireless Personal Communications*, 94:241–252, 02 2016.

[18] Ashley Deuble. Detecting and preventing web application attacks with security onion. *SANS Institute*, 4(1):26–33, 2012.

[19] Eugene Albin and Neil C. Rowe. A realistic experimental comparison of the suricata and snort intrusion-detection systems. *2012 26th International Conference on Advanced Information Networking and Applications Workshops*, 03 2012.

11 Appendices

11.1 Brute Forcing SSH login

```
 {"timestamp":"2023-04-23T20:16:59.103+0000","rule":{"level":5,"description":"sshd: authentication failed.","id":"5716","mitre":{"id":["T1110"],"tactic":["Credential Access"],"technique":["Brute Force"]},"firedtimes":3,"mail":false,"groups":["syslog","sshd","authentication_failed"],"pci_dss":["10.2.4","10.2.5"],"gpg13":["7.1"],"gdpr":["IV_35.7.d","IV_32.2"],"hipaa":["164.312.b"],"nist_800_53":["AU.14","AC.7"],"tsc":["CC6.1","CC6.8","CC7.2","CC7.3"]},"agent":{"id":"001","name":"securityonion","ip":"10.0.2.0"},"manager":{"name":"securityonion-wazuh-manager"},"id":"1682281019.25850","full_log":"Apr 23 20:16:59 securityonion sshd[29364]: Failed password for silvia from 10.0.2.20 port 50528 ssh2","predecoder":{"program_name":"sshd","timestamp":"Apr 23 20:16:59","hostname":"securityonion"},"decoder":{"parent":"sshd","name":"sshd"},"data":{"srcip":"10.0.2.20","srcport":"50528","dstuser":"silvia"},"location":"/var/log/secure"}

```

12 Ethics Approval

Middlesex University London

Research Ethics Screening Form for Students

Middlesex University is concerned with protecting the rights, health, safety, dignity, and privacy of its research participants. It is also concerned with protecting the health, safety, rights, and academic freedom of its students and with safeguarding its own reputation for conducting high quality, ethical research.

<u>This Research Ethics Screening Form will enable students to self-assess and determine whether the research requires ethical review and approval via the Middlesex Online Research Ethics (MORE) form before commencing the study. Supervisors must approve this form after consultation with students.</u>

Student Name:	Silvia Aminul	Email: sa3046@live.mdx.ac.uk
Research project title:	Deploying And Examining Security Onion For Network Threat Detection In A Simulated Network Environment.	
Programme of study/module:	BSc Computer Science	
Supervisor Name:	Dr Kamran Ali	Email: K.Ali@mdx.ac.uk

Please answer whether your research/study involves any of the following given below:		
1. HANIMALS or animal parts.	☐ Yes	☒ No
2. MCELL LINES (established and commercially available cells - biological research).	☐ Yes	☒ No
3. HCELL CULTURE (Primary: from animal/human cells- biological research).	☐ Yes	☒ No
4. HCLINICAL Audits or Assessments (e.g. in medical settings).	☐ Yes	☒ No
5. XCONFLICT of INTEREST or lack of IMPARTIALITY. If unsure see "Code of Practice for Research" (Sec 3.5) at: https://unihub.mdx.ac.uk/study/spotlights/types/research-at-middlesex/research-ethics	☐ Yes	☒ No
6. XDATA to be used that is not freely available (e.g. secondary data needing permission for access or use).	☐ Yes	☒ No
7. XDAMAGE (e.g., to precious artefacts or to the environment) or present a significant risk to society).	☐ Yes	☒ No
8. XEXTERNAL ORGANISATION – research carried out within an external organisation or your reseach is commissioned by a government (or government body).	☐ Yes	☒ No
9. MFIELDWORK (e.g biological research, ethnography studies).	☐ Yes	☒ No
10. HGENETICALLTY MODIFIED ORGANISMS (GMOs) (biological research).	☐ Yes	☒ No
11. HGENE THERAPY including DNA sequenced data (biological research).	☐ Yes	☒ No
12. MHUMAN PARTICIPANTS – ANONYMOUS Questionnaires (participants not identified or identifiable).	☐ Yes	☒ No
13. XHUMAN PARTICIPANTS – IDENTIFIABLE (participants are identified or can be identified): survey questionnaire/ INTERVIEWS / focus groups / experiments / observation studies.	☐ Yes	☒ No

14. ^H^HUMAN TISSUE (e.g., human relevant material, e.g., blood, saliva, urine, breast milk, faecal material).	☐ Yes	☒ No
15. ^H^ILLEGAL/HARMFUL activities research (e.g., development of technology intended to be used in an illegal/harmful context or to breach security systems, searching the internet for information on highly sensitive topics such as child and extreme pornography, terrorism, use of the DARK WEB, research harmful to national security).	☐ Yes	☒ No
16. ^X^PERMISSION is required to access premises or research participants.	☐ Yes	☒ No
17. ^X^PERSONAL DATA PROCESSING (Any activity with data that can directly or indirectly identify a living person). For example data gathered from interviews, databases, digital devices such as mobile phones, social media or internet platforms or apps with or without individuals'/owners' knowledge or consent, and/or could lead to individuals/owners being IDENTIFIED or SPECIAL CATEGORY DATA (GDPR[1]) or CRIMINAL OFFENCE DATA. [1]Special category data (GDPR- Art.9): "personal data revealing racial or ethnic origin, political opinions, religious or philosophical beliefs, or trade union membership, and the processing of genetic data, biometric data for the purpose of uniquely identifying a natural person, data concerning health or data concerning a natural person's sex life or sexual orientation".	☐ Yes	☒ No
18. ^X^PUBLIC WORKS DOCTORATES: Evidence of permission is required for use of works/artifacts (that are protected by Intellectual Property (IP) Rights, e.g. copyright, design right) in a doctoral critical commentary when the IP in the work/artifactis jointly prepared/produced or is owned by another body	☐ Yes	☒ No
19. ^H^RISK OF PHYSICAL OR PSYCHOLOGICAL HARM (e.g., TRAVEL to dangerous places in your own country or in a foreign country (see https://www.gov.uk/foreign-travel-advice), research with NGOs/humanitarian groups in conflict/dangerous zones, development of technology/agent/chemical that may be harmful to others, any other foreseeable dangerous risks).	☐ Yes	☒ No
20. ^X^SECURITY CLEARANCE – required for research.	☐ Yes	☒ No
21. ^X^SENSITIVE TOPICS (e.g., anything deeply personal and distressing, taboo, intrusive, stigmatising, sexual in nature, potentially dangerous, etc).	☐ Yes	☒ No

M – Minimal Risk; X – More than Minimal Risk. H – High Risk

If you have answered 'Yes' to ANY of the items in the table, your application REQUIRES ethical review and approval using the MOREform **BEFORE commencing your research**. Please apply for ethical approval using the MOREform (https://moreform.mdx.ac.uk/). Consult your supervisor for guidance. Also see *Middlesex Online Research Ethics* (MyLearning area) *and* www.tiny.cc/mdx-ethics (CS students).

If you have answered 'No' to ALL of the items in the table, your application is Low Risk and you may NOT require ethical review and approval using the MOREform before commencing your research. Your research supervisor will confirm this below.

Student Signature: [illegible] Date: 13/12/2022

To be completed by the supervisor:

Based on the details provided in the self-assesment form, I confirm that:	Insert Y or N
The study is Low Risk and *does not require* ethical review & approval using the MOREform	
The study *requires* ethical review and approval using the MOREform.	

Middlesex University London

Superivsor Signature:…Kamran Ali….. Date:…14/12/22………………