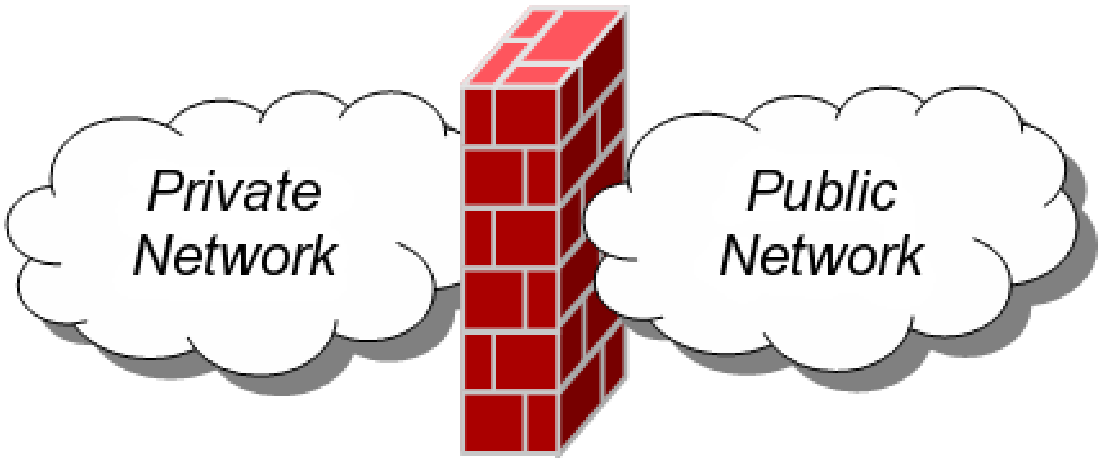
Firewall Report



Group No. 26:

Ahmad Mozan 101080351

Ousama Al-Chami 101077192

Table of contents

Executive Summary

Group 26 decided to choose a project that could be beneficial and one that protects you and your loved ones at home, which is why we chose to build and design a security system that would be free and yet still highly effective. Reason being in our modern society there are many harmful cyber attacks such as sniffing, spoofing and malware, etc. There are many more attacks that are more harmful than the mentioned attacks, but these are the most common ones. Moreover, the work started with the first step which is looking into which virtual machine to have, and the second step which is choosing which open-source firewall to choose from as there are a lot of good ones. The team decided to go with Oracle VM as a virtual machine because we used it throughout our academic years and PFSense as an open-source firewall since it worked well with our design. The third step was to choose which intrusion detection system (IDS) we would use, and the fourth step would be how we were going to test the IDS. The two detection systems we wanted to choose from were Suricata and Snort. Both were solid and effective, so we decided to test both and explore both options. The last step we took was to have a look at the DARPA INTRUSION DETECTION EVALUATION DATASET [1] and their dataset had both PCAP (Packet capture) files and results which we struggled in finding a source that had both. As we dig deeper in the report, we can see the reason behind why Snort (IDS) is the safest and can still protect you from external attacks, as it will be both proven and shown through charts and datasets.

Project

Introduction

The use of firewalls is becoming more common as we progress in the modern society where modern technology is needed, and the safety of these tools and equipment is a requirement. A firewall is a network security protocol that monitors all ingoing and outgoing data (packages). The role of the firewall is to function as a barrier or wall to prevent all unknown traffic such as packages from entering the local or internal network. The unknown traffic must be stopped or otherwise your personal information on your devices would be in jeopardy of being lost or tampered with, or your personal devices would be cyberattacked. A firewall would increase your safety chances and would prevent all these malicious activities. The purpose of having a firewall is to protect us against external packages and detect unknown traffic. Furthermore, the team decided to adopt this project because it will be useful and can be used in our homes. The project will provide more information on how accurate it is against external attacks and traffic, and it will also show the expense of owning or making one. We explored different paths and options to see which ones were more compatible with our requirements.

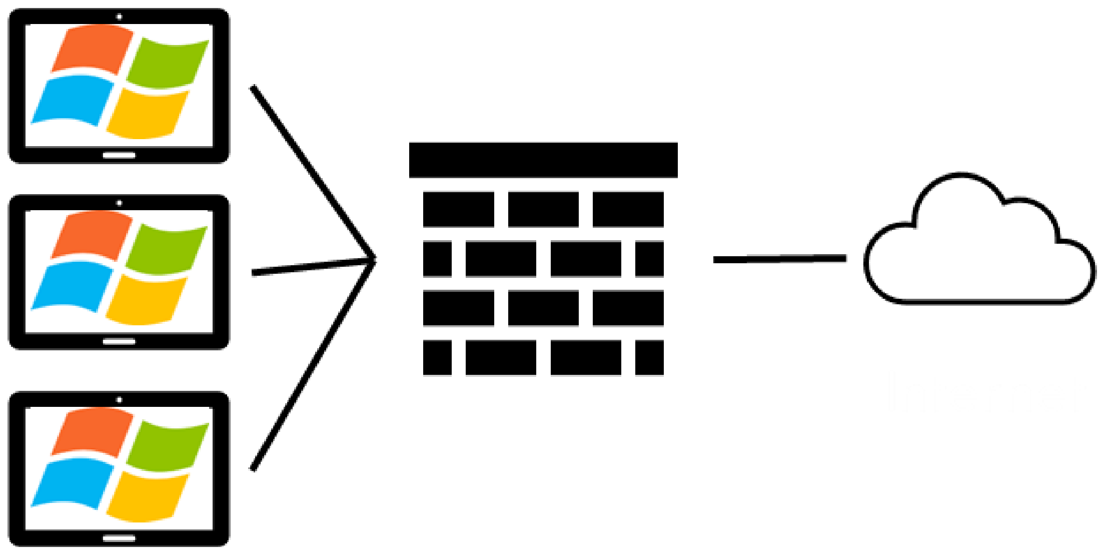


Figure 1: Firewall filtering from the network to devices

Designing a firewall is not just a simple task, it involves minimizing vulnerabilities, lowering risk of human errors, and applying the rules properly.

Background

In 2022, A study suggested that 30,000 websites got hacked everyday [2]. The industry lost 6 trillion dollars due to cyber attacks. To understand how to protect ourselves, one must learn how to use a firewall.

A firewall is a software that watches network packets (network traffic) coming through the network into the system [3]. The firewall software then decides whether it would allow these packets to come in or not. The firewall runs checks for threats and viruses or even patterns to consider all threats possible and stop them before they come through. The objective of a firewall is to be successful at stopping cyber attacks or reduce them as much as possible. The success is depicted upon the amount of successful packet stoppage and the safety it provides to its users. The real question is what it takes to make a good firewall. If the task were to be easy, then all the money lost in 2022 would have been avoided. Creating a good firewall takes a good amount of time and effort in order to perform against all possible threats.

First, a firewall is divided into many types. Three of which are the main types of firewalls: Packet filtering, Circuit-level Gateway, and Stateful firewall. The common one amongst these Three is packet filtering which checks each packet one by one at a time ensuring its full attention to each single packet. Circuit-level gateways use proxy servers to determine whether a packet is legit or not. Using this as an extra defensive line against cyber attacks. Lastly, a stateful firewall keeps track of the traffic it already monitored. When new traffic comes through it checks it upon the list of old packets and if it finds a similar one, it uses the same decision it gave to the older packet. For example, Packet A comes in and it is similar to Packet B. The firewall will then give Packet A the same treatment that Packet B got. Although there are more firewall types and many ways to filter packets, usually the most commonly used firewall is a packet filtering firewall due to its simplicity and effectiveness dealing with day-to-day usage [4].

The biggest issue with firewalls is redundancy in its rules. Firewalls decide whether a packet passes or not based on a rule. These rules are what determines how good a firewall is. Having many rules, although good, also causes huge redundancies. Sometimes when rules are not placed in the right order they could also override each other's effects. Thus, cancelling the effects allows a threat packet to go through or stop a good packet from coming in. The rule that gets affected is called the shadow rule. The more shadowing that happens the worse a firewall will perform [5].

The main 3 problems that happen due to shadowing and redundancy are: consistency, completeness and compactness. To fix these issues it is suggested to use a Structured firewall design, or a Diverse firewall design, or a Stateful firewall design. Structured firewall Design uses a diagram called 'FDD' (firewall design diagram) to design a proper set of rules. This diagram reduces human error while designing rules for a firewall. This method uses 5 techniques to improve rules creation: Reduction, Marking, Generation, Compaction, Simplification. Diverse design is different from a structured design. Two different groups work on creating the same set of rules for a project. This helps reduce human errors. Then this method uses a construction algorithm to combine the two sets of rules created. Afterwards these rules are put into an FDD. Afterwards it is run through similar steps to Structured design in order to optimize these rules. First simplification, node shaping, FDD shaping, comparison algorithms. Finally, the way to design a stateful firewall like mentioned above is to create a method to keep track of older packets (flag system for example). Once this system is in play then a stateful firewall is ready to be tested [6].

In security one must achieve 5 things: Authentication, Data Integrity, Access Control, Data Confidentiality, and Non-Repudiation. Therefore, to test a firewall, most companies used PCAP files. PCAP files are packet capture [10] files, it's a record of network packets put together to be used later. These PCAP files allow these firewalls to be tested ahead of time to maintain the 5 security key objectives as well as allowing for more security research ahead of time. Finally, firewalls could also allow IDSs (intrusion detection system) to be used within the firewall system. These IDSs are developed by high tech companies that study rules and put effort into making these rules as optimized as possible. An IDS holds many rules as well as a system of how to run these rules and packets through the firewall. It monitors the traffic for any suspicious activity using the firewall system. The main types of IDSs are NIDS (network IDS), HIDS (host IDS), SIDS (signature-based IDS), AIDS (anomaly-based IDS). IDS provides tuning to a firewall for how to look at packets and deal with them. It even changes the format of alerts provided by the firewall based on the IDS used. There are many famous IDS companies, mostly Snort and Suricata [7].

Firewall basing is very important for how a firewall design plays. Firewall basing refers to where a firewall would be placed, for example a home firewall would be inside the router or modem. This matters due to how dangerous some attacks may be on leaking certain data. Companies sometimes isolate the firewall system to their outer system to keep the inner system super protected from cyber attacks. A honey pot is a set of data used to lure attackers away from main system data. Companies often use honey pots to ensure extra security.

Objective

The objective that is covered, is to manufacture and design a firewall that is intended to diminish vulnerabilities, implement protocol rules, present techniques and methodologies that defend and extinguish attacks sourced from external and public networks. Moreover, one will attempt to optimize the set of rules to increase performance and reduce redundancy. Next students will then pursue that idea to create a product that can be installed on local modems.

Requirements

Software Requirements:
- The firewall will be built based on a Linux system
- Ubuntu VM (virtual machine)
- Oracle VM (using two VMs a Linux based and windows based)

Functional Requirements:
- Firewall must contain at least 300 rules
- Firewall must not go below 1 gbps link speed
- Firewall must find false positives and false negatives

Deliverables

Students will deliver a set of rules analysis. As well as, a set of optimized rules analysis. These rules are what is used in firewalls to monitor the traffic of network packages. An analysis of PCAP challenge files and results of how well did the firewall perform will also be part of the deliverables. The deliverables will also include a table showing a set of false positives or negatives that the firewall encountered. Show a summary of results concluding the project, this summary will include all research findings as well as experimentation results. Finally a testing table will be included with the deliverables.

All these deliverables will be included as part of reports, presentations, videos and poster fairs. Part of the deliverables will be an oral presentation, a progress report, a final report, and a poster fair presentation. The final report will include a video explaining the solution of this proposal.

System Architecture

Design Paths

In this segment path options and design choices will be discussed in detail.

Different Design options

In order to build a firewall a consideration needs to be done, what kind of system we can implement a firewall on. There are Three different main methods to implement a firewall. One is to build the firewall on a router that is connected to LAN. This router can act like a redirection of the network after the filtration that happens from the firewall. Another is to use a flash memory stick to act like a system by itself. This means that the firewall can be installed on the stick, and it can act as a new network that the PC can connect to. Lastly, a double virtual machine setup can be used to simulate the router and computer thus creating an environment to build a firewall. There is another method which was deemed infeasible, it was to download the firewall on the laptop as its operating system and use it as a router. The issue with a build like that is we basically make the entire laptop unusable for anything else. Other options were not as feasible since these ideas were originally meant to be used in the school with the computer provided by the department.

System design

As for the system designs let us consider the main three options already mentioned. The first one is the router option. The design is as follows, first connect the router to an already existing network (a home modem if done from home). From university that would be connecting the router to an ethernet line. Afterwards the computer in usage will be connected either through an ethernet line or through wi-fi to the router. After the installation of the firewall on the router the setup would be complete, and testing can commence.

The second option is to use a memory stick. The memory stick option is similar to the first option with the router but instead of having a router to install the firewall on, a flash stick will be used. Once the firewall is set up on the flash stick it can be used as a network interface connected directly to the computer, thus advancing in the project.

Finally, the last design is to use a double virtual machine setup. This works by creating the first virtual machine to be directly connected with a bridge connection to the network of choice. Then create a second machine and connect it to the first machine using a LAN connection by the software itself. What this does is simulate a computer to router connection virtually allowing more versatility to the project.

VM options

Among VM options we had: VMware, Oracle VM, and virtualization manager. We chose VMware since we are very familiar with this software from previous courses and experiences. VMware was a very close choice to be picked, sadly VMware is a costly program while we can use the free program to achieve similar results. That being said, based on research oracle VM sometimes struggles with internet simulation in comparison to other VM programs, it might be an option in the future to switch from oracle VM [8].

Software Design

To use a firewall, first an open source software must be used to operate a firewall. Among possible options for firewall open source programs: IPFire, OpenSense, and PFSense. OpenSense is a mix between IPFire and PFsense (also is the most demanding in terms of hardware power). IPFire is considered the one with better UI as well as light on hardware usage, while PFSense is the more used program. Opensense seemed to be very similar to PFSense in terms of UI and usability but since PFSense had more forums and a larger community, PFSense was the choice [9].

Testing Design

Moving forward with our current design as seen above we decided to evaluate it using PCAP files found on the web. PCAP files are files that include several types of viruses and cyber-attacks which can be downloaded and used to evaluate firewalls. The files should be handled in a very proper manner or otherwise it could lead to ruining your device. The PCAP files that we found were assessed back in 1999 and DARPA evaluated them against its own intrusion detection system (IDS) and from there they posted their own results. The PCAP files were categorized from Monday to Friday and each day has two dumps, out and in. We assessed them both and got the results as seen below. Note that we used two different machines and two different (IDS's) Suricata and Snort. The specs of each machine are written below. There will also be different classes for each IDS. Each class will have different amounts of rules and the rules are optimized as the number of rules increases which will also mean upgrading to the next class. Using Snort in red and Suricata in blue. To identify which is the better IDS we had five distinct categories. The first is CPU utilization percentage, to see how much CPU power is being used while running each IDS, second is memory usage percentage to see how much of the memory is being used, third is the number of packets sent at a time, fourth is how many alerts are generated and finally time, to see how much time it took while looking through each file.

Machines	***Specs***	***Rules***	***IDS***
MACH1	Cpu i9-10900 @2.8 GHZ 4GB RAM CPU CORES: 2	Standard: 41921 Added:47527	Suricata
MACH2	Cpu is i9-9900k @3.6ghz 4GB RAM CPU CORES: 2	Com: 4239 Reg: 44669 Sub: 44807	Snort

Note: Machine 2 is around 10% stronger than Machine 1

Path alterations

There are 2 major path alterations that happened during this project. The first one was changing our entire set up from a school set up to a home set up for both students. Due to school policies building a firewall on school network is not allowed (more on this in the setbacks sections).

The second major path change affected how we test PCAP files. As the next section will explain our design (using double VM setup) we essentially run into the issue where we can not run these PCAP files since there is no way for us to run them safely. Running those PCAP files on our network will cause our home networks to be infested with huge packets including viruses. In order to maintain those viruses, we attempted to run the double VM set up within a VM. That way the main VM can run those PCAPs safely while the double VM within can analyze and test those PCAPs. This was changed yet again since running a VM within a VM required altering how our windows runs virtualizing and playing with its settings. Thus, we decided to go for another method to test those PCAPs. Instead of trying to run those PCAPs through the firewall, we can run them through the IDS from the command line. The command line feature plus some extra steps will display all the information the firewall would have analyzed. Therefore, we will get the same results (more on this in design part 2 section).

Design

Part 1 (VM setup)

The design of choice will be a Double VM setup. In order to simulate how a firewall functions, it is needed to run multiple virtual machines. The first VM will act as the firewall. It will have PFSense installed and running on it. This first VM will be filtering the internet to a local network of choice. The second VM will be a mint windows system using the local network we created for the first VM as its main network. This way the second VM will get the filtered network. The following figure shows the set up for this project:

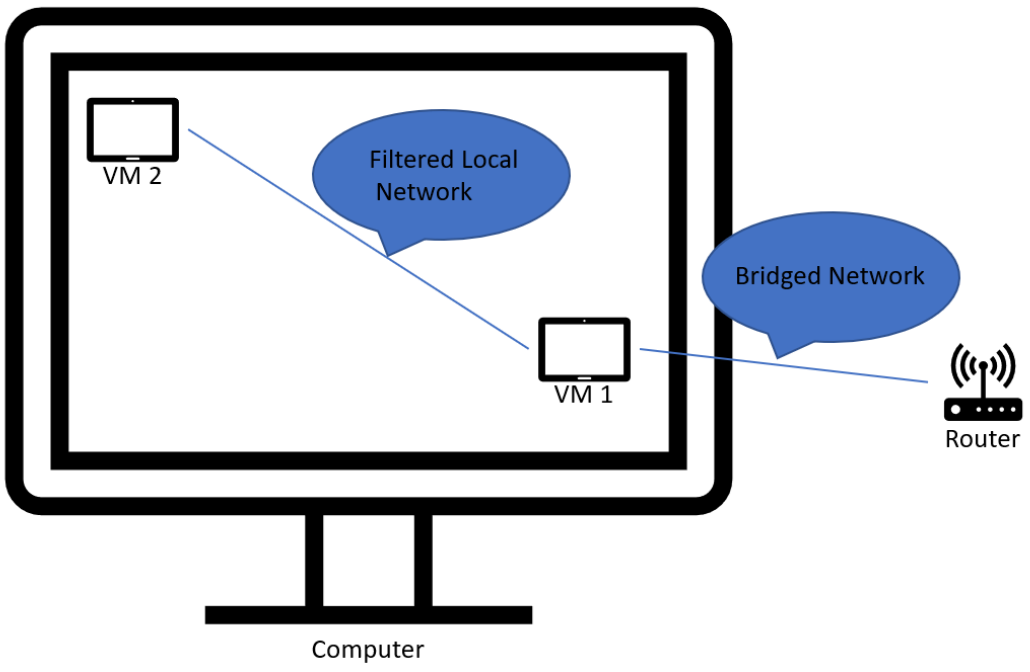
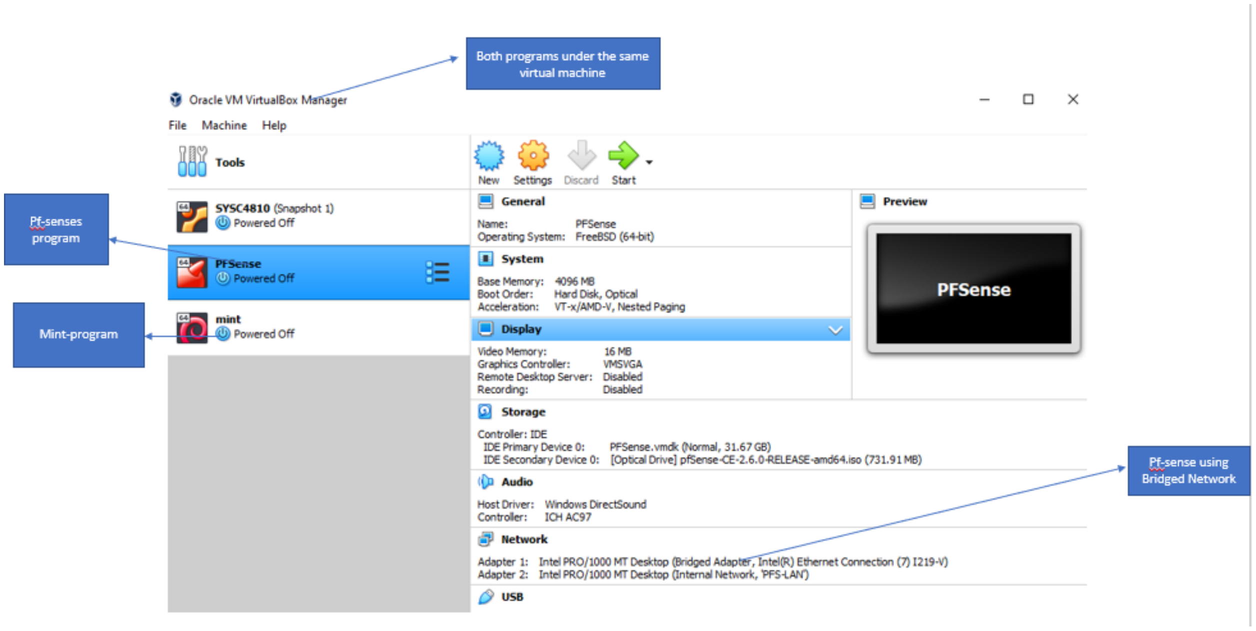


Figure 2: Firewall Setup

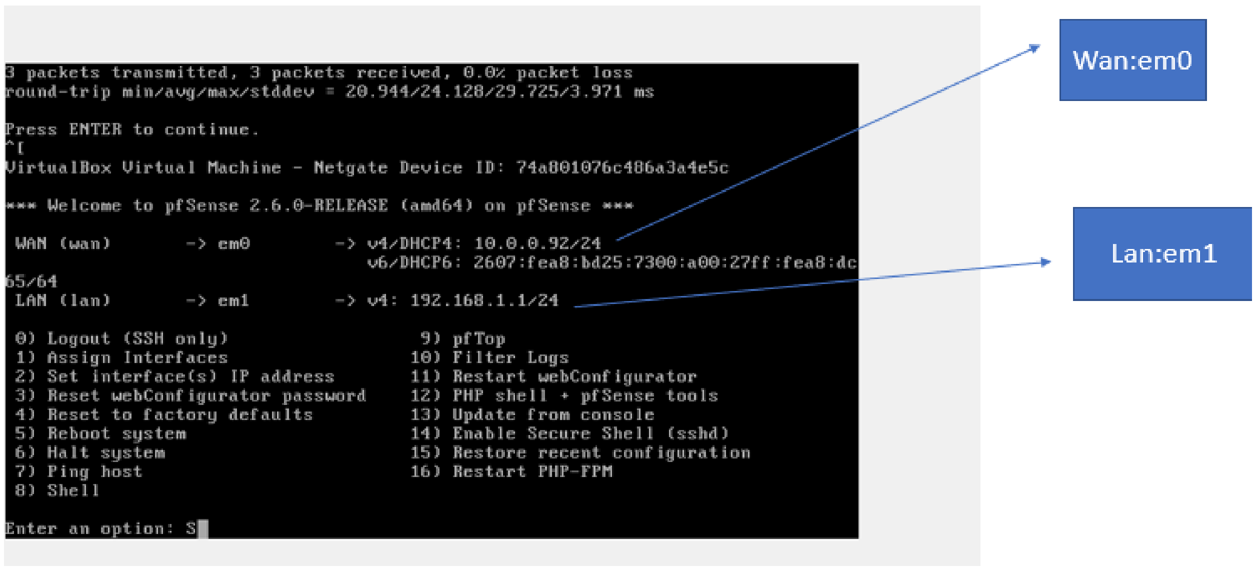
As shown above, the yellow stream signifies the network packets going through the project. At first it goes from the router all the way to the first VM. The firewall VM will then filter this stream of network and through its local connection to the second VM we can monitor this filtered network. The following shows how the setup looks like:



```
General
Name:               PFSense
Operating System:   FreeBSD (64-bit)

System
Base Memory:    4096 MB
Boot Order:     Hard Disk, Optical
Acceleration:   VT-x/AMD-V, Nested Paging

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

Storage
Controller: IDE
 IDE Primary Device 0:     PFSense.vmdk (Normal, 31.67 GB)
 IDE Secondary Device 0:   [Optical Drive] pfSense-CE-2.6.0-RELEASE-amd64.iso (731.91 MB)

Audio
Host Driver:   Windows DirectSound
Controller:    ICH AC97

Network
Adapter 1:   Intel PRO/1000 MT Desktop (Bridged Adapter, Intel(R) Ethernet Connection (7) I219-V)
Adapter 2:   Intel PRO/1000 MT Desktop (Internal Network, 'PFS-LAN')

USB
```

Figure 3: Oracle-VM set-up

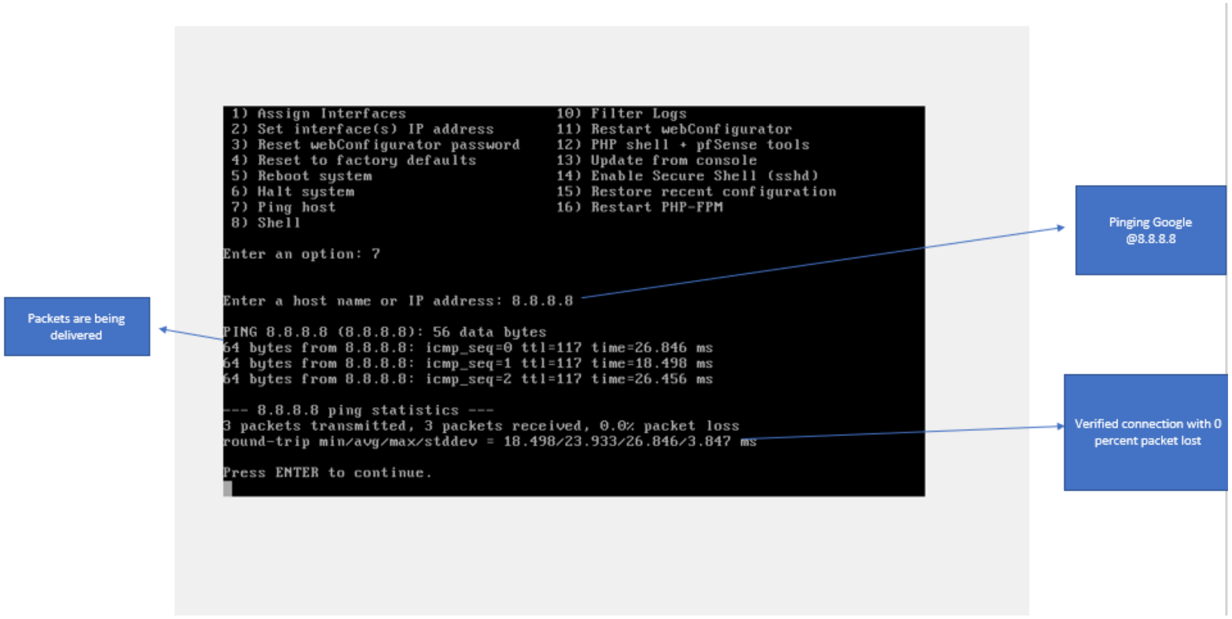
The figure above provides a view of how we set-up the virtual machine, as seen above PF-sense will be using a bridged network. Bridged network allows the VM to directly use our own network. It will also have a second network adapter called internal network (this is the local network being filtered). The mint VM will be using the internal network to access the filtered network.



```
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 20.944/24.128/29.725/3.971 ms

Press ENTER to continue.
^[
VirtualBox Virtual Machine - Netgate Device ID: 74a801076c486a3a4e5c

*** Welcome to pfSense 2.6.0-RELEASE (amd64) on pfSense ***

 WAN (wan)       -> em0        -> v4/DHCP4: 10.0.0.92/24
                                  v6/DHCP6: 2607:fea8:bd25:7300:a00:27ff:fea8:dc
65/64
 LAN (lan)       -> em1        -> v4: 192.168.1.1/24

 0) Logout (SSH only)                  9) pfTop
 1) Assign Interfaces                 10) Filter Logs
 2) Set interface(s) IP address       11) Restart webConfigurator
 3) Reset webConfigurator password    12) PHP shell + pfSense tools
 4) Reset to factory defaults         13) Update from console
 5) Reboot system                     14) Enable Secure Shell (sshd)
 6) Halt system                       15) Restore recent configuration
 7) Ping host                         16) Restart PHP-FPM
 8) Shell

Enter an option: S
```

Figure 4: PfSense setup

The first time PFsense was run, it triggered an installer. After the installation and setup is complete it has this display page. This will show us the basic controls of the program and how to use it as needed. We can see that it is detecting the Wan (bridged network) and Lan(local network), this is good because without them we won't be able to proceed to the next step.

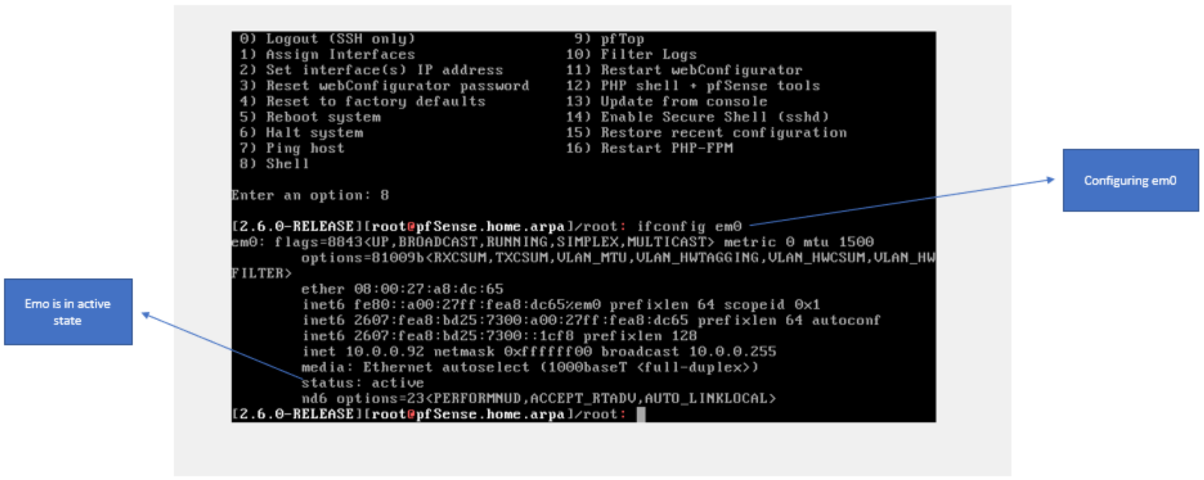


```
 1) Assign Interfaces                     10) Filter Logs
 2) Set interface(s) IP address           11) Restart webConfigurator
 3) Reset webConfigurator password        12) PHP shell + pfSense tools
 4) Reset to factory defaults             13) Update from console
 5) Reboot system                         14) Enable Secure Shell (sshd)
 6) Halt system                           15) Restore recent configuration
 7) Ping host                             16) Restart PHP-FPM
 8) Shell

Enter an option: 7

Enter a host name or IP address: 8.8.8.8

PING 8.8.8.8 (8.8.8.8): 56 data bytes
64 bytes from 8.8.8.8: icmp_seq=0 ttl=117 time=26.846 ms
64 bytes from 8.8.8.8: icmp_seq=1 ttl=117 time=18.498 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=117 time=26.456 ms

--- 8.8.8.8 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 18.498/23.933/26.846/3.847 ms

Press ENTER to continue.
```

Pinging Google @8.8.8.8
Packets are being delivered
Verified connection with 0 percent packet lost

Figure 5: Pinging google test

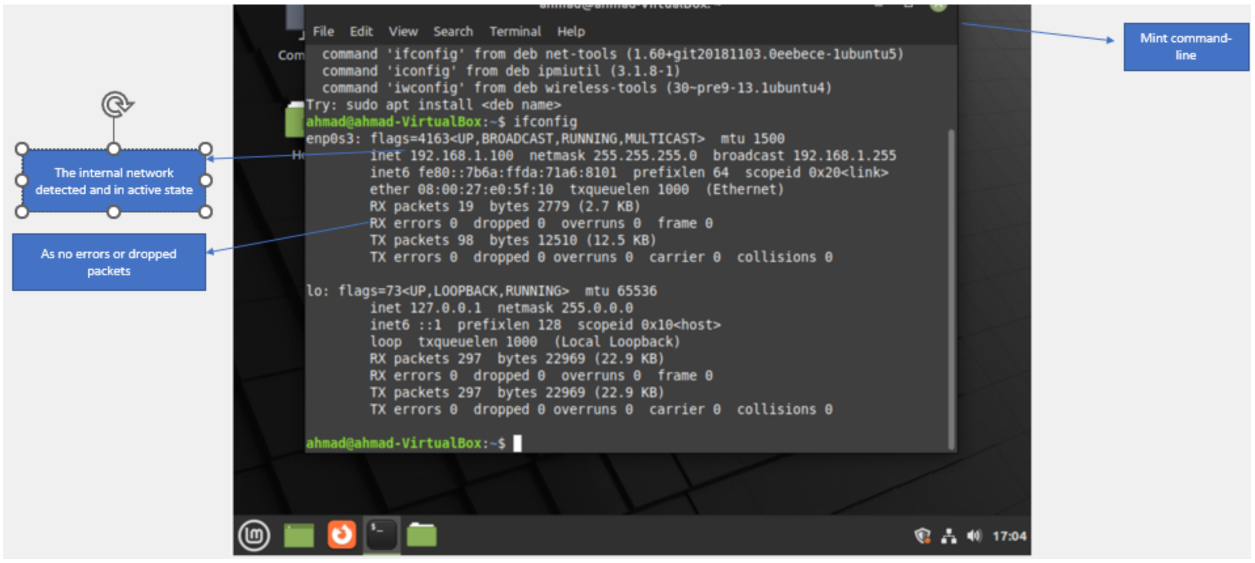
In this figure we tried to ping google to see if the network was running as intended. We used Google's public server which is ‘8.8.8.8’. As seen above it pinged and the bytes were being transferred, with 0 percent packets lost.



```
 0) Logout (SSH only)                      9) pfTop
 1) Assign Interfaces                     10) Filter Logs
 2) Set interface(s) IP address           11) Restart webConfigurator
 3) Reset webConfigurator password        12) PHP shell + pfSense tools
 4) Reset to factory defaults             13) Update from console
 5) Reboot system                         14) Enable Secure Shell (sshd)
 6) Halt system                           15) Restore recent configuration
 7) Ping host                             16) Restart PHP-FPM
 8) Shell

Enter an option: 8

[2.6.0-RELEASE][root@pfSense.home.arpa]/root: ifconfig em0
em0: flags=8843<UP,BROADCAST,RUNNING,SIMPLEX,MULTICAST> metric 0 mtu 1500
        options=81009b<RXCSUM,TXCSUM,VLAN_MTU,VLAN_HWTAGGING,VLAN_HWCSUM,VLAN_HW
FILTER>
        ether 08:00:27:a8:dc:65
        inet6 fe80::a00:27ff:fea8:dc65%em0 prefixlen 64 scopeid 0x1
        inet6 2607:fea8:bd25:7300:a00:27ff:fea8:dc65 prefixlen 64 autoconf
        inet6 2607:fea8:bd25:7300::1cf8 prefixlen 128
        inet 10.0.0.92 netmask 0xffffff00 broadcast 10.0.0.255
        media: Ethernet autoselect (1000baseT <full-duplex>)
        status: active
        nd6 options=23<PERFORMNUD,ACCEPT_RTADV,AUTO_LINKLOCAL>
[2.6.0-RELEASE][root@pfSense.home.arpa]/root:
```

Figure 6: em0 status

This shows a shell command to check for the IP configuration of the Wan and Lan connections. em0 being the Wan and em1 being the Lan. As it shows that the status is up which means it is connected to the receiving network.


File Edit View Search Terminal Help
Mint command-line
The internal network detected and in active state
As no errors or dropped packets
17:04

```
 command 'ifconfig' from deb net-tools (1.60+git20181103.0eebece-1ubuntu5)
 command 'iconfig' from deb ipmiutil (3.1.8-1)
 command 'iwconfig' from deb wireless-tools (30~pre9-13.1ubuntu4)
Try: sudo apt install <deb name>
ahmad@ahmad-VirtualBox:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.100  netmask 255.255.255.0  broadcast 192.168.1.255
        inet6 fe80::7b6a:ffda:71a6:8101  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:e0:5f:10  txqueuelen 1000  (Ethernet)
        RX packets 19  bytes 2779 (2.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 98  bytes 12510 (12.5 KB)
        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 297  bytes 22969 (22.9 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 297  bytes 22969 (22.9 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

ahmad@ahmad-VirtualBox:~$
```

Figure 7: Mints command line

When running Mint after running PF-sense we wanted to check if the network was being detected or not. The figure above shows that it is running fine with no errors and we are connected through the internal network. The Inet shows the same IP as pf-sesne.

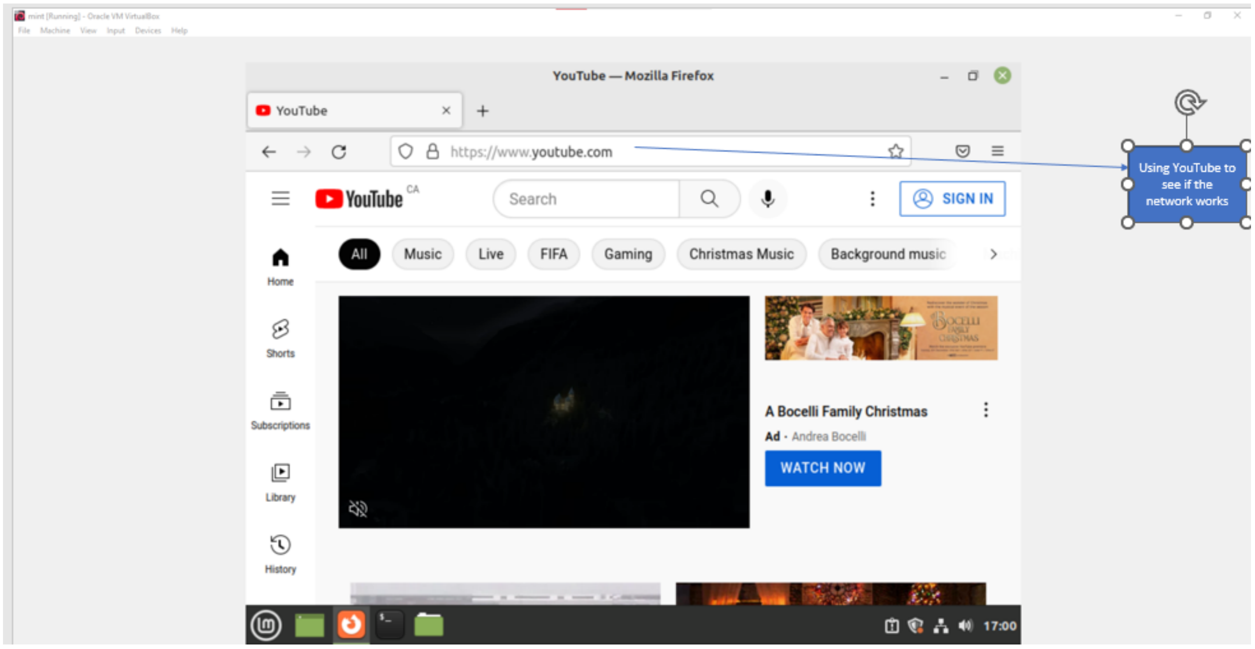


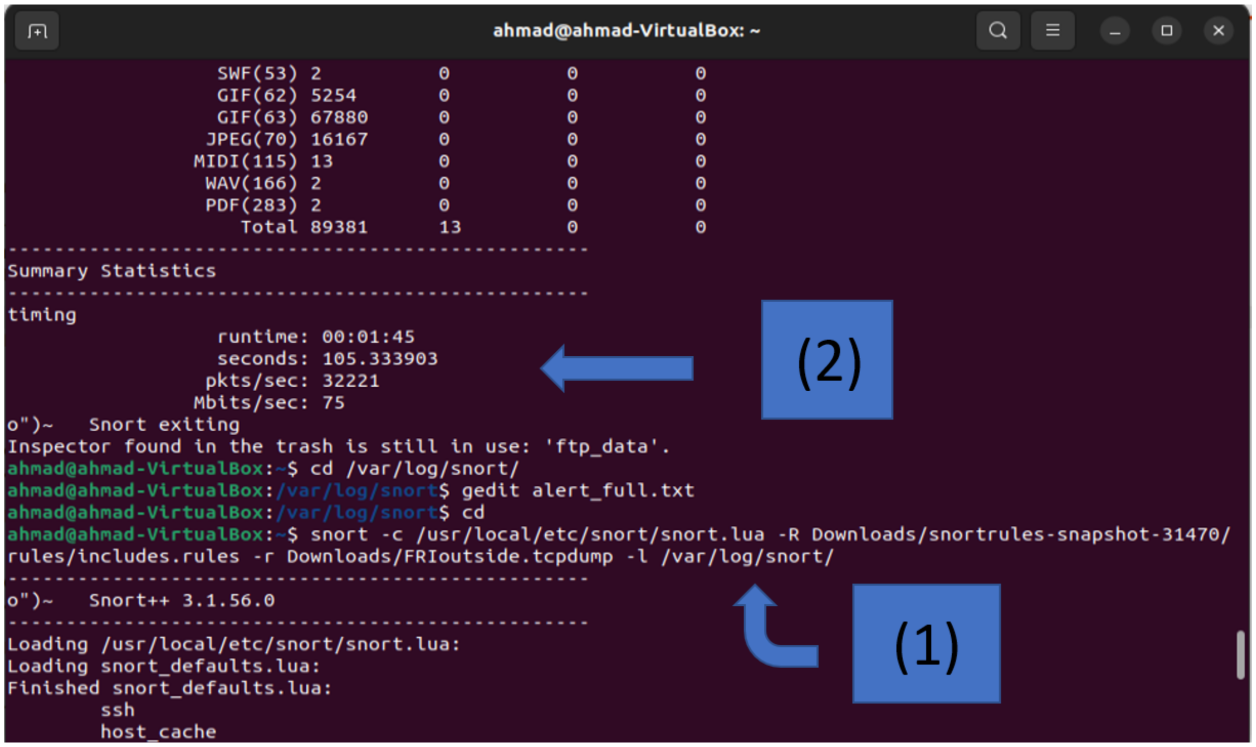
Figure 8: Testing the network

Lastly, we wanted to see if it actually works and we used Firefox to run Youtube and it loaded Youtube fine. This proves that the setup works as wanted and from here we can go ahead and start running the rules.

Part 2 (PCAP Testing)

In order to fully test PCAP files and run them against the IDS choices. We had to somehow run the PCAP files through the network of the firewall. To do so we will need to create a double VM setup inside a VM. The reason is that PCAP files are not safe. Running them through our network means the network that our families use will be transmitting viruses.

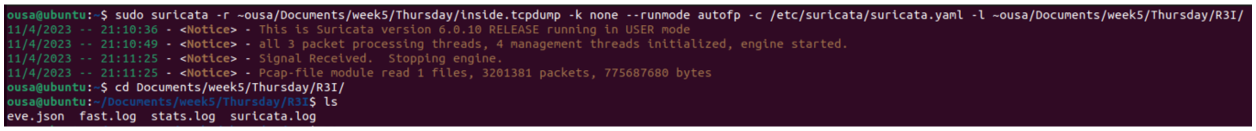
Since we could not run the double VM set up inside a VM without playing with the windows settings. We instead opted to use the Suricata and Snort command line testing. These command lines allow us to run a PCAP file through the IDS and it will generate alerts and give out data that the firewall would give out.



```
                  SWF(53) 2         0          0          0
                  GIF(62) 5254      0          0          0
                  GIF(63) 67880     0          0          0
                 JPEG(70) 16167     0          0          0
                MIDI(115) 13        0          0          0
                 WAV(166) 2         0          0          0
                 PDF(283) 2         0          0          0
                    Total 89381     13         0          0
--------------------------------------------------
Summary Statistics
--------------------------------------------------
timing
                  runtime: 00:01:45
                  seconds: 105.333903
                 pkts/sec: 32221
                Mbits/sec: 75
o")~   Snort exiting
Inspector found in the trash is still in use: 'ftp_data'.
ahmad@ahmad-VirtualBox:~$ cd /var/log/snort/
ahmad@ahmad-VirtualBox:/var/log/snort$ gedit alert_full.txt
ahmad@ahmad-VirtualBox:/var/log/snort$ cd
ahmad@ahmad-VirtualBox:~$ snort -c /usr/local/etc/snort/snort.lua -R Downloads/snortrules-snapshot-31470/
rules/includes.rules -r Downloads/FRIoutside.tcpdump -l /var/log/snort/
--------------------------------------------------
o")~   Snort++ 3.1.56.0
--------------------------------------------------
Loading /usr/local/etc/snort/snort.lua:
Loading snort_defaults.lua:
Finished snort_defaults.lua:
        ssh
        host_cache
```

Figure 9: Snort running through a command line

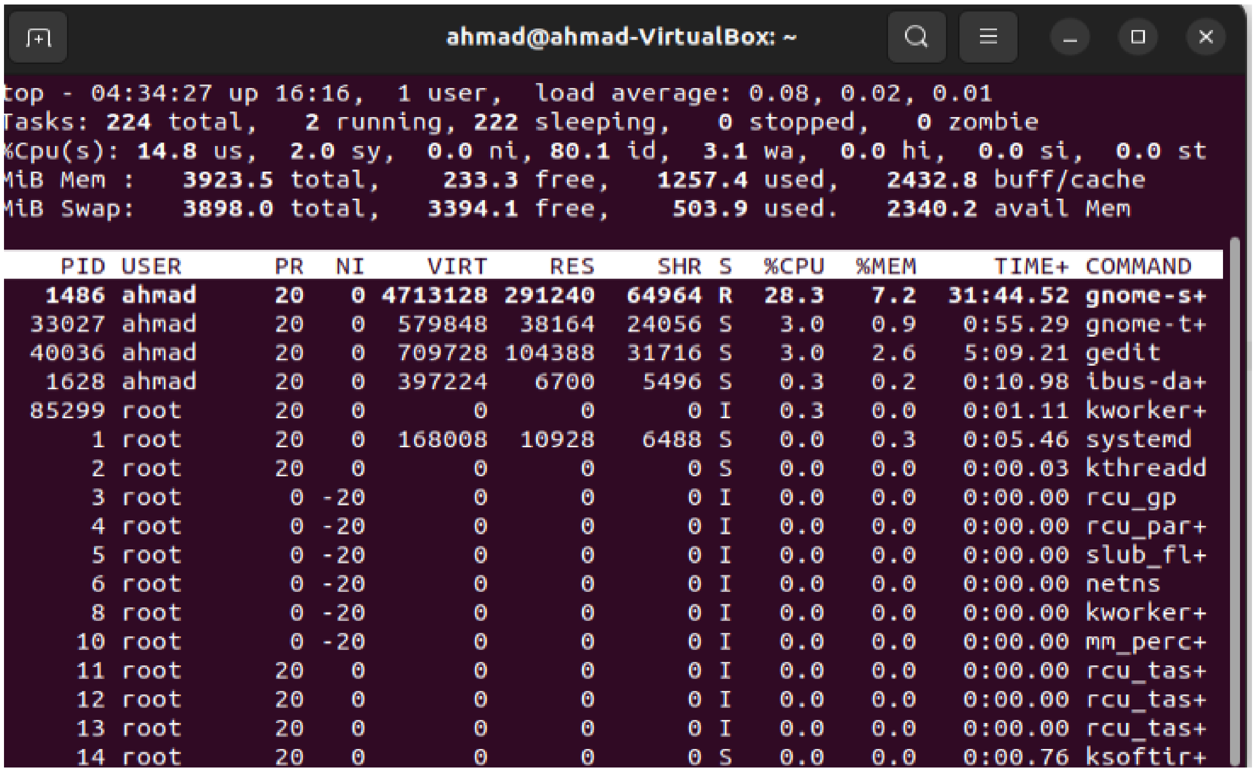
Figure 9 shows what running snort through a command line looks like. Number 1 shows what the command is. Number 2 shows sample information that snort prints out about the PCAP file that was just tested. In the command (like number 1), we can choose what type of rules levels to use as well as what kind of PCAP file to test. The last part helps save all the alerts and logs them into a long format file in the given file location. Sample data like alert count/time/memory usage/CPU usage is all shown by the snort command line.



```
ousa@ubuntu:~$ sudo suricata -r ~ousa/Documents/week5/Thursday/inside.tcpdump -k none --runmode autofp -c /etc/suricata/suricata.yaml -l ~ousa/Documents/week5/Thursday/R3I/
11/4/2023 -- 21:10:36 - <Notice> - This is Suricata version 6.0.10 RELEASE running in USER mode
11/4/2023 -- 21:10:49 - <Notice> - all 3 packet processing threads, 4 management threads initialized, engine started.
11/4/2023 -- 21:11:25 - <Notice> - Signal Received.  Stopping engine.
11/4/2023 -- 21:11:25 - <Notice> - Pcap-file module read 1 files, 3201381 packets, 775687680 bytes
ousa@ubuntu:~$ cd Documents/week5/Thursday/R3I/
ousa@ubuntu:~/Documents/week5/Thursday/R3I$ ls
eve.json  fast.log  stats.log  suricata.log
```

Figure 10: Suricata running through a command line

Figure 10 shows what running through a Suricata command looks like. The first line in white indicates the command line that Suricata uses to run. The Suricata command saves all information about the time, alerts, and number of rules in the logs. These logs are saved with custom settings that the user sets while installing Suricata.



```
top - 04:34:27 up 16:16,  1 user,  load average: 0.08, 0.02, 0.01
Tasks: 224 total,   2 running, 222 sleeping,   0 stopped,   0 zombie
%Cpu(s): 14.8 us,  2.0 sy,  0.0 ni, 80.1 id,  3.1 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   3923.5 total,    233.3 free,   1257.4 used,   2432.8 buff/cache
MiB Swap:   3898.0 total,   3394.1 free,    503.9 used.   2340.2 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
   1486 ahmad     20   0 4713128 291240  64964 R  28.3   7.2  31:44.52 gnome-s+
  33027 ahmad     20   0  579848  38164  24056 S   3.0   0.9   0:55.29 gnome-t+
  40036 ahmad     20   0  709728 104388  31716 S   3.0   2.6   5:09.21 gedit
   1628 ahmad     20   0  397224   6700   5496 S   0.3   0.2   0:10.98 ibus-da+
  85299 root      20   0       0      0      0 I   0.3   0.0   0:01.11 kworker+
      1 root      20   0  168008  10928   6488 S   0.0   0.3   0:05.46 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.03 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par+
      5 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 slub_fl+
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
      8 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker+
     10 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_perc+
     11 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     12 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     13 root      20   0       0      0      0 I   0.0   0.0   0:00.00 rcu_tas+
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.76 ksoftir+
```

Figure 11: TOP command used in Linux system

Figure 11 shows a command called 'top' which essentially showcases the memory and CPU usage of each program. This includes Snort and Suricata running. This is how the CPU% and MEM% were measured.

Set Backs

First when deciding which open source firewall to choose. We first choose IPFire. IPFire was a mess to get it to work due to issues with how the VM works. So, we switched to PFSense. Initially PFsense was not working which we soon found out that the way PFSense installed itself was weird so we had to fix how Oracle VM storage worked.

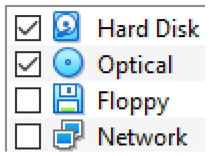
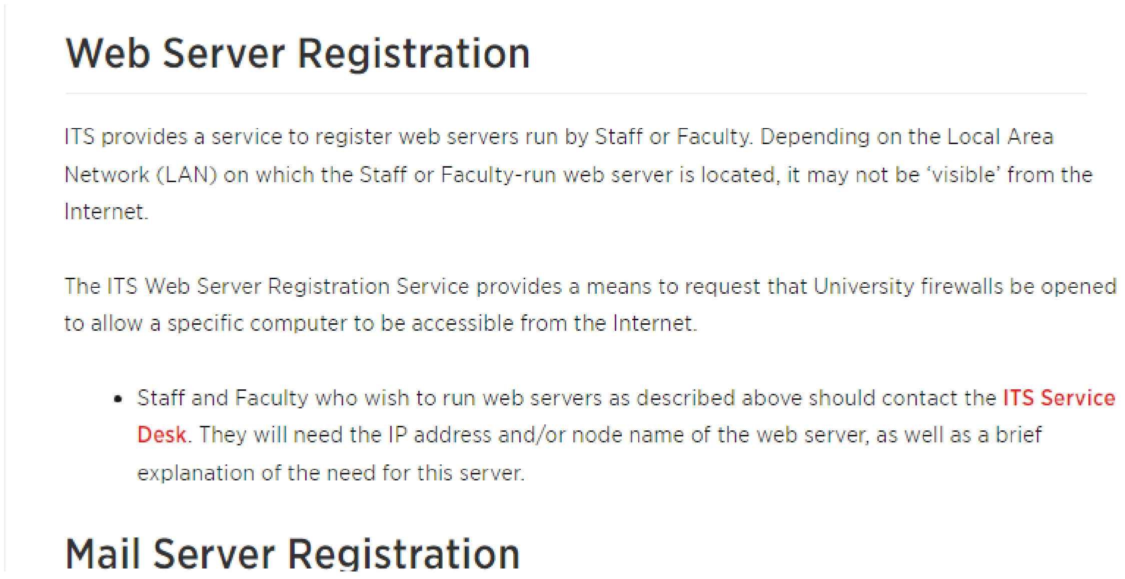


Figure 12: Storage system

As shown in figure 12, we had to move down the floppy disk and disable it while moving up the Hard Disk. After this Issue was resolved we needed to connect the second VM network to the first one. This was a very quick solution since all we needed to do is add a second network adapter. The next step was to figure out a way to run this on a computer on campus.

At first when setting this up on campus we found out that the computer at campus does not have a wi-fi card. This means that we were restricted to only using ethernet. When running the same setup, the network was not working even though it was connected to the network. When setting it up it would say that it detects the IP address for the WAN but not shows the ip address for some reason. When we used the shell commands just like figure 6, the Wan status was up but with IP ‘0.0.0.0’. This was the first time we saw an issue like this. We proceeded to debug this issue. First we tried reading online about this and testing different methods to detect the proper IP address. We used manual approaches but nothing worked. Next we assumed that maybe the ethernet connection was the issue not our setup itself. So a friend gave us their laptop to use to download the setup while using the school wi-fi (since we can't use ethernet on the school computer). Even though we changed the network completely it still did not work. Next we tried to use a phone hotspot to connect to the laptop. Surprisingly this worked just fine. This means that there is something wrong with the school network that does not allow us to use its network. After sending emails to the service desk and some research on the school website we found out that the school prohibits the use of servers on its network architecture.



Web Server Registration

ITS provides a service to register web servers run by Staff or Faculty. Depending on the Local Area Network (LAN) on which the Staff or Faculty-run web server is located, it may not be ‘visible’ from the Internet.

The ITS Web Server Registration Service provides a means to request that University firewalls be opened to allow a specific computer to be accessible from the Internet.

- Staff and Faculty who wish to run web servers as described above should contact the ITS Service Desk. They will need the IP address and/or node name of the web server, as well as a brief explanation of the need for this server.

Mail Server Registration

Figure 13: School website about servers

The figure above shows how servers are not allowed to run on the school network. Our next step was to try and buy a router which could potentially allow us to bypass that issue so we could run this firewall in school. Sadly, the router did not change this outcome. This meant that we now need a new plan for our poster fair since we can not run this firewall in university, instead we will both run this firewall at home and develop at home as well.

When looking at PCAP files we were searching the web for a source that included the PCAP files and results. This task by itself was daunting since we could not find a source that had those two requirements, and they are necessary. Otherwise, how are we going to test the intrusion detection system? Until we found the DARPA test, which took place in 1999 and was named DARPA INTRUSION DETECTION EVALUATION DATASET. It had almost everything we were looking for, but we were hesitant to use it as it was almost 24 years old. We decided to use it as it was the best option on the internet at the time, although it was missing a few details that we needed such as their rulesets and false negatives.

After setting up at home. Moving to the next step was to test PCAP files. This is where our next set back came. The PCAP files needed to run somewhere outside the firewall setup so that when they run through the firewall, it can be filtered. This way we can test the performances and apply some analysis. This was an issue since the double VM setup depends on the bridged connection to get its packets. This means that we needed to run those PCAPs on our home networks.

Running those Packets in our network was not safe due to those PCAPs including viruses and various types of attacks. The solution that first came in hand was to try to have a double VM setup inside a VM. That did not work since we needed to alter a lot of windows settings to make it happen. The issue did not stop there since PFSense had direct support to Snort IDS but did not have a direct line to connect to Suricata. The way to connect to Sucirata was to use a plugin and manually connect the rules or the sets. This was inconvenient since it required a lot of manual work. Suricata included so many rules that having to manually add some of them was a huge task.

To get around this issue we started using the IDS command line features which allowed us to run the IDS in a Linux system. This made it very easy to test those PCAP files safely inside a VM while still getting all the data we needed from a firewall to analyse and gather data.

Implementations

In order to run all this and have consistent results between both of us. We both needed to run this from home, thus we needed something with similar machines or at least minimal difference in performance to make sure the results are consistent.

Both Students who had machine one or two downloaded Oracle VM. Created the two Virtual machines and ran through the setup. Then both students downloaded the proper VMs from authentic online sources. Next in order to bypass a bug in Oracle VM trying to run PFSense some settings in Oracle VM needed to be adjusted. Afterwards The two students went through the PFSense set up and connected it to the Linux virtual machine (second VM). Then proceeded to run through tests to confirm the firewall is working as intended. Both students then connected their respective IDSs to the firewall through the firewall GUI. Both students ran some tests before switching into the new plan to test the PCAPs.

Both students then downloaded their respective IDSs through the command line in the Linux virtual machine. Afterwards Students downloaded the PCAP files form the DARPA research and started running through their tests. Those results were recorded and noted down in the report.

Scope

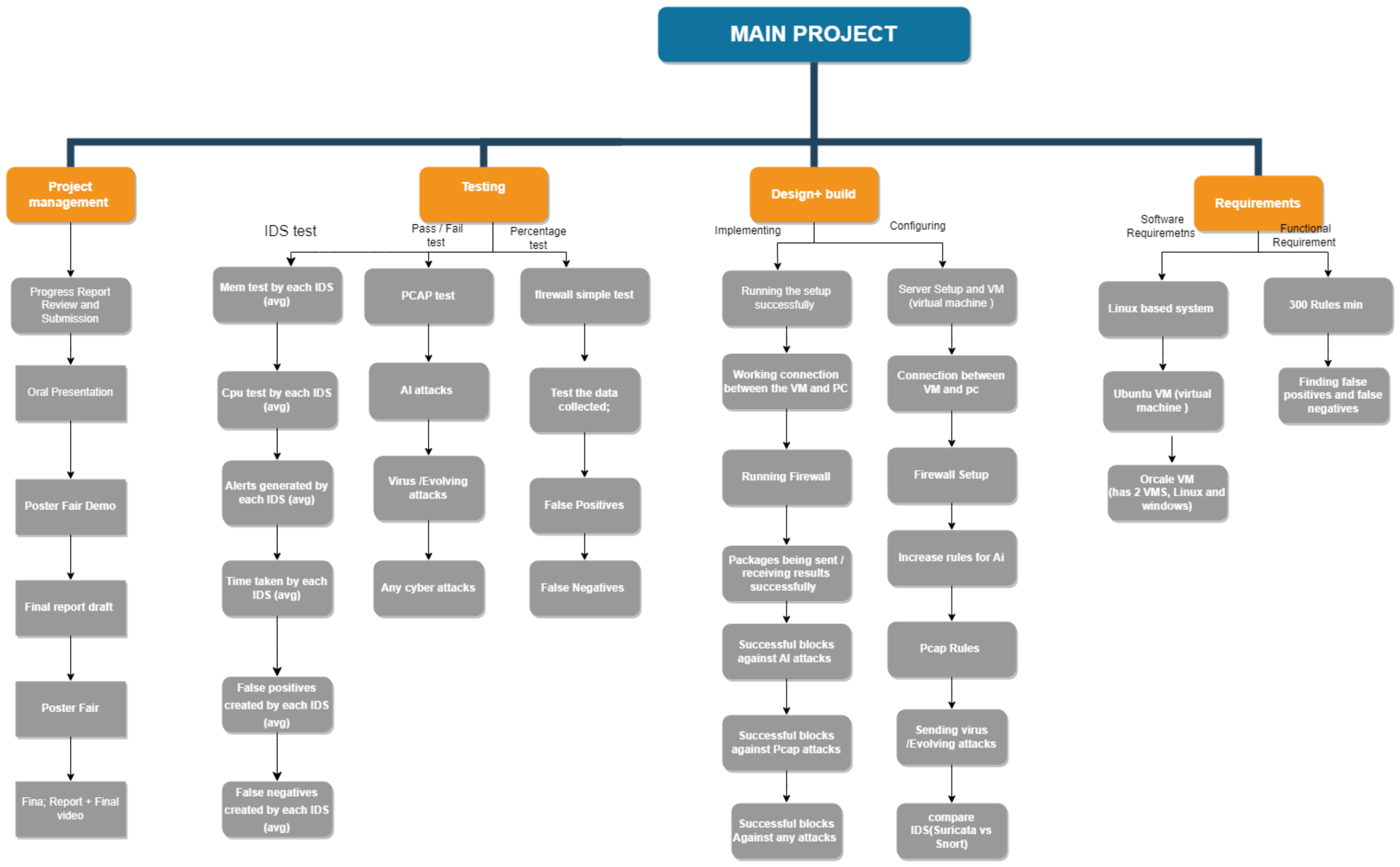
Activity List

The following is a table of activities that the students will have during this project period:

	Activity Name	**Description**	**Expected length (in days)**
1	Group comes up with requirements	Both students work on finding proper requirements for the firewall	3
2	Acquire Room/Pc	Students ask for a room and a pc in university to work on capstone project	1
3	Run the pc and set up	Build up the pc and download any needed programs. Figure out the languages needed.	3
4	Learn server/VM set up	Find a proper VM to run on pc	3
5	Set up the Pc with the server/VM	Run the VM and set it up on the pc	3
6	Proposal write up	Write a proposal to delivery	2
7	Figure out how to use the firewall open source program	Research firewall open source programs	3
8	Run firewall on PC and observe incoming traffic	Run the open source program on the VM and start to observe the traffic it detects	2
9	Start using 1 rule into 5 rules into 10	In the firewall start adding rules to filter the packets. Start by one then increase it to 10	14
10	Collect data on those 10 rules/research about Suricata and Snort rules	Observe and note down how these 10 rules affected the traffic. Research about Suricata rules and Snort rules to use them in project	3
11	Progress Report	Write a progress report that includes what has been done for the past couple weeks	3

12	Poster fair demo	Preparing something to showcase at the poster fair. Mainly to present our project to the public	3
13	Oral Presentation	Creating a presentation for certain speakers	4
14	Increase amount of rules until satisfied	Increase the rules by adding Snort IDS into the firewall	20
15	Start prepping a PCAP test/challenge and try it out.	Get PCAP files from an online source to test our rules	10
16	Compare Snort vs Suricata	Compare the two IDS softwares against each other in performance	20
17	Send a strong virus and see how the rules react to it	Send in PCAPs that include strong viruses	20
18	Final report Draft	Start building the final report and submit it an early version to get feedback	4
19	Poster Fair	This is where we show what we prepared in the demo to the public	3
20	Finish video on project	Finish and edit a video to submit	2
21	Oral presentation 2	Creating a presentation for certain speakers	4
22	Final Report including video	Final report write up and making a video including all of our work	6

Work Breakdown Structure


MAIN PROJECT
Project management
Progress Report Review and Submission
Oral Presentation
Poster Fair Demo
Final report draft
Poster Fair
Fina; Report + Final video
Testing
IDS test
Pass / Fail test
Percentage test
Mem test by each IDS (avg)
Cpu test by each IDS (avg)
Alerts generated by each IDS (avg)
Time taken by each IDS (avg)
False positives created by each IDS (avg)
False negatives created by each IDS (avg)
PCAP test
AI attacks
Virus /Evolving attacks
Any cyber attacks
firewall simple test
Test the data collected;
False Positives
False Negatives
Design+ build
Implementing
Configuring
Running the setup successfully
Working connection between the VM and PC
Running Firewall
Packages being sent / receiving results successfully
Successful blocks against AI attacks
Successful blocks against Pcap attacks
Successful blocks Against any attacks
Server Setup and VM (virtual machine)
Connection between VM and pc
Firewall Setup
Increase rules for Ai
Pcap Rules
Sending virus /Evolving attacks
compare IDS(Suricata vs Snort)
Requirements
Software Requiremetns
Functional Requirement
Linux based system
Ubuntu VM (virtual machine)
Orcale VM (has 2 VMS, Linux and windows)
300 Rules min
Finding false positives and false negatives

Testing

The testing section was divided into three tables: the first one is the percentage table, second is the pass/fail and finally the IDS test. We needed three categories because they did not all fit on the same table as they would be tested through various aspects. Looking at the percentage table for example we can see that the tests can pass the test with just getting over 50 percent, of course that means it passed with bare minimum which will need to be worked on longer until it reaches a better rate, but it will still pass. On the second hand we have the pass and fail table which the tests pass or fail not in between. This is because the success of the project depends on the passing of these tests or otherwise the project will be facing a lot of problems and issues and will not be finished on time. Finally, we have the IDS test that will be testing each intrusion detection system individually. The tests are chosen to examine the performance of Suricata and Snort, as each assessment will inform us which IDS is the best for our project. The tests are chosen carefully to make sure the project meets the requirements and exceeds them.

Percentage Table :

The firewall simple test scored 100 percent as it passed with excellence and blocked the attacks as intended. The collected data scored 100 percent as well because the data provided was the data needed, for example it showed how many alerts generated or time taken and the blocked packets were the correct ones, etc. The false positive and false negative should be the exact opposite as it means that the errors were minimal, and the percentage should be below 50 %.

Test name	Description	Results (Percentage)
Firewall simple test	*This will test our firewall against small attacks which have only 5 rules such as can not be too big or too small etc.*	100 %
Collected data test	*This will test our data that is collected to see if the firewall was working successfully or not and to see if the data collections make sense*	100 %
False positive	*Will provide results using both IDS's to know the average of how many false positives we had in the whole system*	42 %
False negative	*Will provide results using both IDS's to know the average of how many false negatives we had in the whole system*	ERROR %*

*The reason this was an error was because both IDSs had different false negative results and those results were not accurate due to the nature of the test. More on this in the ANALYSIS section.

Pass/Fail Table :

The PCAP test was a pass because all the PCAP files that attacked both IDS's were blocked. The AI attacks and the evolving attacks that were in the PCAP files were also blocked, that is why it passed. The minimum packages sent at a time were three hundred, but we went beyond and got up to 3 million plus packets that were sent at a time and that made it pass.

Test name	Description	Results (pass/fail)
PCAP test	*Test that will test our firewall and see how strong it is against attacks*	PASS
AI attacks	*Testing the firewall against the attacks that will keep on attacking and trying to pass all the rules*	PASS
Virus/Evolving attacks	*This test will be showing if the virus and the evolving attacks can go through the firewall*	PASS
Minimum packages at a time	*This will insure that the sent packages will be 300 at a time*	PASS

IDS tests :

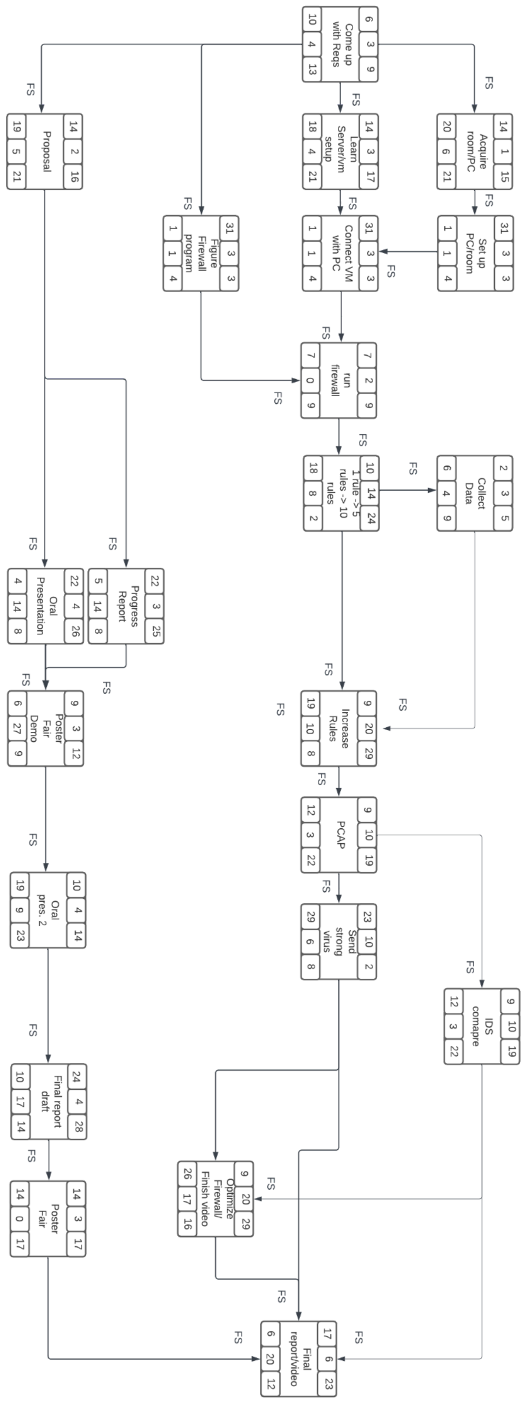
The sample used for this test was using the Friday dumps. We couldn't do the other days as they were humanly impossible to assess and also Friday had the least number of alerts. The average of the tests here were done based on Friday inside TCP dumps and out TCP dumps, which can be found below in the results. These tests showcase how much hardware is being used while they are generating these alerts. Another reason for these tests would be to show how accurate the blockages are.

Suricata Mem test (avg)	*Test will provide accurate data on how much memory utilization is being used by each IDS*	21.13%
Snort Mem test (avg)	*Test will provide accurate data on how much memory utilization is being used by each IDS*	7.85%
Suricata CPU test (avg)	*Test will provide accurate data on how much CPU power is being used by each IDS*	99.05%
Snort CPU test (avg)	*test will provide accurate data on how much CPU power is being used by each IDS*	97.97%
Alerts Snort (avg)	*Test will show how many alerts were triggered on average by each IDS*	15,634.5
Alerts Suricata (avg)	*Test will show how many alerts were triggered on average by each IDS*	96,852
Suricata- time (average)	*This will provide the time taken by each IDS on average while testing the PCAP files*	24.2185 (s)
Snort- time (average)	*This will provide the time taken by each IDS on average while testing the PCAP files*	41.467 (s)
False positives-Suricata	*This test will show how many false positives we are getting when compared to right false (they shouldn't have passed through)*	3 (alerts)
False negatives-Suricata	*This test will show how many false negatives we are getting*	23091 (alerts)

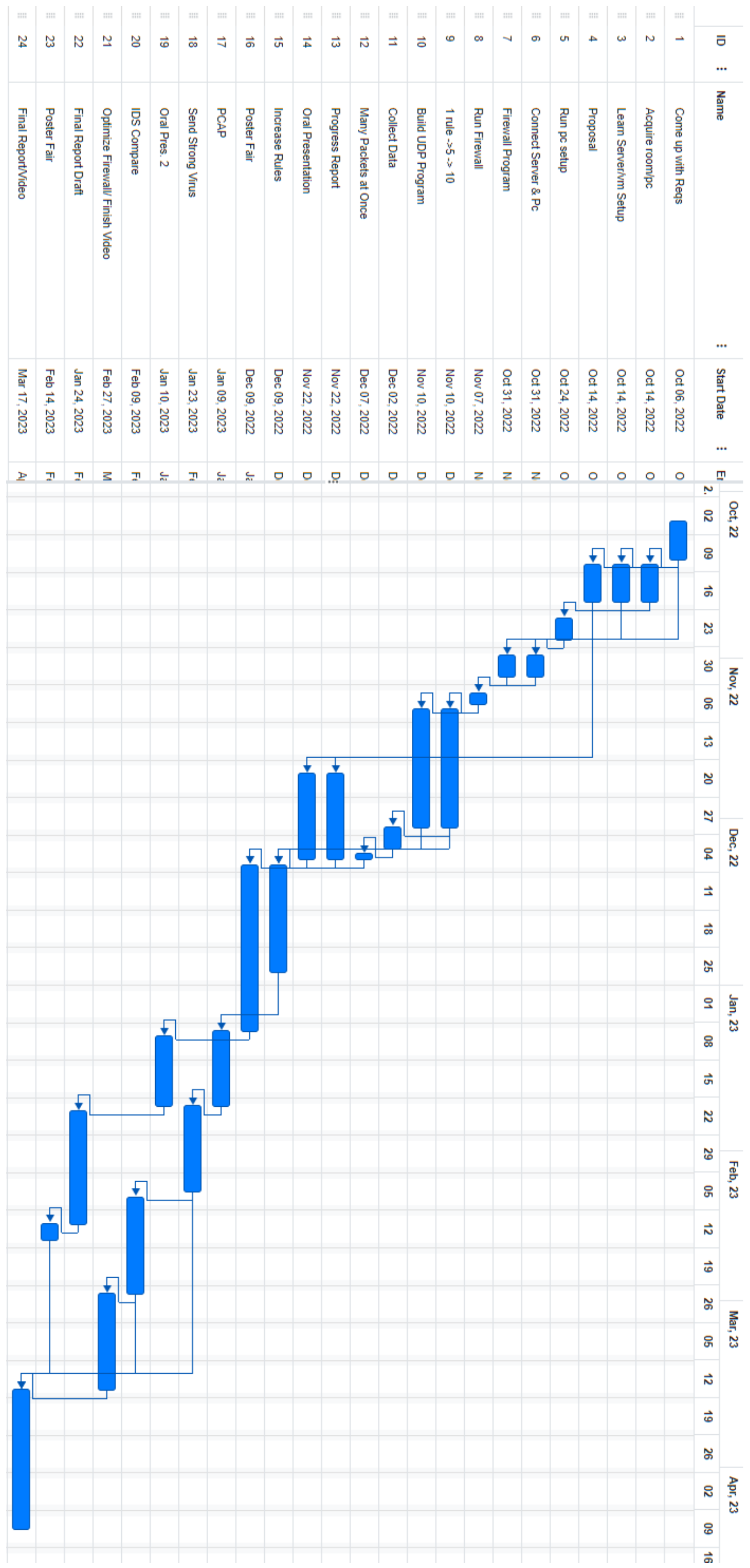
	when compared to actual false data (they should have passed through)	
False positives-Snort	*This test will show how many false positives we are getting when compared to right false (they shouldn't have passed through)*	6 (alerts)
False negatives-Snort	*This test will show how many false negatives we are getting when compared to actual false data (they should have passed through)*	3841 (alerts)

Scheduling

Schedule Network Diagram



Gantt Chart



ID	Name	Start Date	E
1	Come up with Reqs	Oct 06, 2022	O
2	Acquire room/pc	Oct 14, 2022	O
3	Learn Server/vm Setup	Oct 14, 2022	O
4	Proposal	Oct 14, 2022	O
5	Run pc setup	Oct 24, 2022	O
6	Connect Server & Pc	Oct 31, 2022	N
7	Firewall Program	Oct 31, 2022	N
8	Run Firewall	Nov 07, 2022	N
9	1 rule ->5 -> 10	Nov 10, 2022	D
10	Build UDP Program	Nov 10, 2022	D
11	Collect Data	Dec 02, 2022	D
12	Many Packets at Once	Dec 07, 2022	D
13	Progress Report	Nov 22, 2022	D
14	Oral Presentation	Nov 22, 2022	D
15	Increase Rules	Dec 09, 2022	D
16	Poster Fair	Dec 09, 2022	Ja
17	PCAP	Jan 09, 2023	Ja
18	Send Strong Virus	Jan 23, 2023	Fe
19	Oral Pres. 2	Jan 10, 2023	Ja
20	IDS Compare	Feb 09, 2023	Fe
21	Optimize Firewall/ Finish Video	Feb 27, 2023	M
22	Final Report Draft	Jan 24, 2023	Fe
23	Poster Fair	Feb 14, 2023	Fe
24	Final Report/Video	Mar 17, 2023	Ap

Assignment Matrix

This following table will represent who is responsible for what and who is the approval that will check over the work and make sure it is good:

	Activity Name	**Responsible**	**Approval**
1	Group comes up with requirements	Both	-
2	Acquire Room/Pc	Ahmad	Ousama
3	Run the pc and set up	Ahmad	Ousama
4	Learn server/VM set up	Ahmad	Ousama
5	Set up the Pc with the server/VM	Both	Ousama
6	Proposal write up	Both	-
7	Figure out how to use the firewall open source program	Both	-
8	Run firewall on PC and observe incoming traffic	Both	Ousama
9	Start using 1 rule into 5 rules into 10	Both	Ahmad
10	Collect data on those 10 rules/research about Suricata and Snort rules	Both	-
11	Progress Report	Both	-
12	Poster fair demo	Both	-
13	Oral Presentation	Both	-
14	Increase amount of rules until satisfied	Both	-
15	Start prepping a PCAP test/challenge and try it out.	Both	-
16	Compare Snort vs Suricata	Both	-

17	Send a strong virus and see how the rules react to it	Both	-
18	Final report Draft	Ousama	Ahmad
19	Poster Fair	Both	-
20	Finish video on project	Ousama	Ahmad
21	Oral presentation 2	Both	-
22	Final Report including video	Both	-

Results

The dataset provided below is based on the five categories chosen to show the performance of each IDS. **Note the blue results are Suricata and the red results are Snort's.** The tables in results as seen below were divided into five days which were Monday through Friday and two sets which were in and out. Each day and each set were tested individually. As mentioned earlier the results for the five categories are shown in terms of percentage for the CPU and memory power usage, numbers for the packages sent and Alerts generated and finally the time taken to execute the command line in seconds. The days below are divided by a horizontal line to differentiate them. The table below will first provide the day, then it will provide Suricata's results in blue and then Snort's results which will be in red. Each table's first row will be the in and out indicating which TCP dumps. The following row will be the classes for each IDS. Below each day there will be an analysis sentence that would help explain the tables and the values below.

Monday tests

	IN		OUT	
Rules	Standard	Added	Standard	Added
Cpu %	99.7	99.7	99.3	99.7
Mem %	23.4	22.8	20.1	24.3
Packets	2,291,319	2,291,319	1.376.598	1.376.598
Alerts	44,044	44033	41,677	41,674
Time (s)	24.217	23.025	9.348	9.601

	IN			OUT		
Rules	Com	Reg	Sub	Com	Reg	Sub
cpu%	96	97.0	99.7	90	98.7	98.7
mem%	4	11.5	10.5	3.5	10.5	11.6
Packets	2,291,319	2,291,319	2,291,319	1,376,598	1,376,598	1,376,598
alerts	3,621	20,683	15,788	14,337	24,737	24,738
Time (s)	10.4	27.7	26.9	7.4	18.6	19.3

The tables show the following: the CPU usage is almost the same between each IDS, but snort used way less Memory and took a bit longer time than Suricata. The alerts generated by Suricata are way more than Snorts. The following tables will show similar results but tiny differences which will be pointed out.

Tuesday tests

	IN		OUT	
Rules	Standard	Added	Standard	Added
Cpu %	93.1	99.7	99	99.3
Mem %	23.1	25.1	13.8	11.3
Packets	3,404,824	3,404,824	2,558,481	2,558,481
Alerts	38,159	38,155	50,168	51,161
Time (s)	24.947	26.273	14.913	15.044

	IN			OUT		
Rules	Com	Reg	Sub	Com	Reg	Sub
cpu%	99.0	99.7	98.0	99.0	98.7	99.0

mem%	2.7	10.7	10.7	2.6	10.5	10.6
Packets	3,404,824	3,404,824	3,404,824	2,558,481	2,558,481	2,558,481
alerts	3,517	16,173	16,172	10,480	13,543	13,544
Time (s)	12.4	31.1	32.2	9.9	25.7	26.8

This part we can the alerts and time difference by snort and Suricata is getting larger**.**

Wednesday tests

	IN		OUT	
Rules	Standard	Added	Standard	Added
Cpu %	99.3	99.3	99.7	99.7
Mem %	7.3	24.4	23	24.3
Packets	2,087,942	2,087,942	1,385,130	1,385,130
Alerts	44,339	44,342	71,005	71,013
Time (s)	25.088	23.162	11.416	11.989

	IN			OUT		
Rules	Com	Reg	Sub	Com	Reg	Sub
cpu%	94	95.3	99.7	95.7	97.3	99.0
mem%	2.6	10.7	10.7	2.4	10.5	10.6
Packets	2,087,942	2,087,942	2,087,942	1,385,130	1,385,130	1,385,130
alerts	3,487	26,141	26,149	3,416	8,213	8,214
Time (s)	13.9	35.5	33.5	10.4	26.2	25.9

The tables show the alerts generated by Suricata are way more than snorts and the time gap is almost similar.

Thursday Tests

	IN		OUT	
Rules	Standard	Added	Standard	Added
Cpu %	99.2	99.3	99.7	99.7
Mem %	22.7	15.4	23.1	22.7
Packets	3,201,381	3,201,381	2,308,273	2,308,273
Alerts	69,622	69,613	74,166	74,158
Time (s)	37.038	39.121	21.493	19.596

	IN			OUT		
Rules	Com	Reg	Sub	Com	Reg	Sub
cpu%	98.0	95.3	97.0	98.6	99.0	99.7
mem%	3.9	10.7	12.0	2.4	10.4	10.5
Packets	3,201,381	3,201,381	3,201,381	2,308,273	2,308,273	2,308,273
alerts	9,324	39,256	72,629	9,680	14,797	14,795
Time (s)	21.0	61.4	63.7	17.0	48.9	49.9

Similar situation seen here but snort's sub class in the inside dump beats Suricata's alerts and this shows the improvements as it detects the cyber-attacks better.

.

Friday tests

	IN		OUT	
Rules	Standard	Added	Standard	Added
Cpu %	99.3	98.3	99.3	99.3
Mem %	23.6	12.8	22.8	25.3
Packets	3,393,918	3,393,918	595,903	595,903
Alerts	170,601	170,596	23,106	23,105
Time (s)	43.081	43.519	5.387	4.887

	IN			OUT		
Rules	Com	Reg	Sub	Com	Reg	Sub
cpu%	95.7	99.1	98.7	96.3	99.7	98.3
mem%	2.6	10.5	10.5	2.4	10.5	10.6
Packets	3,393,918	3,393,918	3,393,918	595,903	595,903	595,903
alerts	5,259	38,496	38,495	2,542	4,507	4,508
Time (s)	32.5	92.4	91.8	5.0	13.3	13.8

Finally, when looking at the Friday's tables we see similar results but in regards to alerts generated we see that Suricata generated a lot more while snort did not

Charts:

As mentioned earlier all the data provided in the charts are taken from Friday TCP dumps. The charts show the IDS Suricata and snort next to each other to make it easier to visualize the difference between them. As seen here the CPU power usage by both is almost the same and can see that the community rules- snort used the least amount of CPU percentage. Next, we can see that memory utilization is excessively used by Suricata while by snort not as much, again if we look at snort community rules it uses the least amount of memory. Moving to alerts we can establish that Suricata generated way more than all of the snort's classes. In addition, if we look at the time taken by Suricata it is much less than by snort. We can see that although Suricata uses way more hardware it provides more information than Snort. Proceeding to the false positives and false negatives, Suricata generated less false positives than Snort and this shows that Suricata allowed less viruses than Snort did. This shows that both IDS's blocked as many attacks as possible and only a few went through, which were between 3 to 4 attacks. Looking at false negatives it can be seen that Suricata detected way more than snort. This puts Suricata in a bad position as to why these packets were detected. We have to take into consideration that Suricata will detect any suspicious packets. The data provided by DARPA was extremely limited when it came to false negatives. These two tests showed what each IDS is detecting.

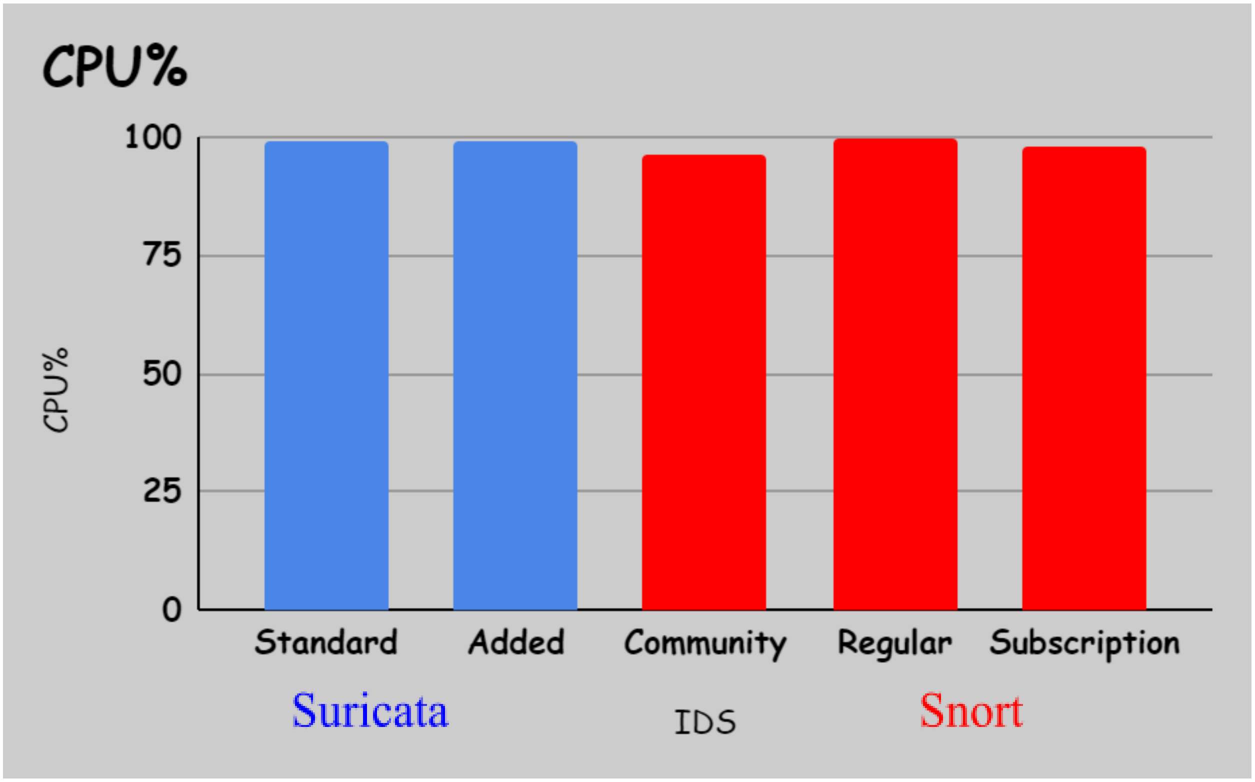


Figure 14: CPU utilization used by each class of each ids

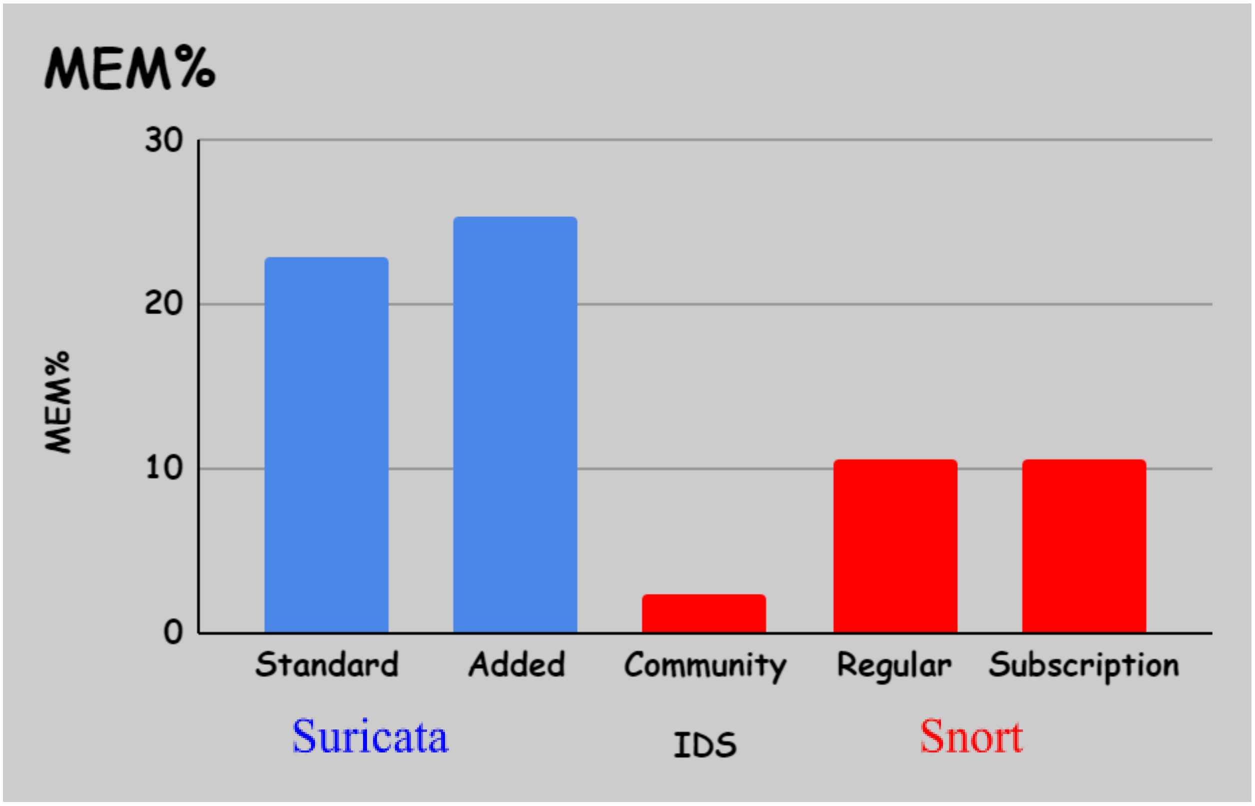


Figure 15: Mem utilization used by each class of each IDS

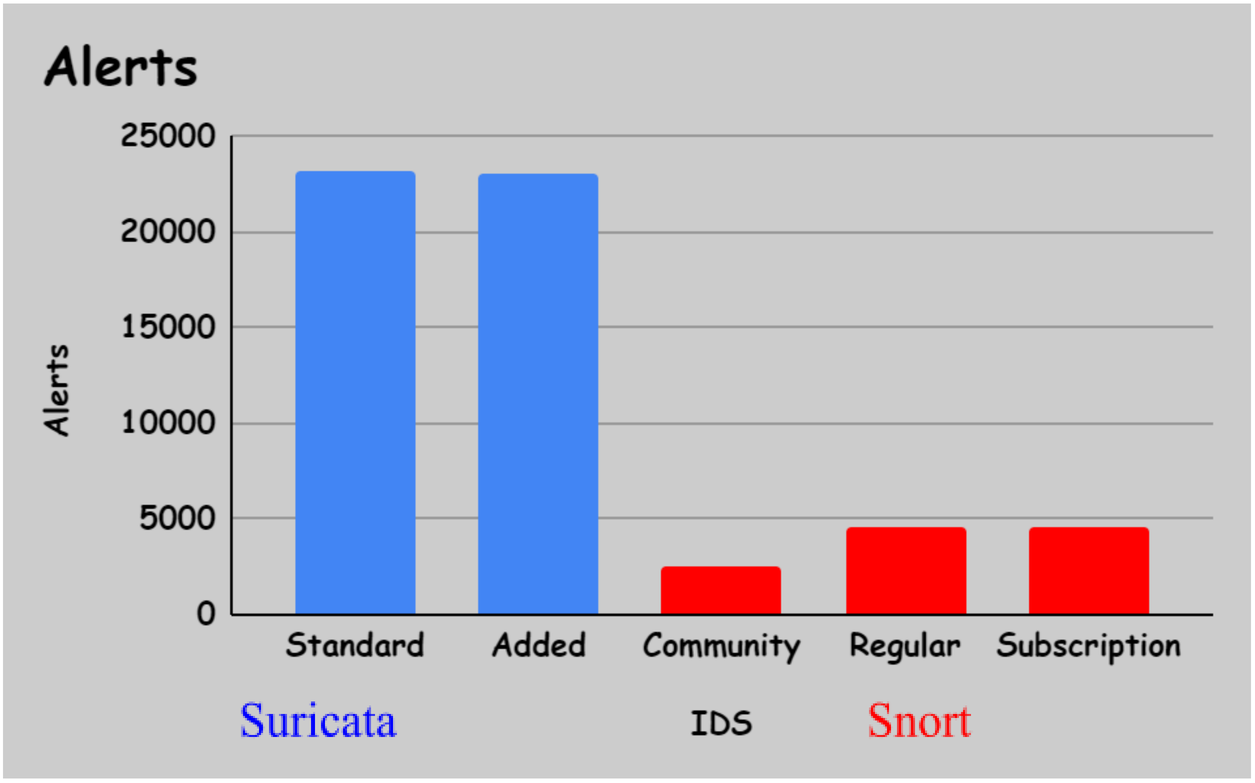


Figure 16: Alerts comparison between the difference sets

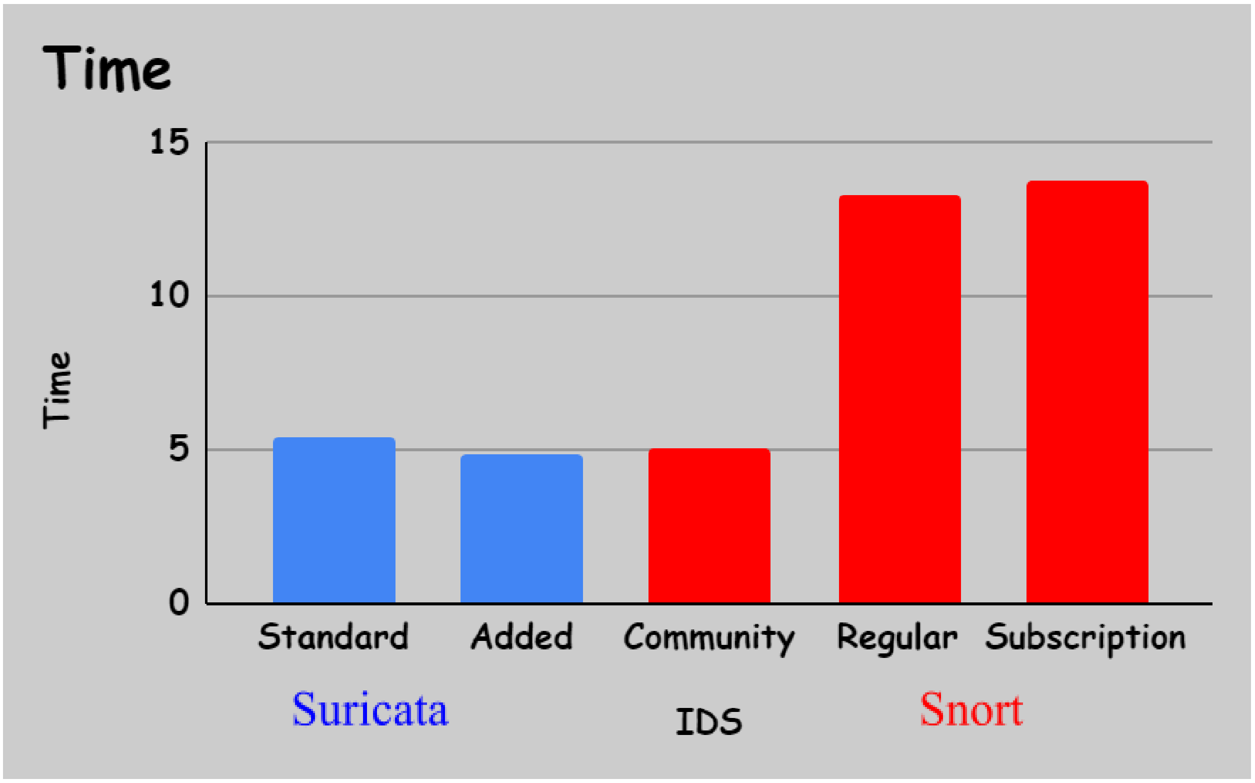


Figure 17: Time took comparison between the difference sets

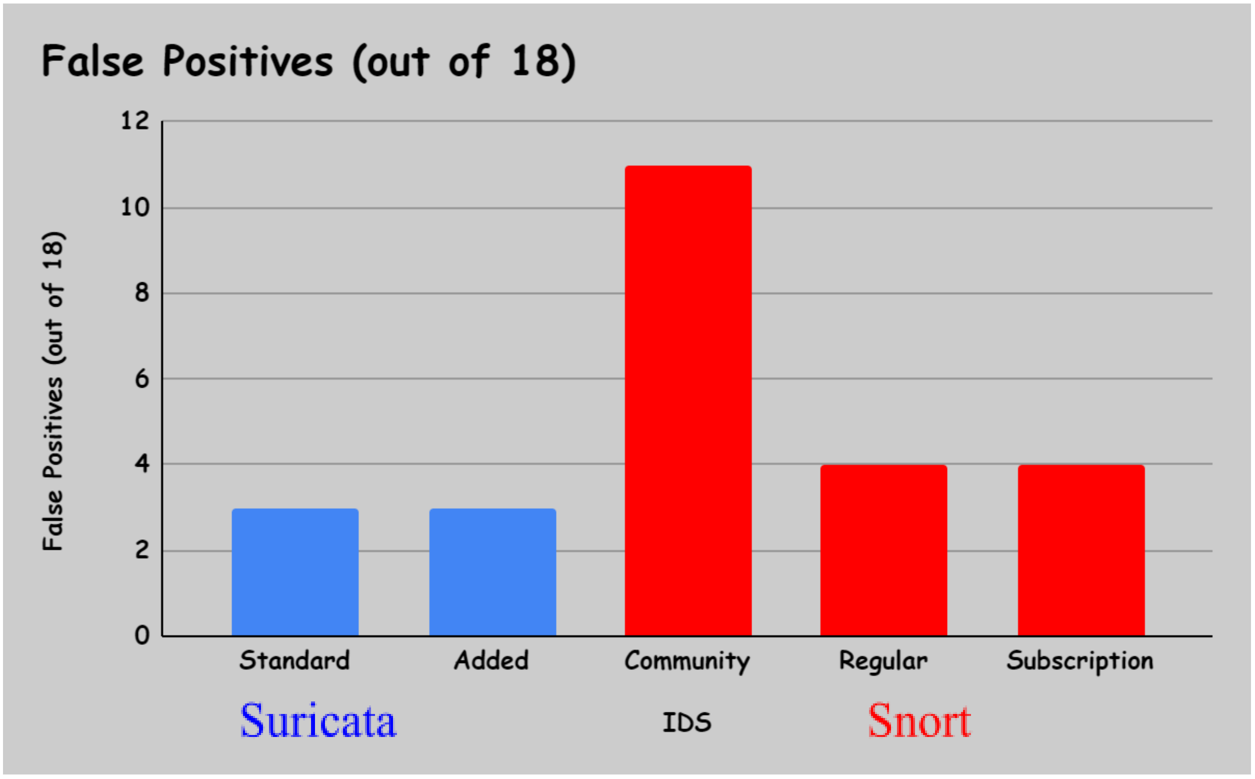


Figure 18: False Positives for each set of rules

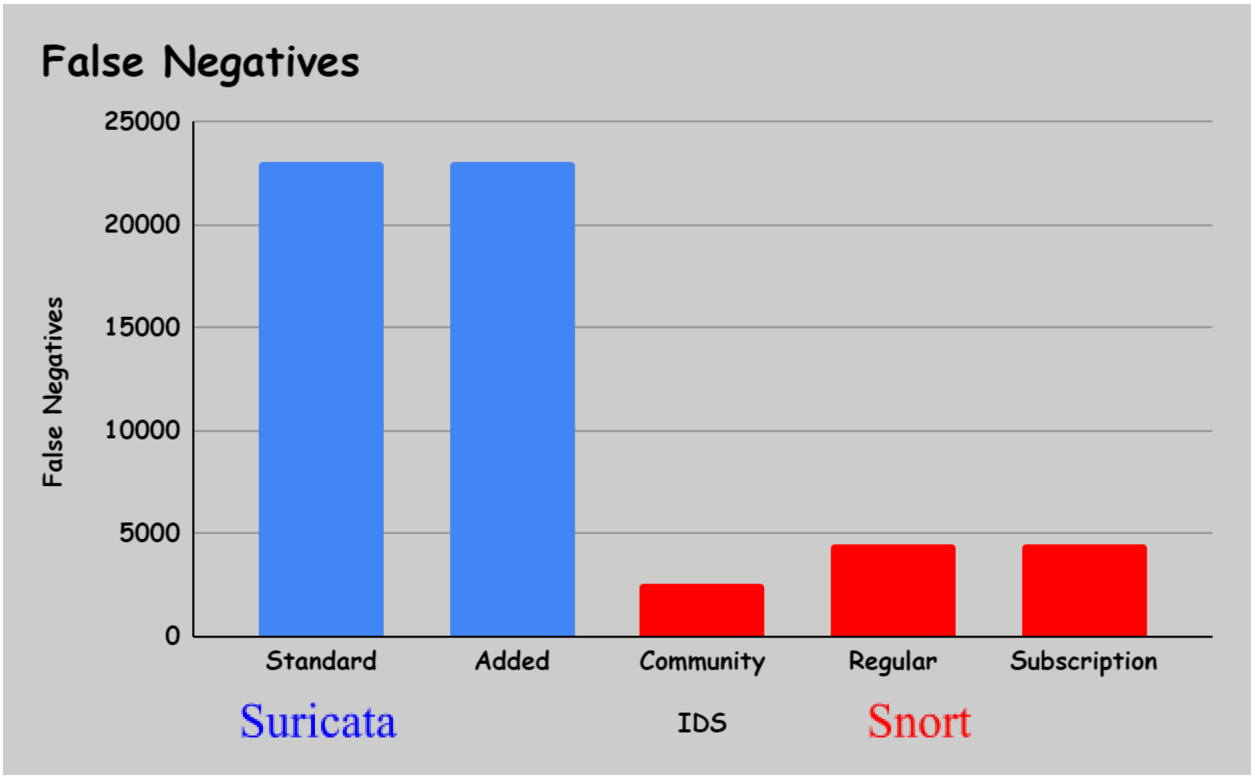


Figure 19: False Negatives for each set of rules

Analysis

First analyzing the tables from the results section. It is clear that the CPU% being used is 90% and higher among all of the tests. This proves that both Snort and Suricata decided to use as much power as the system allowed them to run as quickly as possible. This could be an issue when trying to download these settings and PFSense into a router since the router would have a much weaker system. Thankfully if we were to use this system on a home router it would not need that much power since it would only need to run those tests on a small amount of packets in comparison. Most commercial routers have very weak CPUs compared to the ones used in the tests done. This means that although we meet our requirements, we need to find a way to run those tests on a very weak CPU and check those results once again (Next Steps in conclusion touches on this).

The memory utilization was surprisingly high for a 4gb memory cap, especially that of Suricata compared to Snort. The results meant that Suricata roughly used around 20%-25% which is a whole 1gb by itself without even the firewall working (this applies to CPU usage as well). Most Memories for commercial routers are around the 100mb mark. This means that they can only handle 10% of the amount that was used in this test [11]. More on this in conclusion at the end.

The number of generated alerts was significantly higher with Suricata than Snort. This Further proves that Suricata is a lot more secure since it checks every single one of the packets sent its way while also stopping anything that is deemed suspicious. From these results we are assuming that snort does not stop every single suspicious packet but instead focus on the ones that are actual threats. This means that Suricata is overall much more secure while Snort seems to be more hardware friendly and optimized. Time consumed to perform the analysis by either IDS seemed to be fast. Regular and Subscript packages seem to take more time than the community set, or the packages provided by Suricata.

False positives are when a threat that is known and should be detected passes through the firewall or IDS into the system. As seen from the charts above, the False positives for both Suricata and the upgraded packages from Snort are doing very well. Whilst the community package set from Snort struggles to detect a lot of these main threats. False negatives were a very disappointing stat (*). False negatives is how many none threats were detected by the IDS or firewall. In our case since the DARPA study only included the main threats they were trying to detect, not the entire list of alerts, it made our False negatives number for each package very high. It should not have been this high if the list of alerts were to be included in their research website. This made this last statistic very inaccurate thus resulting in an error in our initial testing table (next steps in conclusion talks about how to fix this in future projects). By the end of the project period we have found a reliable website that does testing (and is new) with proper sets of PCAP files and alert lists.

Based on the results we acquired (from the Results section), it is very clear that Suricata has the more secure IDS while Snort is the more system resource saver. On average Suricata had more alerts detected in comparison to Snort while on average Snort used less memory power than Suricata. As far as comparison between the packets it seems like between standard and added there is not much of a difference, just more optimization in the added packets. While on the Snort side a huge difference is noticed between the community set of rules and the other two. The regular and subscription set of rules seem to have similar results but subscription being a little more optimized than the regular set (similar to Suricata's standard vs added).

Conclusion

The goal of this project was to find the best way to build a firewall and the most effective one as well, multiple tests were done to provide accurate results. As the results provided above using a sample from our tests, we saw that the best option as an IDS would be snort community rules. When testing each product, we ran each IDS for approximately a min. When studying the results, we saw that the Snort Community rules were the best in terms of CPU power and memory utilization. It used the least amount of CPU and memory percentage. Although it did not detect as many packets as the Suricata or Snort upgraded options, we still took into consideration that we were testing it with a lot of packets which were being sent at a time; each PCAP file includes one million to three million packets. When facing a real-world problem an attacker might send a couple packets that could reach a maximum of a couple hundred at a time. We looked at this option also because you want the device that has your firewall to last as long as possible while still protecting you, you will not require expensive hardware, a normal device will suffice. If you require more protection you can upgrade to the Snort Registered rules; it is free as well but will use a bit more hardware. Although the results above showed that Suricata detected way more alerts, it is still an overkill to use, since it kills the hardware of the device. Suricata is better used in cyber companies or any company that requires high level of security. This is because Suricata detects anything that will seem suspicious. It can still be used as an IDS for home, but it will cost more than Snort to contain and maintain, so it can work well over time and last.

Additionally, looking back at the starting point of this project the team learned a lot of innovative ideas, concepts, and terms. In this project we dealt with an open-source firewall which we never dealt with, and the project taught us how to download it, set it up and how to use it for our needs. Using the open-source firewall was a really enjoyable experience and it showed us how to manipulate settings to make it convenient for us. One of the earliest things that we learned was the way to block Facebook and it was really fun. But the most tiring part of using the open-source firewall was setting it up as it needed a lot of time. Next would be the term PCAP files. We used a lot of PCAP files in our project and dealing with them was really exciting as they simulate real life attacks. Also, when dealing with them we had to be incredibly careful as any wrong step could corrupt our devices. The PCAP files would not be limited to one type of cyber-attacks, some would have phishing attacks, some would have malwares, and some would have spoofing attacks and many more. The PCAP files that we used in this project were tested by DARPA. This was another impressive part of this project, we got to compare our data against DARPA's results. Further, we got the chance to use the IDS (intrusion detection systems). Using the IDS was an incredibly beautiful experience as we got to use them in our project, we were using the rules as our base rules in our firewall set-up. We learned the command lines that ran them, and we also learned how to analyze the results as there were a lot of alerts generated. For example, to differentiate diverse types of attacks we had to look at the name of the attack, time sent, Ip sent from and the Ip that received it. The whole project was a new and exciting experience, and we learned a lot from it, especially because it had a lot of cyber security concepts that will stick with us forever. The project taught us how to do the appropriate research and how to analyze data. The final key it gave us is how to be patient and focus until you reach the end and never give up no matter how many setbacks a person may get.

The next step for us would be to use this research and work and summit all up together to create a device to be sold. This device can be something like a chip or a memory stick. Inside this device we would have a batch file that can do all the steps of installation and setting up automatically. We can also have these products preloaded up with the IDS package of choice to the customer. This product is very niche yet very important since a lot of attacks are constantly happening. We hope our experience and the stuff we learned from this project make the world a safer place. For future Research for this project the next steps would be to test this project using a proper router set up. The weaker CPU and memory could result in a totally different number causing the decision of what's better to be changed. As for the False positives the next steps would be to use a proper data set which is provided by the UNB (university of new Brunswick). They got many data sets that satisfied many different tests. They also provide full alert numbers as well as analysis data from research they have done. Very useful data to be used for the future of this project.

Security is very important and is not an easy task to be fully secure. Hackers and cyber attacks evolve everyday and so must we. Taking the first step by protecting ourselves using a firewall is only the start.

References

[1] *1999 DARPA Intrusion Detection Evaluation Dataset*. MIT Lincoln Laboratory. (n.d.). Retrieved April 12, 2023, from https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset

[2] *How many cyber attacks happen per day in 2023?* Techjury. (n.d.). Retrieved April 12, 2023, from https://techjury.net/blog/how-many-cyber-attacks-per-day/

[3] Cisco. (2023, February 10). *What is a Firewall?* Cisco. Retrieved April 12, 2023, from https://www.cisco.com/c/en_ca/products/security/firewalls/what-is-a-firewall.html

[4] Amy Larsen DeCarlo; Robert G. Ferrell. (2021, January 19). *The 5 different types of firewalls explained*. Security. Retrieved April 12, 2023, from https://www.techtarget.com/searchsecurity/feature/The-five-different-types-of-firewalls

[5] . Failover, High Availability, Redundancy, Clustering and RAID differences tutorial. (n.d.). Retrieved April 12, 2023, from http://www.internet-computer-security.com/Firewall/Failover.html

[6] *Structured Firewall Design - CS.UTEXAS.EDU*. (n.d.). Retrieved April 12, 2023, from https://www.cs.utexas.edu/~gouda/cs386m/fdd.pdf

[7] Marketing, H. (2022, August 18). *Understanding the 5 types of intrusion detection systems*. Helixstorm. Retrieved April 12, 2023, from https://www.helixstorm.com/blog/types-of-intrusion-detection-systems/

[8] *Oracle VM*. Oracle VM Overview. (n.d.). Retrieved April 12, 2023, from https://www.oracle.com/virtualization/technologies/vm/

[9] *Choosing router operating system pfSense VS OPNSense vs openwrt*. TekLager. (n.d.). Retrieved April 12, 2023, from https://teklager.se/en/knowledge-base/choosing-router-operating-system-pfsense-vs-opnsense-vs-openwrt/#:~:text=OPNsense%20has%20a%20nicer%20user,if%20you%20already%20know%20it.

[10] Keary, T. (2022, November 2). *PCAP: Packet capture, what it is & what you need to know*. Comparitech. Retrieved April 12, 2023, from https://www.comparitech.com/net-admin/pcap-guide/

[11] *Router vs PC performance?* Reddit. (n.d.). Retrieved April 12, 2023, from https://www.reddit.com/r/homelab/comments/lm9zl9/router_vs_pc_performance/