Table of Content

Intrusion Detection System (IDS)

An Intrusion Detection System is a security tool used to guard computer networks. It does this by monitoring the network traffic or system activities to spot anything suspicious, like hacking attempts, malware, unauthorized attempts or anything out of normality. When it does detect anything odd, it alerts the system admin so proper action can be taken.

Type of Intrusion Detection System

There are 2 main types of IDS - Network Based IDS and Host Based IDS.

Network-Based IDS

Network Based IDS basically watches the network traffic to detect unusual patterns in data packets for signs of attacks like spike in traffic coming from a network which might indicate DOS attack.

Host-Based IDS

Host-Based IDS on the other hand is specific to a single device like laptop or server. It checks things like system logs, file changes or weird login attempts.

Mode of Detection

Intrusion Detection System uses 2 methods to spot something/anything fishy.

Signature Based Detection

The IDS compares network traffic or system activity to a list of known attack “signature”, something like a pattern of bad behavior, like a specific malware code. It matches a system activity to a database to check if something similar can be found there.

The method is great for catching known attacks but what happens in the case of new attacks that are not in the database. This is the major flaw of Signature Based Detection.

Anomaly Based Detection

In this method detection, your system must have studied what’s “normal” over a period of time and when it sees anything weird “anormal” **like a spike in network traffic at 3 am,** it flags it and alerts the system admin. This is great for unknown and new threats but comes with **False Positive** sometimes.

Some advanced IDS systems also use Behavior-Based or Heuristic-Based detection, which look at how programs or users behave over time. It's just a modernized/upgraded version to Anomaly-Based Detection.

Why it is Important

- IDS catches threats early - it can spot an attack before they do serious damage (stealing data or crashing the system).
- Improve Security - It helps you understand what's happening in your system, so you can fix weak spots.
- Compliance - many industries (like finance and healthcare) acquire IDS to meet security regulations

Example of IDS

A popular example I will be looking at in my project is **Snort** - A free, open-source Network Based IDS. Its widely used and has a big community. It can be set up to monitor network traffic and practice analyzing alerts.

Another one is **Suricata** - just like snort with some extra features like multi-threading and faster performance.

False Positive and False Negative

I mentioned earlier when talking about Anomaly Based Detection that it is great for new and unknown threat but might generate innocent oddity - False Positive
.

False Positive

When an Intrusion Detection System flags something as a threat but it is harmless. For example, your IDS alerts you about a ‘suspicious’ login attempt, but it is just you logging in from a new device. Too many false positives can be overwhelming and make it hard to spot real threats. One way to fix this would be to fine tune IDS rules to reduce unnecessary alerts (like whitelisting trusted activities).

False Negative

This is when an IDS misses a real threat thinking its a normal activity like a hacker sneaking into your network but IDS sees it as regular traffic. This is dangerous because an actual attack is going unnoticed. Anomaly based detection is best to handle this or update signature regularly to feature new rules and catch new threats

Snort

Snort is an open-source Network Intrusion Detection System created by Sourcefire founder and former CTO Martin Aesch. Cisco now develops and maintains Snort. It is referred to as a packet sniffer that monitors network traffic, sanitizing each packet closely to detect dangerous payload or suspicious anomalies.

How does it work

Snort is based on Library Packet Capture (Libpcap). Libpcap is a tool that is widely used in TCP/IP address traffic sniffers, content searching and analysis for packet logging, real-time packet analysis and content matching.

Users can configure Snort to be a packet sniffer, packet logger - like TCP dump and Wireshark - or Network Intrusion Prevention System. IPS goes beyond monitoring and alerting when suspicious traffic is detected, it tries to take preventive action against fishy activities. It is often located between a company's firewall and the external network.

Snort has 3 main Operation Mode

- Packet Sniffing - Wireshark type of function; collects and display network traffic
- Packet logging - Collects and network traffic onto a file
- Network Intrusion Detection - analyze packets and matches traffic against signature (rules).

When Snort is functioning as IPS, ideally it should be placed between the switch and the Router. This means all traffic incoming and outgoing passes through Snort. This is a **choke** point. This means any communication happening in your network must meet Snort and the Action is taken based on specific predefined rules.

When Snort is in IDS mode, the system running a Snort is connected to the switch whether its LAN or virtualized environments. If you have more than one VM connected to the same type of network adapter, then they are connected through a virtual switch. So ideally, the actual server running Snorts connected or part of the network the VMs are connected to. They are going to be on the same subnet, but they are all connected to the switch. The snort is essentially monitoring all the traffic that is coming in and out of the devices on the network and based off that traffic, it tells you, based on the rules whether there is an intrusion, whether ping has been performed, whether there is any form of scan and so on.

Configuring Snort as an IPS is best done on a physical, non-virtualized network but IDS is fine in a Virtualized Environment (VM).

Rules

Predefined rules are at the core functionality of Snort. Without them, they don't function. Snort rules are very similar to a typical firewall rule whereby, they are used to match network activity against specific patterns or signature and consequently decide whether to send an alert or drop the traffic (IPS).

Snort Rules

Snort has 3 types of Rule/Rules sit:

- Community Rules; free rules created by Snort community
- Registered Rules: free rules created by Talos. To use them you must register for an account.
- Subscription Only Rules: These rules require an active paid subscription to access and use the rules.
- Or the last option which is available to everyone is to customize your own rules based on your requirement.

Snort Rules Syntax

Whenever you are writing Snort Rules yourself, its very important you understand the syntax. The rule is divided into 2 parts - Rule Header and Rule Options.

The Rules header has:

- ➔ Action: which can be Log, Alert, Drop and so on
- ➔ Protocol: ICMP, TCP, UDP or IP
- ➔ Source Address: The address of the where the traffic is coming from is mostly specified to be **Any** because confining the address to a particular range can allow traffic outside the range bypass the rule.
- ➔ Source Port: can be any port as well ideally or specified.
- ➔ Direction: denoted with a **hyphen** and **greater-than sign ->**
- ➔ Destination Address: mostly the address of the organization we configured Snort to monitor denoted with **$HOME_NET**
- ➔ Destination Port: The port the traffic we are trying to capture is running on.

The Rule Option, on the other hand has:

- ➔ Message: denoted with **Msg** stands for the alert message to display when s traffic is capture
- ➔ Signature ID: denoted by **SID** this is unique for each rule to distinct it from another. Ideally should start from 1000 and above when creating your own rules.
- ➔ Revision: denoted with rev 1; for the first rule under a specific rules meaning a rule can be revised to meet other requirement

A typical rule looks like this

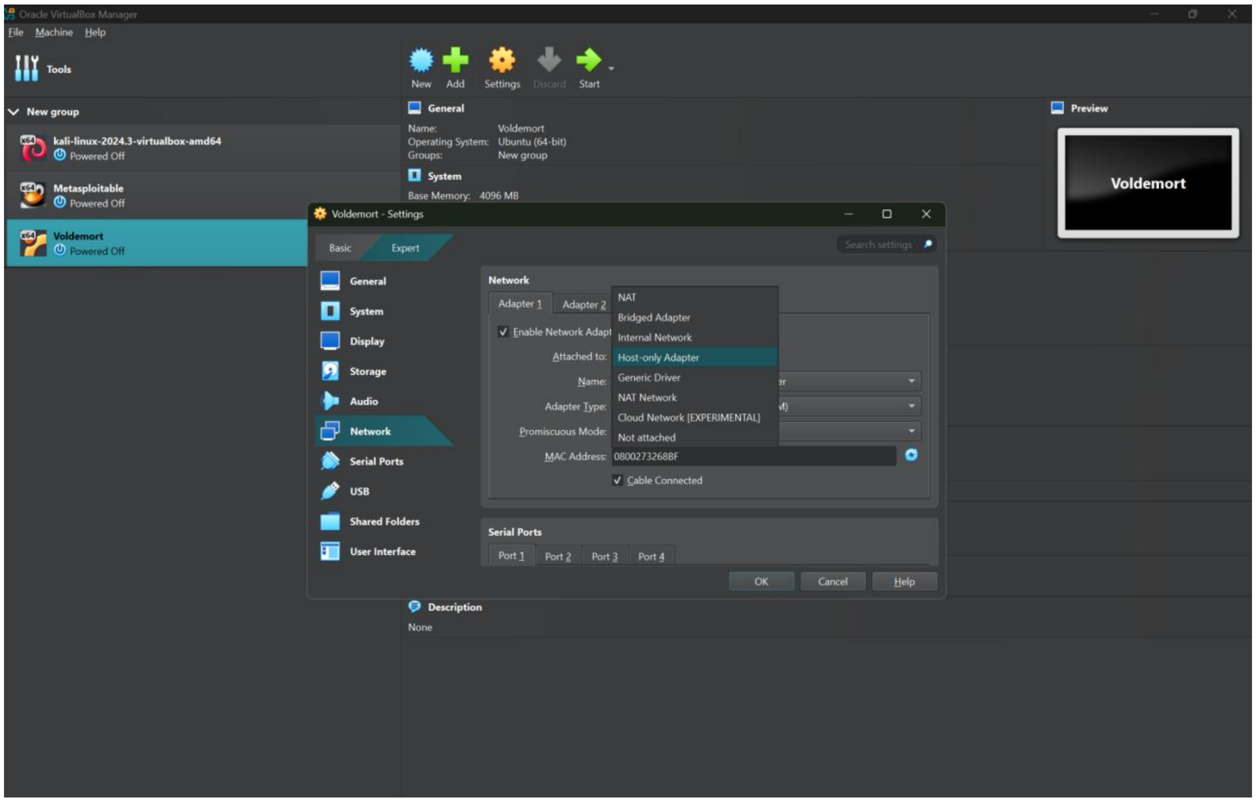
```
Alert Tcp Any Any -> $HOME_NET 21 (msg: “FTP Attempt Alert”;  sid: 100001; rev:1;)
```

Explanation
This rule is stating that any Tcp traffic coming from Any IP address on any port into our monitored network on port 21 <FTP> should display the alert FTP ATTEMPT ALERT and should carry the signature id of 100001 and the first of this rule.

Snort Setup and Lab Environment - VMs

- Kali Linux - To simulate External Network and Intrusion
- Ubuntu 25.04 (Voldemort) - To run snort and monitor Network Traffic
- Metasploitable 2 - Vulnerable Linux Server

The aim is to make all 3 VMs run on the same Network Adapter and be part of the same subnet. To this effect, all the VMs were attached to `Host-Only`, allowing Snort to monitor all lab traffic without affecting other VMs.



New group
kali-linux-2024.3-virtualbox-amd64 Powered Off
Metasploitable Powered Off
Voldemort Powered Off

General
Name: Voldemort
Operating System: Ubuntu (64-bit)
Groups: New group

System
Base Memory: 4096 MB

Voldemort - Settings
Network
✓ Enable Network Adapt
Attached to: Host-only Adapter
Name:
Adapter Type:
Promiscuous Mode:
MAC Address: 0800273268BF
✓ Cable Connected

Serial Ports

Description
None

Virtual box showing the VMs' network getting attached to Host-Only Adapter

Snort Installation on Ubuntu 24.04 VM

After completing the Installation of the Ubuntu VM, run the command

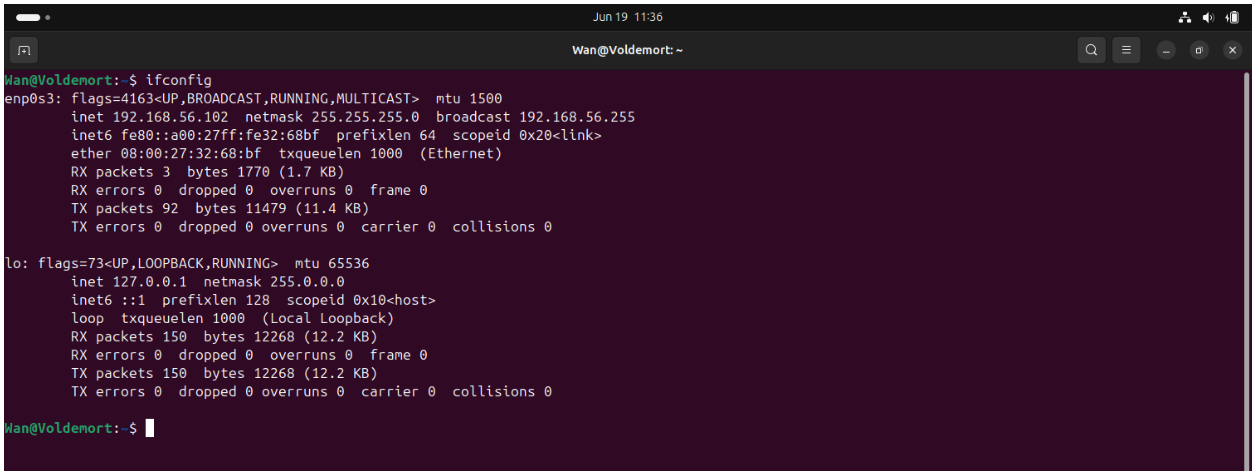
```
sudo apt update
sudo apt upgrade
```

This ensures the installation of the latest version of the packages from updated repositories.

Then use the command bellow to install Snort

```
sudo apt install snort -y
```

During the set up, you will be prompted to specify the address range for the local network that you are trying to monitor. You can confirm this with the `ifconfig` command in the terminal.

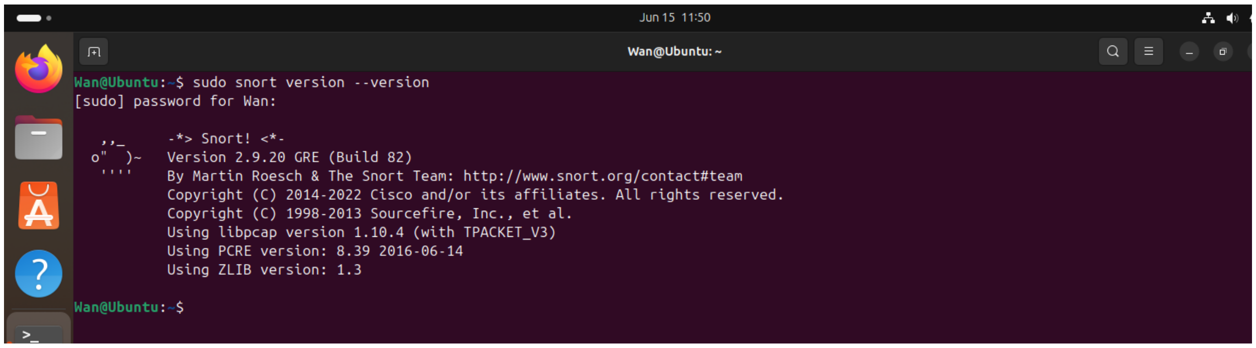


```
Wan@Voldemort:~$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.56.102  netmask 255.255.255.0  broadcast 192.168.56.255
        inet6 fe80::a00:27ff:fe32:68bf  prefixlen 64  scopeid 0x20<link>
        ether 08:00:27:32:68:bf  txqueuelen 1000  (Ethernet)
        RX packets 3  bytes 1770 (1.7 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 92  bytes 11479 (11.4 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 150  bytes 12268 (12.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 150  bytes 12268 (12.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

Wan@Voldemort:~$
```

Ubuntu Terminal showing the Network Info of the VM

Snort would/should be successfully installed after this. To confirm, use the command `snort --version`

```
sudo snort --version
```



```
Wan@Ubuntu:~$ sudo snort version --version
[sudo] password for Wan:

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3

Wan@Ubuntu:~$
```

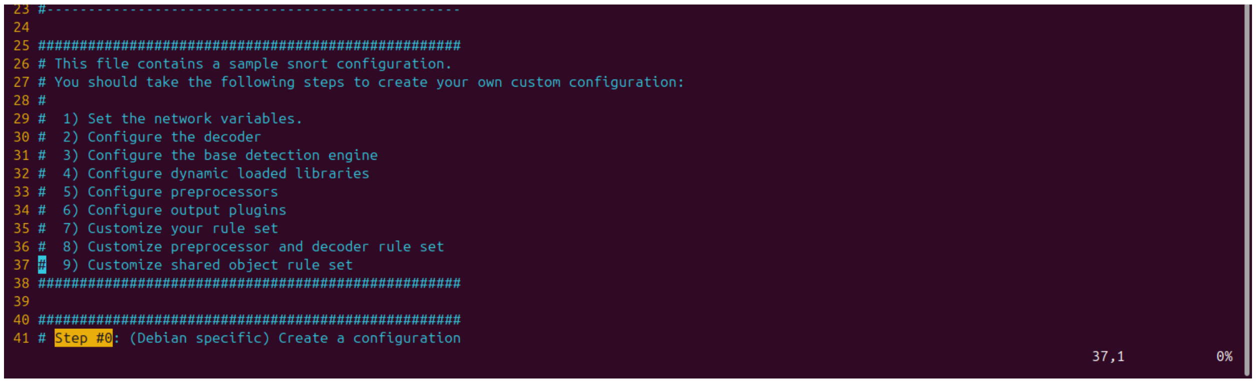
Ubuntu Terminal displaying the version of Snort on the VM

The next thing would be to list all the content of the Snort directory. This is the directory where all the Snort changes will be happening. The directory has a `rule` directory and a couple of files that are core to the functionality of Snort - including the `snort.conf`. This file is very important so as a beginner, it would be wise to make a copy so as to have a backup.

To view the content of the `snort.conf` file, you can use any text editor available in your VM to view and make changes to the file. `Vim` was used for this setup.

```
sudo vim /etc/snort/snort.conf
```

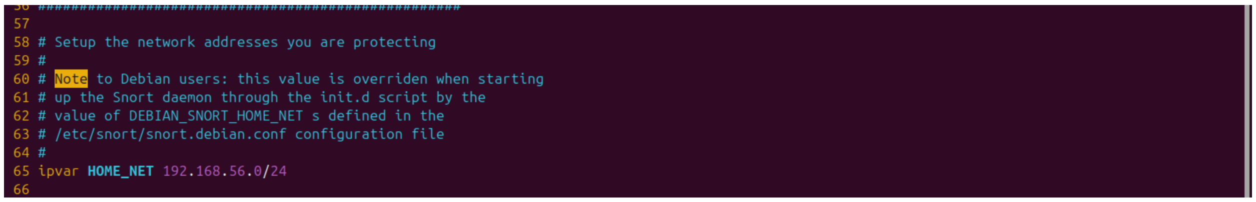
One key area that is noteworthy while configuring the `snort.conf` file is the `Step to Create your own custom configuration`. You should go through the steps one-by-one to make sure the steps are streamlined toward the proper functionality of Snort.



```
#--------------------------------------------------

###################################################
# This file contains a sample snort configuration.
# You should take the following steps to create your own custom configuration:
#
#  1) Set the network variables.
#  2) Configure the decoder
#  3) Configure the base detection engine
#  4) Configure dynamic loaded libraries
#  5) Configure preprocessors
#  6) Configure output plugins
#  7) Customize your rule set
#  8) Customize preprocessor and decoder rule set
#  9) Customize shared object rule set
###################################################

###################################################
# Step #0: (Debian specific) Create a configuration
                                                              37,1          0%
```

Text editor showing steps to create custom config

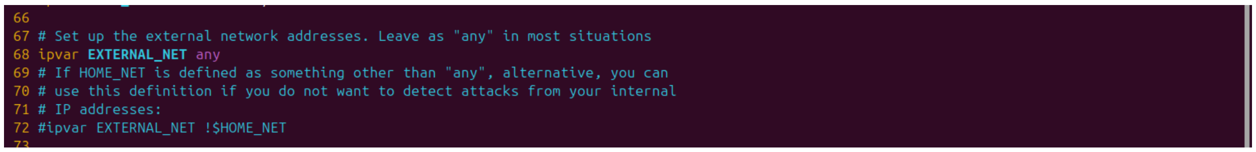
The first step is to set up the network variables. You want to scroll down to the section `Set the network variables` - this is basically where you set up the network addresses you would be protecting. The one specified during the Snort installation.



```
###################################################

# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overriden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.56.0/24

```

Text editor showing the HOME_NET

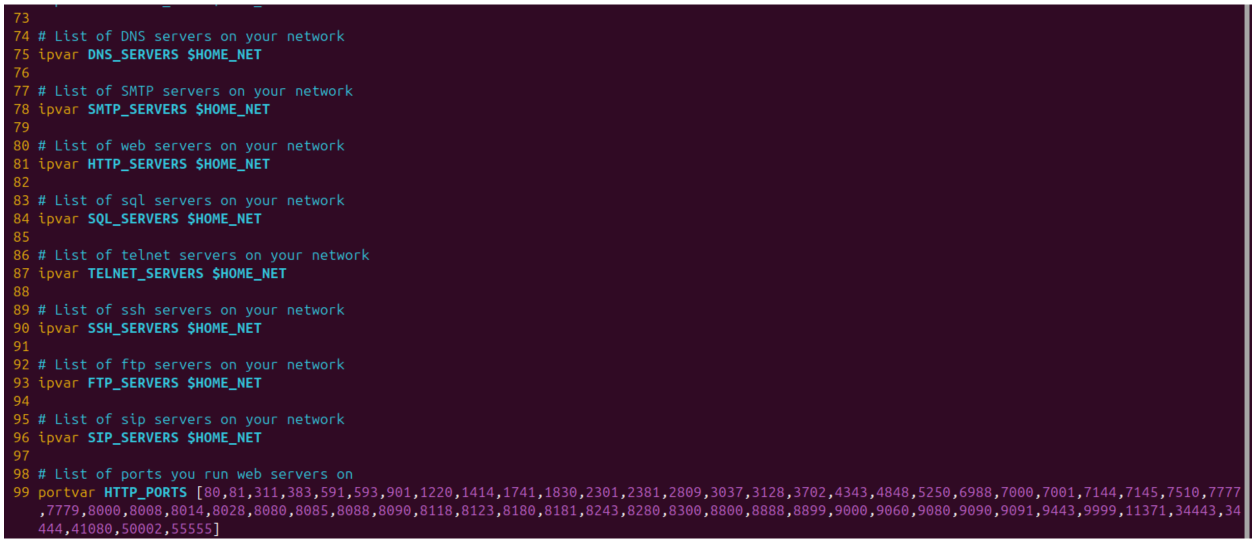
You would also be able to set up the external network addresses and would be advised to leave as it as `any` in most situations. This is because ideally you do not want to confine the external variable to a specific subnet. Attacks and intrusions can come from any network or IP address.



```

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET
```

Text editor showing the EXTERNAL_NET

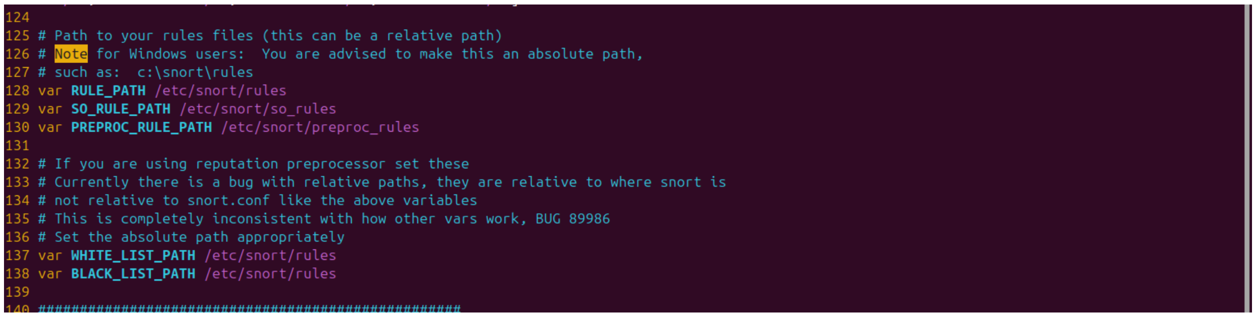
One of the great features of the snort configuration file is that it allows users to list, specifically, the hosts that might be part of the critical infrastructure of a network. It allows you to specify the actual list of DNS servers on your network as well as the list of ports you run web servers on



```

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,901,1220,1414,1741,1830,2301,2381,2809,3037,3128,3702,4343,4848,5250,6988,7000,7001,7144,7145,7510,7777
,7779,8000,8008,8014,8028,8080,8085,8088,8090,8118,8123,8180,8181,8243,8280,8300,8800,8888,8899,9000,9060,9080,9090,9091,9443,9999,11371,34443,34
444,41080,50002,55555]
```

Text editor showing list of DNS servers on HOME_NET and the port they are running on

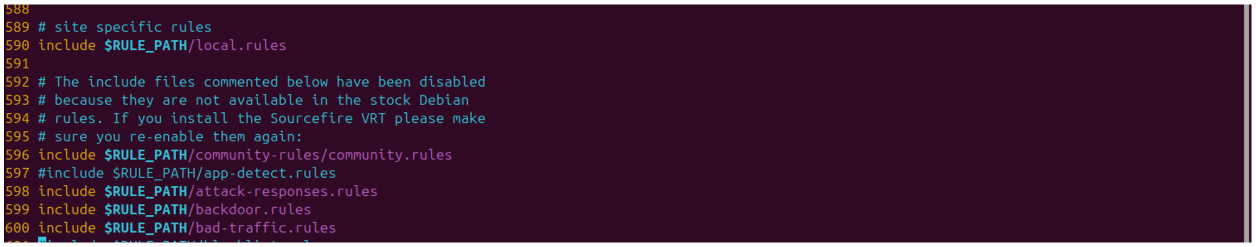
Another key area that is noteworthy is the `Path to your rules files`. This means that this configuration file will essentially pull in rules or load in rules within the rules path directory.



```

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var SO_RULE_PATH /etc/snort/so_rules
var PREPROC_RULE_PATH /etc/snort/preproc_rules

# If you are using reputation preprocessor set these
# Currently there is a bug with relative paths, they are relative to where snort is
# not relative to snort.conf like the above variables
# This is completely inconsistent with how other vars work, BUG 89986
# Set the absolute path appropriately
var WHITE_LIST_PATH /etc/snort/rules
var BLACK_LIST_PATH /etc/snort/rules

###################################################
```

Text editor showing the rules path

Another aspect of the file that is very important is the `Step #7: Customize your rule set` in the `snort.conf` file. There, we have `site specific rules` - which includes the actual rules that you will be creating yourself. This can be stored or written in a file called `local.rules` and that file is stored under `/etc/snort/RULE_PATH`. When you run snort, there are 2 categories of rules that will be loaded, the custom rules in the `local.rules` and the community rules, that will be mentioned later in the document.

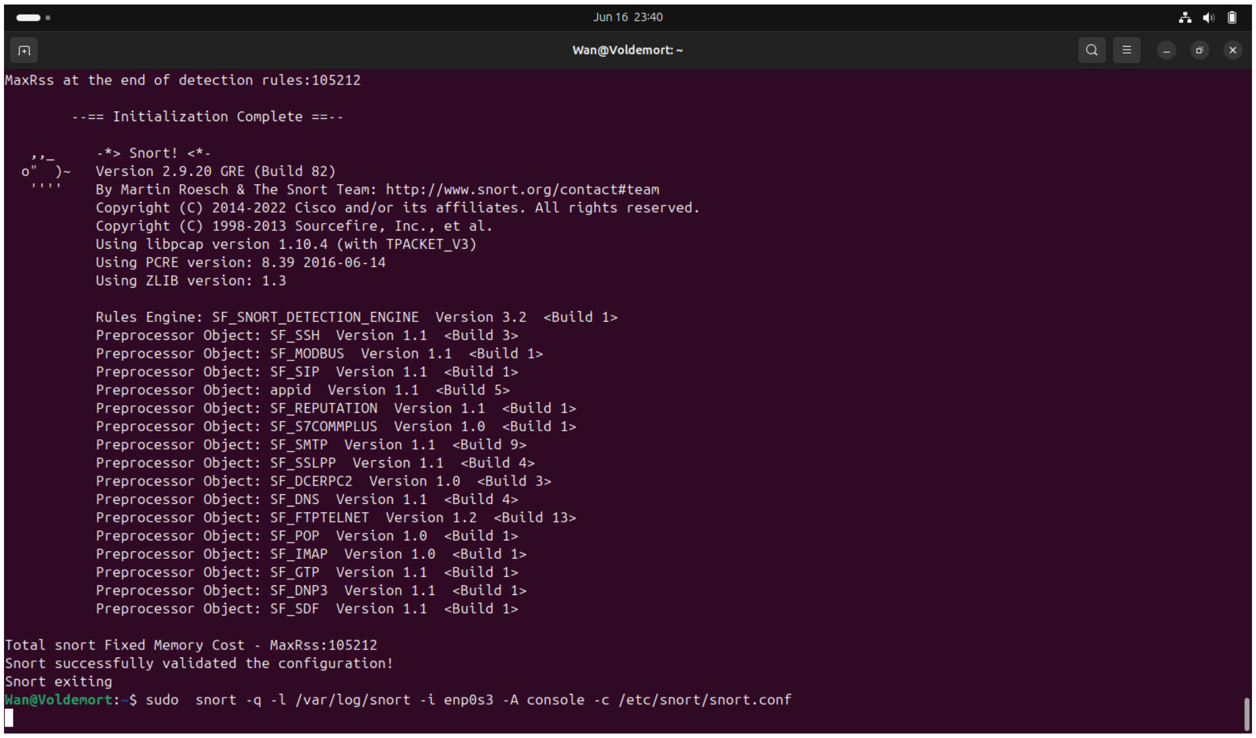


```

# site specific rules
include $RULE_PATH/local.rules

# The include files commented below have been disabled
# because they are not available in the stock Debian
# rules. If you install the Sourcefire VRT please make
# sure you re-enable them again:
include $RULE_PATH/community-rules/community.rules
#include $RULE_PATH/app-detect.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/backdoor.rules
include $RULE_PATH/bad-traffic.rules
```

Text editor showing rules customization instructions

Finally to test if changes made to the configuration files are valid, save and exit the text editor and run the command below in the terminal

```
Sudo -T -c /etc/snort/snort.conf
```



Jun 16 23:40

Wan@Voldemort: ~

```
MaxRss at the end of detection rules:105212

        --== Initialization Complete ==--

   ,,_     -*> Snort! <*-
  o"  )~   Version 2.9.20 GRE (Build 82)
   ''''    By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
           Copyright (C) 2014-2022 Cisco and/or its affiliates. All rights reserved.
           Copyright (C) 1998-2013 Sourcefire, Inc., et al.
           Using libpcap version 1.10.4 (with TPACKET_V3)
           Using PCRE version: 8.39 2016-06-14
           Using ZLIB version: 1.3

           Rules Engine: SF_SNORT_DETECTION_ENGINE  Version 3.2  <Build 1>
           Preprocessor Object: SF_SSH  Version 1.1  <Build 3>
           Preprocessor Object: SF_MODBUS  Version 1.1  <Build 1>
           Preprocessor Object: SF_SIP  Version 1.1  <Build 1>
           Preprocessor Object: appid  Version 1.1  <Build 5>
           Preprocessor Object: SF_REPUTATION  Version 1.1  <Build 1>
           Preprocessor Object: SF_S7COMMPLUS  Version 1.0  <Build 1>
           Preprocessor Object: SF_SMTP  Version 1.1  <Build 9>
           Preprocessor Object: SF_SSLPP  Version 1.1  <Build 4>
           Preprocessor Object: SF_DCERPC2  Version 1.0  <Build 3>
           Preprocessor Object: SF_DNS  Version 1.1  <Build 4>
           Preprocessor Object: SF_FTPTELNET  Version 1.2  <Build 13>
           Preprocessor Object: SF_POP  Version 1.0  <Build 1>
           Preprocessor Object: SF_IMAP  Version 1.0  <Build 1>
           Preprocessor Object: SF_GTP  Version 1.1  <Build 1>
           Preprocessor Object: SF_DNP3  Version 1.1  <Build 1>
           Preprocessor Object: SF_SDF  Version 1.1  <Build 1>

Total snort Fixed Memory Cost - MaxRss:105212
Snort successfully validated the configuration!
Snort exiting
Wan@Voldemort:~$ sudo  snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
```

Ubuntu Terminal showing Snort Config successfully validated

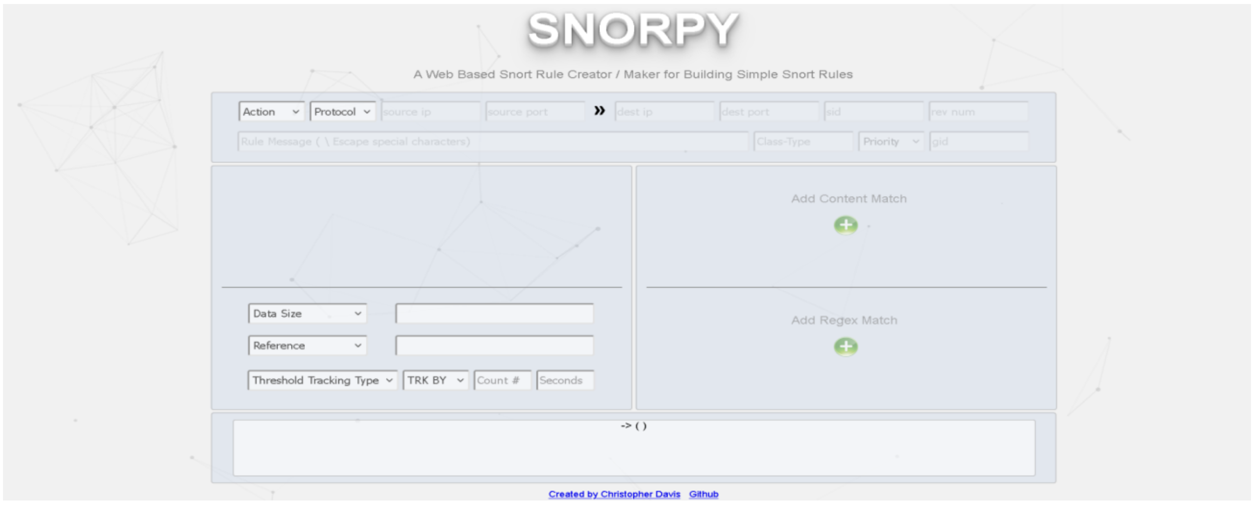
The last part of the output should be like this. Errors can be traced back as instructed on the terminal.

Creating Rules within the Local.rule File

The `Local.rule` file, as mentioned above, is the file where we get to write our own rules that snort will load in when trying to monitor network traffic. The first thing to do will be to use a text editor to edit the file.

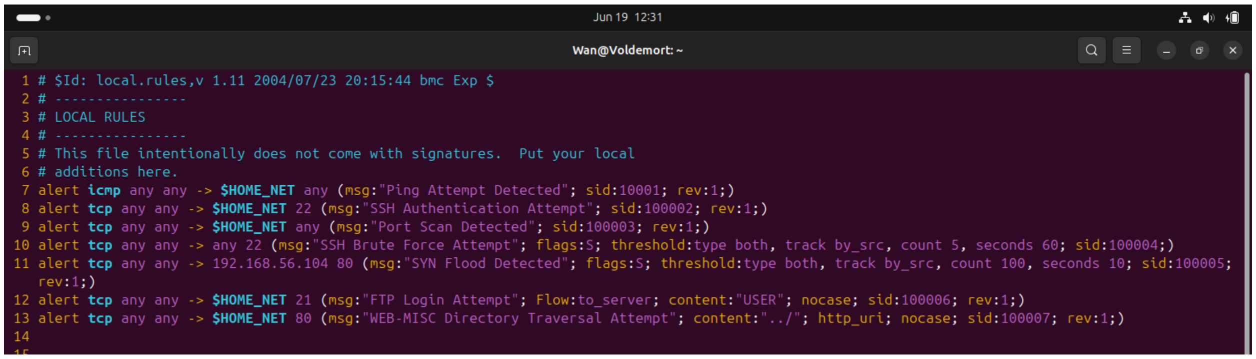
```
sudo vim /etc/snort/rules/local.rules
```

Websites `Snorpy` can help beginners to get familiar with creating rules.



Here are few rules created for the sake of this lab inside the `local.rules` file

- ➔ `alert icmp any any -> $HOME_NET any (msg:"Ping Attempt Detected"; sid:10001; rev:1;)`
- ➔ `alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)`
- ➔ `alert tcp any any -> $HOME_NET any (msg:"Port Scan Detected"; sid:100003; rev:1;)`
- ➔ `alert tcp any any -> any 22 (msg:"SSH Brute Force Attempt"; flags:S; threshold:type both, track by_src, count 5, seconds 60; sid:100004;)`
- ➔ `alert tcp any any -> 192.168.56.104 80 (msg:"SYN Flood Detected"; flags:S; threshold:type both, track by_src, count 100, seconds 10; sid:100005; rev:1;)`
- ➔ `alert tcp any any -> $HOME_NET 21 (msg:"FTP Login Attempt"; Flow:to_server; content:"USER"; nocase; sid:100006; rev:1;)`
- ➔ `alert tcp any any -> $HOME_NET 80 (msg:"WEB-MISC Directory Traversal Attempt"; content:"../"; http_uri; nocase; sid:100007; rev:1;)`



```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert icmp any any -> $HOME_NET any (msg:"Ping Attempt Detected"; sid:10001; rev:1;)
alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
alert tcp any any -> $HOME_NET any (msg:"Port Scan Detected"; sid:100003; rev:1;)
alert tcp any any -> any 22 (msg:"SSH Brute Force Attempt"; flags:S; threshold:type both, track by_src, count 5, seconds 60; sid:100004;)
alert tcp any any -> 192.168.56.104 80 (msg:"SYN Flood Detected"; flags:S; threshold:type both, track by_src, count 100, seconds 10; sid:100005;
rev:1;)
alert tcp any any -> $HOME_NET 21 (msg:"FTP Login Attempt"; Flow:to_server; content:"USER"; nocase; sid:100006; rev:1;)
alert tcp any any -> $HOME_NET 80 (msg:"WEB-MISC Directory Traversal Attempt"; content:"../"; http_uri; nocase; sid:100007; rev:1;)

```

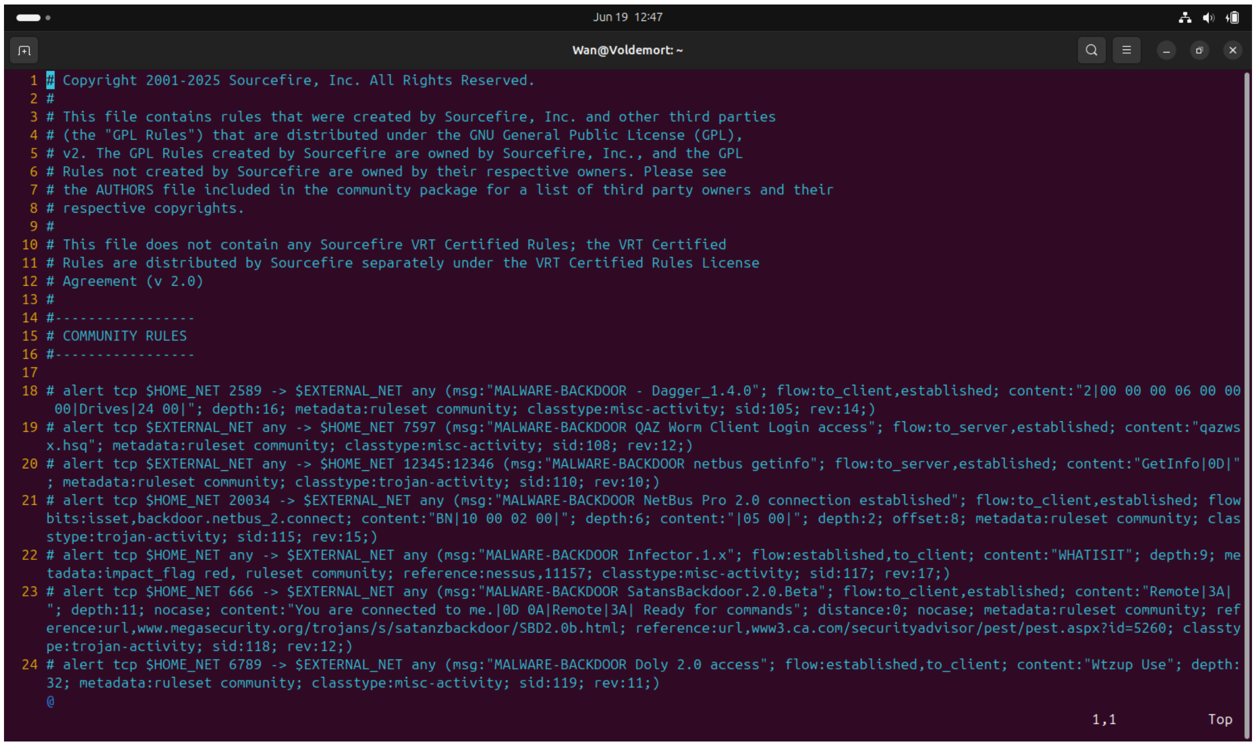
Text editor showing local rule

Community Rules Integration

Looking at the community rules can help with rules writing as well. The first thing to do would be to enable the community rules in the `snort.conf file` by uncommenting - looking for the line where it is included and remove the <#> - or if it was not included, we then have to include it.

```
include $RULE_PATH/community-rules/community.rules
```

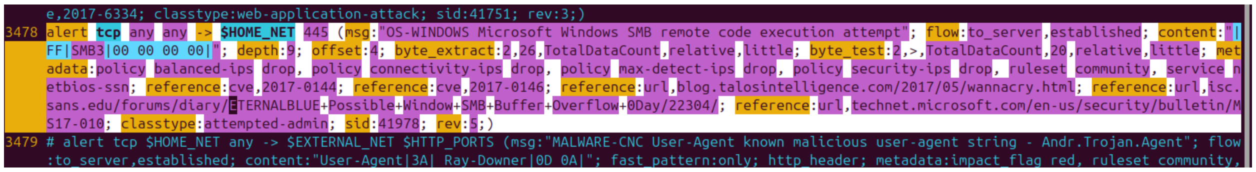
The community rules can be downloaded via Snort website - snort.org. You have to download a version that is compatible with the version of your snort and extract it ideally straight into your `rules` directory under the `/etc/snort/rules/` path. After extraction, it will contain a `community-rules` directory, and inside this directory, there is a `snort.conf` file based on the community rules. It will also have the `community.rules` file which is where the community rules are.



```
# Copyright 2001-2025 Sourcefire, Inc. All Rights Reserved.
#
# This file contains rules that were created by Sourcefire, Inc. and other third parties
# (the "GPL Rules") that are distributed under the GNU General Public License (GPL),
# v2. The GPL Rules created by Sourcefire are owned by Sourcefire, Inc., and the GPL
# Rules not created by Sourcefire are owned by their respective owners. Please see
# the AUTHORS file included in the community package for a list of third party owners and their
# respective copyrights.
#
# This file does not contain any Sourcefire VRT Certified Rules; the VRT Certified
# Rules are distributed by Sourcefire separately under the VRT Certified Rules License
# Agreement (v 2.0)
#
#----------------
# COMMUNITY RULES
#----------------

# alert tcp $HOME_NET 2589 -> $EXTERNAL_NET any (msg:"MALWARE-BACKDOOR - Dagger_1.4.0"; flow:to_client,established; content:"2|00 00 00 06 00 00
 00|Drives|24 00|"; depth:16; metadata:ruleset community; classtype:misc-activity; sid:105; rev:14;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 (msg:"MALWARE-BACKDOOR QAZ Worm Client Login access"; flow:to_server,established; content:"qazws
x.hsq"; metadata:ruleset community; classtype:misc-activity; sid:108; rev:12;)
# alert tcp $EXTERNAL_NET any -> $HOME_NET 12345:12346 (msg:"MALWARE-BACKDOOR netbus getinfo"; flow:to_server,established; content:"GetInfo|0D|"
; metadata:ruleset community; classtype:trojan-activity; sid:110; rev:10;)
# alert tcp $HOME_NET 20034 -> $EXTERNAL_NET any (msg:"MALWARE-BACKDOOR NetBus Pro 2.0 connection established"; flow:to_client,established; flow
bits:isset,backdoor.netbus_2.connect; content:"BN|10 00 02 00|"; depth:6; content:"|05 00|"; depth:2; offset:8; metadata:ruleset community; clas
stype:trojan-activity; sid:115; rev:15;)
# alert tcp $HOME_NET any -> $EXTERNAL_NET any (msg:"MALWARE-BACKDOOR Infector.1.x"; flow:established,to_client; content:"WHATISIT"; depth:9; me
tadata:impact_flag red, ruleset community; reference:nessus,11157; classtype:misc-activity; sid:117; rev:17;)
# alert tcp $HOME_NET 666 -> $EXTERNAL_NET any (msg:"MALWARE-BACKDOOR SatansBackdoor.2.0.Beta"; flow:to_client,established; content:"Remote|3A|
"; depth:11; nocase; content:"You are connected to me.|0D 0A|Remote|3A| Ready for commands"; distance:0; nocase; metadata:ruleset community; ref
erence:url,www.megasecurity.org/trojans/s/satanzbackdoor/SBD2.0b.html; reference:url,www3.ca.com/securityadvisor/pest/pest.aspx?id=5260; classty
pe:trojan-activity; sid:118; rev:12;)
# alert tcp $HOME_NET 6789 -> $EXTERNAL_NET any (msg:"MALWARE-BACKDOOR Doly 2.0 access"; flow:established,to_client; content:"Wtzup Use"; depth:
32; metadata:ruleset community; classtype:misc-activity; sid:119; rev:11;)
@
                                                                                                      1,1           Top
```

Text editor showing community rules

One of the examples of community rules to consider is `Eternal Blue`.



```
e,2017-6334; classtype:web-application-attack; sid:41751; rev:3;)
alert tcp any any -> $HOME_NET 445 (msg:"OS-WINDOWS Microsoft Windows SMB remote code execution attempt"; flow:to_server,established; content:"|
FF|SMB3|00 00 00 00|"; depth:9; offset:4; byte_extract:2,26,TotalDataCount,relative,little; byte_test:2,>,TotalDataCount,20,relative,little; met
adata:policy balanced-ips drop, policy connectivity-ips drop, policy max-detect-ips drop, policy security-ips drop, ruleset community, service n
etbios-ssn; reference:cve,2017-0144; reference:cve,2017-0146; reference:url,blog.talosintelligence.com/2017/05/wannacry.html; reference:url,isc.
sans.edu/forums/diary/ETERNALBLUE+Possible+Window+SMB+Buffer+Overflow+0Day/22304/; reference:url,technet.microsoft.com/en-us/security/bulletin/M
S17-010; classtype:attempted-admin; sid:41978; rev:5;)
# alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS (msg:"MALWARE-CNC User-Agent known malicious user-agent string - Andr.Trojan.Agent"; flow
:to_server,established; content:"User-Agent|3A| Ray-Downer|0D 0A|"; fast_pattern:only; http_header; metadata:impact_flag red, ruleset community,
```

Text editor showing Eternal Blue rule crafted to detect Intrusion

Eternal Blue is an exploit or vulnerability within Windows SMB version one that allows you to perform remote code execution.

Rules can be added to the `local.rules` file from the `community.rules` in order to have it configured for the Snort operations. Most of the rules inside the community rules are more complex and difficult for beginners to write but are important for the functionality of the Snort. You can search for a specific rule using any part of the syntax and copy it to be pasted inside the `local.rules` file. You just have to make sure the signature ID is unique.

Here are few rules copied from the community rules to the local rules

➔ `alert tcp $EXTERNAL_NET any -> $HOME_NET 7597 (msg:"MALWARE-BACKDOOR QAZ Worm Client Login access"; flow:to_server,established; content:"qazws          x.hsq"; metadata:ruleset community; classtype:misc-activity; sid:100008; rev:12;)`

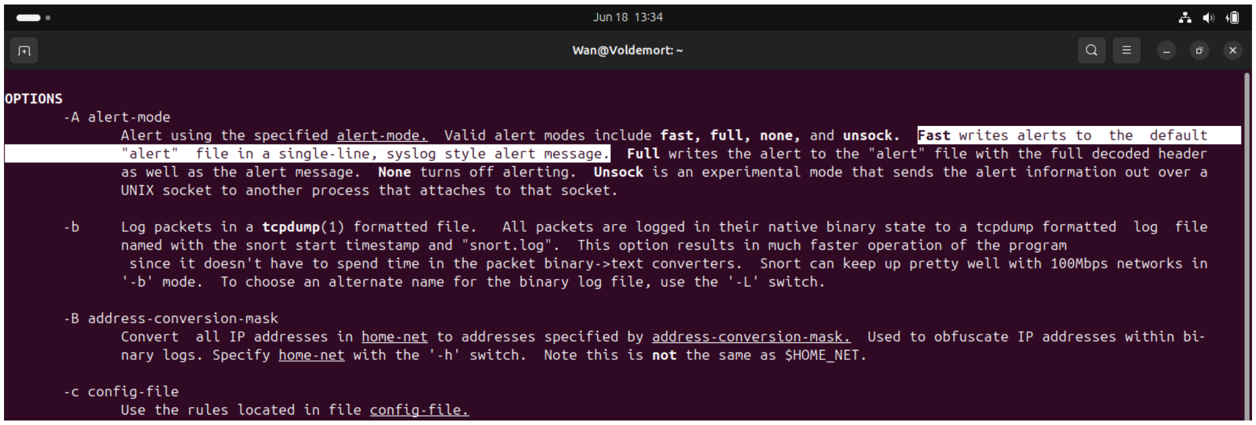
➔ `alert udp $EXTERNAL_NET 3345 -> $HOME_NET 3344 (msg:"MALWARE-BACKDOOR Matrix 2.0 Server access"; flow:to_server; content:"logged in"; metadata:ru    leset community; classtype:misc-activity; sid:100009; rev:10;)`

Another great thing with Snort rules is that it allows you to also specify this reference point so it can specify the CVE code as well as the URL where you can read more about a specific vulnerability.

Running Snort

You need to understand a few options when running Snort. The most important one is the `Alert Mode <-A>`. It allows you to specify the actual format of alerts. It ranges from `fast`, `full`, `none` and `unsock`**.** More will be discussed on this function later but what they each do can be accessed with the command below. It displays the entire manual for using Snort.

```
man snort
```



```
OPTIONS
       -A alert-mode
              Alert using the specified alert-mode.  Valid alert modes include fast, full, none, and unsock.  Fast writes alerts to  the  default
              "alert"  file in a single-line, syslog style alert message.  Full writes the alert to the "alert" file with the full decoded header
              as well as the alert message.  None turns off alerting.  Unsock is an experimental mode that sends the alert information out over a
              UNIX socket to another process that attaches to that socket.

       -b     Log packets in a tcpdump(1) formatted file.   All packets are logged in their native binary state to a tcpdump formatted  log  file
              named with the snort start timestamp and "snort.log".  This option results in much faster operation of the program
               since it doesn't have to spend time in the packet binary->text converters.  Snort can keep up pretty well with 100Mbps networks in
              '-b' mode.  To choose an alternate name for the binary log file, use the '-L' switch.

       -B address-conversion-mask
              Convert  all IP addresses in home-net to addresses specified by address-conversion-mask.  Used to obfuscate IP addresses within bi-
              nary logs. Specify home-net with the '-h' switch.  Note this is not the same as $HOME_NET.

       -c config-file
              Use the rules located in file config-file.
```

`Text editor showing Snort Manual`

Another option is the `log-dir <-l>` option. This basically sets the output logging directory to `log-dir`. All plain text alerts and packet logs go into this directory. Another one is the `interface <-i>` option which specifies the interface. Another option to take note of will be `Quiet operation <-q>` which tells Snort to run in quiet operation which is very important.

Running snort would require combining some of the arguments above . Consider the command below

```
sudo snort -q -i enp0s3 -l /var/log/snort -A console -c
/etc/snort/snort.conf
```

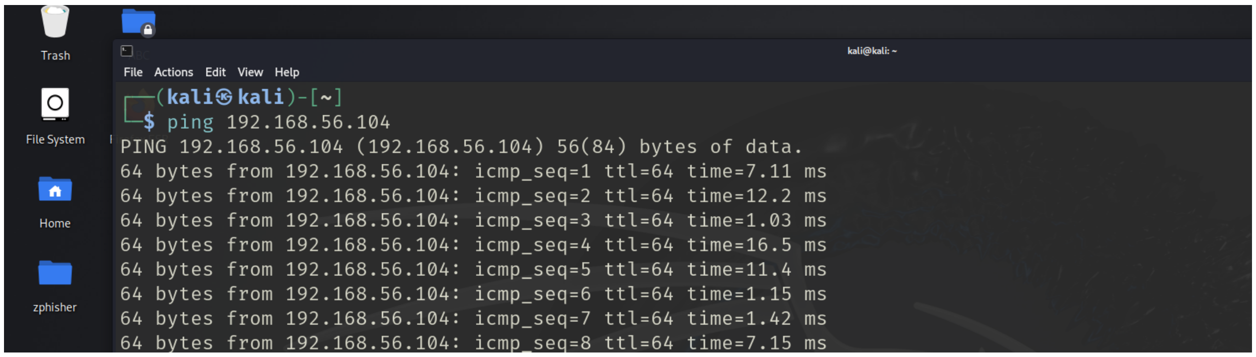
`-l  /vat/log/snort` is specifying the path to where all packets should be logged. `Console -c` is stating that traffic captured should be displayed on the terminal. If it doesn't bring output or error after entering that command, then snort is running (quietly <-q>).

Simulating Intrusion

One way to test that all configurations and rules are functioning as intended is to simulate intrusion as well. For the purpose of this lab set up, Kali Linux was used to simulate intrusion on Metasploitable 2. These Intrusion are specifically based on the rules that were created to make sure alerts are captured in real time.

The aim here is to see from a defender's perspective , what this network activity looks like. And if there are written rules for it, they should be able to detect what's being done on the network. That's the essence of IDS and Network Analysis. In the context of Network Intrusion Detection, we are essentially fine-tuning our network traffic analysis to identify and automate the process of identifying intrusion or threat.

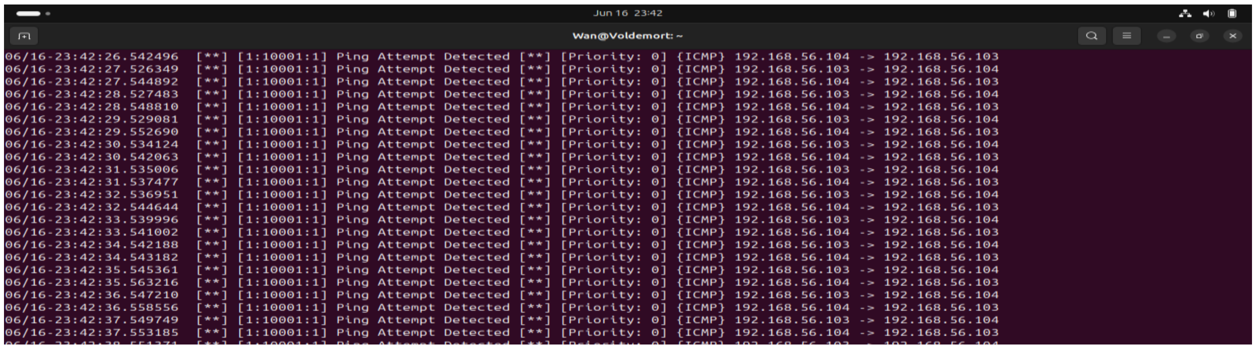
The first one was a ping attempt on the vulnerable machine from the Kali VM



```
┌──(kali㉿kali)-[~]
└─$ ping 192.168.56.104
PING 192.168.56.104 (192.168.56.104) 56(84) bytes of data.
64 bytes from 192.168.56.104: icmp_seq=1 ttl=64 time=7.11 ms
64 bytes from 192.168.56.104: icmp_seq=2 ttl=64 time=12.2 ms
64 bytes from 192.168.56.104: icmp_seq=3 ttl=64 time=1.03 ms
64 bytes from 192.168.56.104: icmp_seq=4 ttl=64 time=16.5 ms
64 bytes from 192.168.56.104: icmp_seq=5 ttl=64 time=11.4 ms
64 bytes from 192.168.56.104: icmp_seq=6 ttl=64 time=1.15 ms
64 bytes from 192.168.56.104: icmp_seq=7 ttl=64 time=1.42 ms
64 bytes from 192.168.56.104: icmp_seq=8 ttl=64 time=7.15 ms
```

`Kali Terminal showing Ping Attempt`

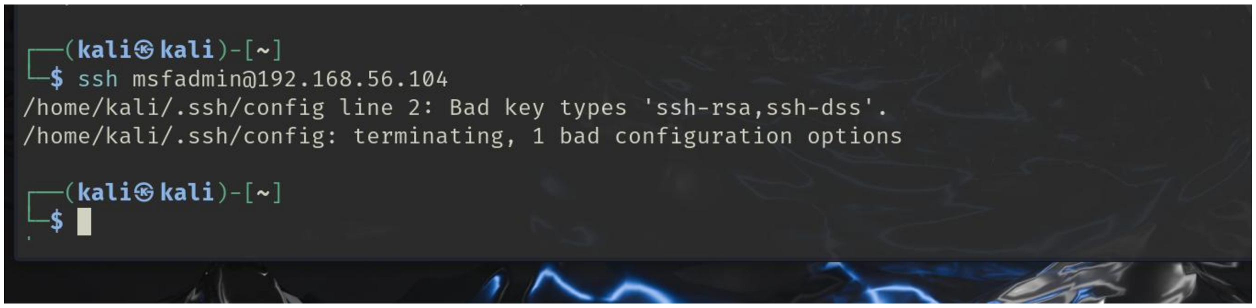
And as expected, Ubuntu captured the intrusion and displayed it on the console as specified by the command that was used when running the Snort system.



```
06/16-23:42:26.542496  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:27.526349  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:27.544892  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:28.527483  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:28.548810  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:29.529081  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:29.552690  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:30.534124  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:30.542063  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:31.535006  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:31.537477  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:32.536951  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:32.544644  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:33.539996  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:33.541002  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:34.542188  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:34.543182  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:35.545361  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:35.563216  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:36.547210  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:36.558556  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/16-23:42:37.549749  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/16-23:42:37.553185  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
```

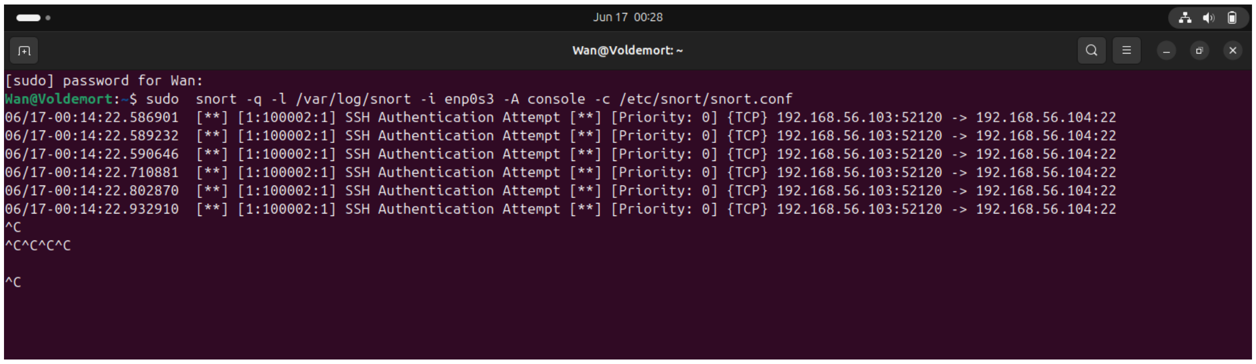
Ubuntu displaying Ping Attempt Alert

Other Intrusion was also simulated in the order of how the rules were crafted and subsequently the alerts were captured and displayed on the Ubuntu Snort terminal.



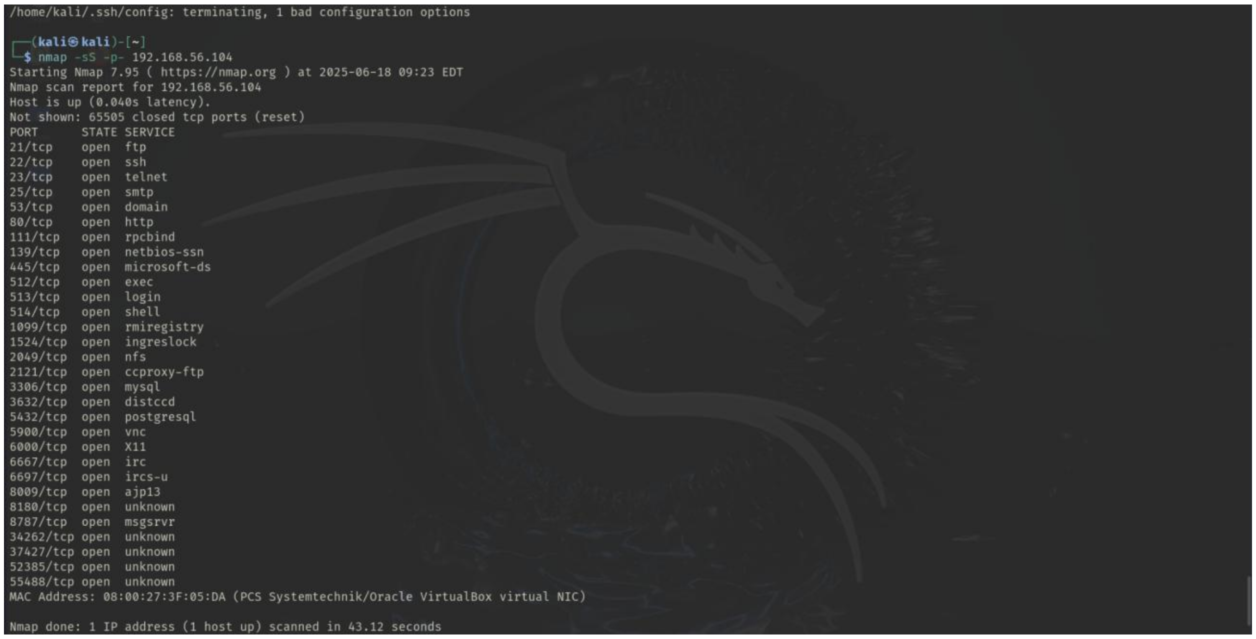
```
┌──(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.56.104
/home/kali/.ssh/config line 2: Bad key types 'ssh-rsa,ssh-dss'.
/home/kali/.ssh/config: terminating, 1 bad configuration options

┌──(kali㉿kali)-[~]
└─$ 
```

Kali Terminal showing SSH Authentication Attempt simulation



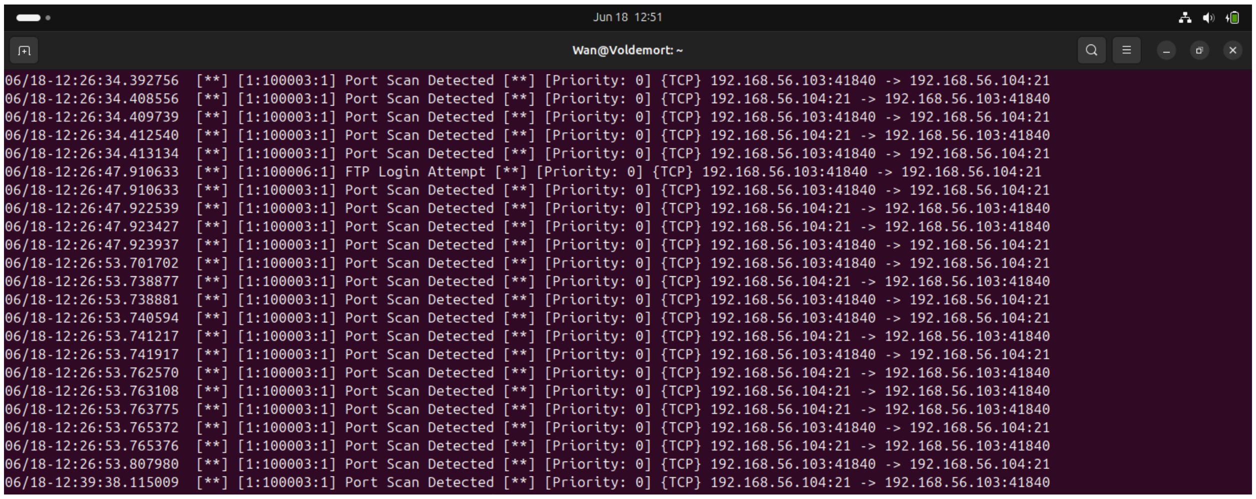
```
[sudo] password for Wan:
Wan@Voldemort:~$ sudo  snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
06/17-00:14:22.586901  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
06/17-00:14:22.589232  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
06/17-00:14:22.590646  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
06/17-00:14:22.710881  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
06/17-00:14:22.802870  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
06/17-00:14:22.932910  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.56.103:52120 -> 192.168.56.104:22
^C
^C^C^C^C

^C
```

Ubuntu displaying SSH Authentication Attempt Alert



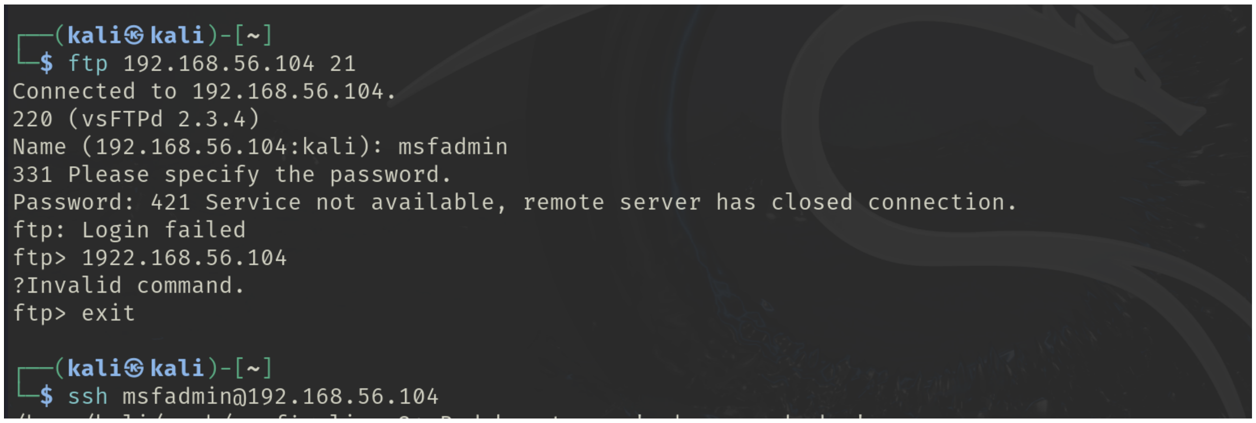
```
/home/kali/.ssh/config: terminating, 1 bad configuration options

┌──(kali㉿kali)-[~]
└─$ nmap -sS -p- 192.168.56.104
Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-18 09:23 EDT
Nmap scan report for 192.168.56.104
Host is up (0.040s latency).
Not shown: 65505 closed tcp ports (reset)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
23/tcp    open  telnet
25/tcp    open  smtp
53/tcp    open  domain
80/tcp    open  http
111/tcp   open  rpcbind
139/tcp   open  netbios-ssn
445/tcp   open  microsoft-ds
512/tcp   open  exec
513/tcp   open  login
514/tcp   open  shell
1099/tcp  open  rmiregistry
1524/tcp  open  ingreslock
2049/tcp  open  nfs
2121/tcp  open  ccproxy-ftp
3306/tcp  open  mysql
3632/tcp  open  distccd
5432/tcp  open  postgresql
5900/tcp  open  vnc
6000/tcp  open  X11
6667/tcp  open  irc
6697/tcp  open  ircs-u
8009/tcp  open  ajp13
8180/tcp  open  unknown
8787/tcp  open  msgsrvr
34262/tcp open  unknown
37427/tcp open  unknown
52385/tcp open  unknown
55488/tcp open  unknown
MAC Address: 08:00:27:3F:05:DA (PCS Systemtechnik/Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 43.12 seconds
```

Terminal showing Nmap Scan for Open Port



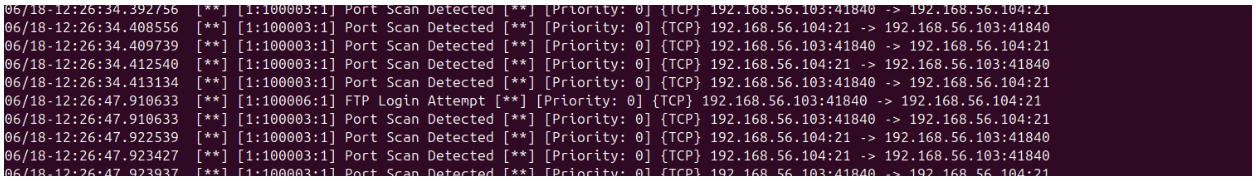
```
06/18-12:26:34.392756  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:34.408556  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:34.409739  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:34.412540  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:34.413134  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.910633  [**] [1:100006:1] FTP Login Attempt [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.910633  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.922539  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:47.923427  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:47.923937  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.701702  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.738877  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.738881  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.740594  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.741217  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.741917  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.762570  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.763108  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.763775  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.765372  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:53.765376  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:53.807980  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:39:38.115009  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
```

Ubuntu Snort Terminal displaying Port Scan Detected Alert



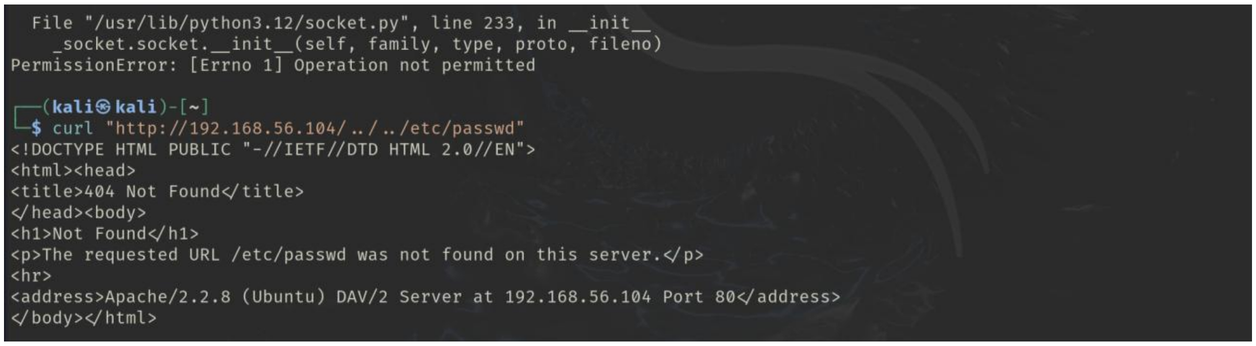
```
┌──(kali㉿kali)-[~]
└─$ ftp 192.168.56.104 21
Connected to 192.168.56.104.
220 (vsFTPd 2.3.4)
Name (192.168.56.104:kali): msfadmin
331 Please specify the password.
Password: 421 Service not available, remote server has closed connection.
ftp: Login failed
ftp> 1922.168.56.104
?Invalid command.
ftp> exit

┌──(kali㉿kali)-[~]
└─$ ssh msfadmin@192.168.56.104
```

Kali Terminal displaying FTP Connection Attempt



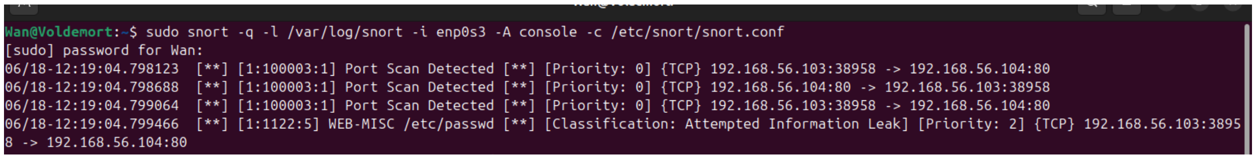
```
06/18-12:26:34.392756  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:34.408556  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:34.409739  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:34.412540  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:34.413134  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.910633  [**] [1:100006:1] FTP Login Attempt [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.910633  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
06/18-12:26:47.922539  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:47.923427  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:21 -> 192.168.56.103:41840
06/18-12:26:47.923937  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:41840 -> 192.168.56.104:21
```

Ubuntu Snort Terminal showing FTP login Attempt amidst other alerts



```
  File "/usr/lib/python3.12/socket.py", line 233, in __init__
    _socket.socket.__init__(self, family, type, proto, fileno)
PermissionError: [Errno 1] Operation not permitted

┌──(kali㉿kali)-[~]
└─$ curl "http://192.168.56.104/../../etc/passwd"
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Not Found</title>
</head><body>
<h1>Not Found</h1>
<p>The requested URL /etc/passwd was not found on this server.</p>
<hr>
<address>Apache/2.2.8 (Ubuntu) DAV/2 Server at 192.168.56.104 Port 80</address>
</body></html>
```

Kali Terminal showing Web Misc Dir Traversal Attempt



```
Wan@Voldemort:~$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
[sudo] password for Wan:
06/18-12:19:04.798123  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:38958 -> 192.168.56.104:80
06/18-12:19:04.798688  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.104:80 -> 192.168.56.103:38958
06/18-12:19:04.799064  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:38958 -> 192.168.56.104:80
06/18-12:19:04.799466  [**] [1:1122:5] WEB-MISC /etc/passwd [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP} 192.168.56.103:3895
8 -> 192.168.56.104:80
```

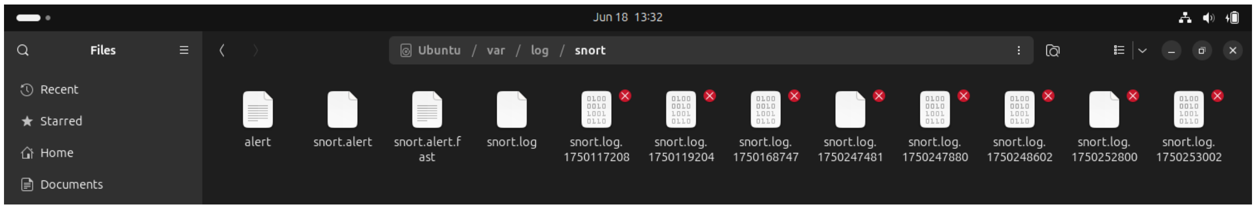
Ubuntu Snort Terminal showing WED MISC Attempt Alert

Logging Alert

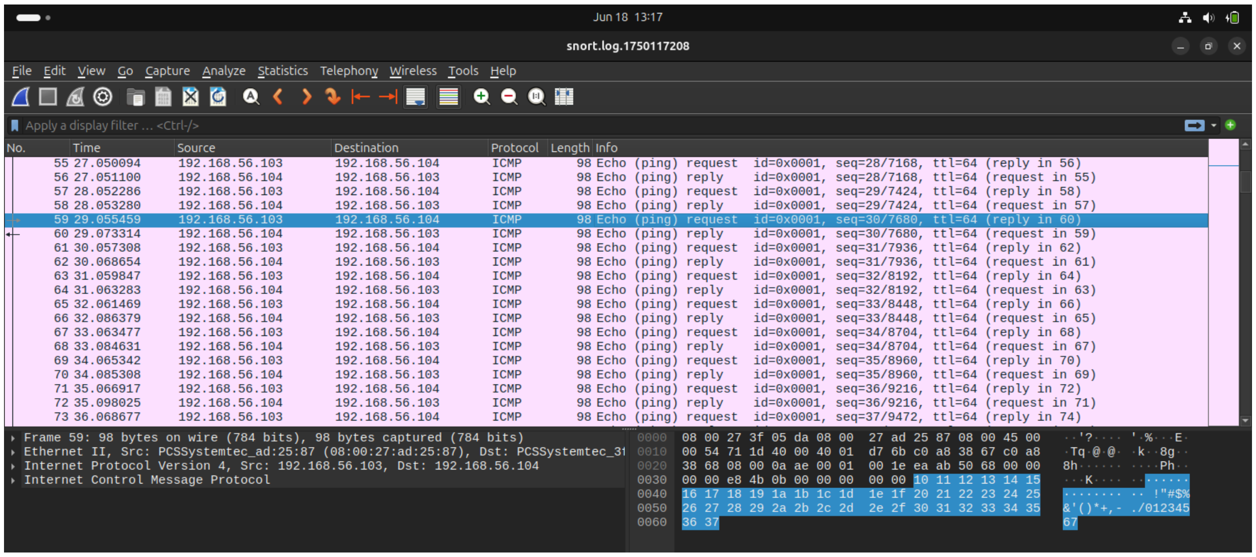
When running Snort, based on the command, the alerts are either displayed on the console or logged inside a file named `Alert` under the `/var/log/snort` path. If you want the alerts displayed on the console, you have to add the argument `-A console -c` when writing the command

```
sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
```

The traffic generated during the Snort activity is also logged under the `/var/log/snort` can be opened with `wireshark` for further analysis.



File manager showing files containing traffic captured by snort



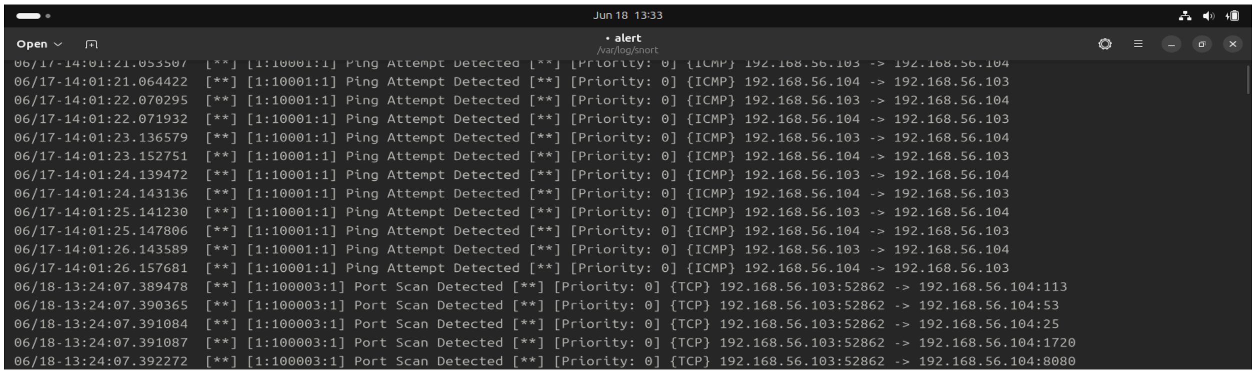
```
No. Time      Source          Destination     Protocol Length Info
 55 27.050094 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=28/7168, ttl=64 (reply in 56)
 56 27.051100 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=28/7168, ttl=64 (request in 55)
 57 28.052286 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=29/7424, ttl=64 (reply in 58)
 58 28.053280 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=29/7424, ttl=64 (request in 57)
 59 29.055459 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=30/7680, ttl=64 (reply in 60)
 60 29.073314 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=30/7680, ttl=64 (request in 59)
 61 30.057308 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=31/7936, ttl=64 (reply in 62)
 62 30.068654 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=31/7936, ttl=64 (request in 61)
 63 31.059847 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=32/8192, ttl=64 (reply in 64)
 64 31.063283 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=32/8192, ttl=64 (request in 63)
 65 32.061469 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=33/8448, ttl=64 (reply in 66)
 66 32.086379 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=33/8448, ttl=64 (request in 65)
 67 33.063477 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=34/8704, ttl=64 (reply in 68)
 68 33.084631 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=34/8704, ttl=64 (request in 67)
 69 34.065342 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=35/8960, ttl=64 (reply in 70)
 70 34.085308 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=35/8960, ttl=64 (request in 69)
 71 35.066917 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=36/9216, ttl=64 (reply in 72)
 72 35.098025 192.168.56.104  192.168.56.103  ICMP     98 Echo (ping) reply    id=0x0001, seq=36/9216, ttl=64 (request in 71)
 73 36.068677 192.168.56.103  192.168.56.104  ICMP     98 Echo (ping) request  id=0x0001, seq=37/9472, ttl=64 (reply in 74)

Frame 59: 98 bytes on wire (784 bits), 98 bytes captured (784 bits)
Ethernet II, Src: PCSSystemtec_ad:25:87 (08:00:27:ad:25:87), Dst: PCSSystemtec_31
Internet Protocol Version 4, Src: 192.168.56.103, Dst: 192.168.56.104
Internet Control Message Protocol

0000  08 00 27 3f 05 da 08 00  27 ad 25 87 08 00 45 00   ··'?···· '·%···E·
0010  00 54 71 1d 40 00 40 01  d7 6b c0 a8 38 67 c0 a8   ·Tq·@·@· ·k··8g··
0020  38 68 08 00 0a ae 00 01  00 1e ea ab 50 68 00 00   8h······ ····Ph··
0030  00 00 e8 4b 0b 00 00 00  00 00 10 11 12 13 14 15   ···K···· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                              67
```

Wireshark displaying traffic captured by snort

To specifically log the actual alert, the arguments `-A fast -c` should be added to the command when running snort.

```
sudo snort -q -l /var/log/snort -i enp0s3 -A fast -c /etc/snort/snort.conf
```

The alerts will not be visible on the Snort Terminal but will be logged in `Alert` under the `/var/log/snort` path and ready for exporting for further analysis using tools like spunk.



```
06/17-14:01:21.053507  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:21.064422  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/17-14:01:22.070295  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:22.071932  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/17-14:01:23.136579  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:23.152751  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/17-14:01:24.139472  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:24.143136  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/17-14:01:25.141230  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:25.147806  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/17-14:01:26.143589  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.103 -> 192.168.56.104
06/17-14:01:26.157681  [**] [1:10001:1] Ping Attempt Detected [**] [Priority: 0] {ICMP} 192.168.56.104 -> 192.168.56.103
06/18-13:24:07.389478  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:52862 -> 192.168.56.104:113
06/18-13:24:07.390365  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:52862 -> 192.168.56.104:53
06/18-13:24:07.391084  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:52862 -> 192.168.56.104:25
06/18-13:24:07.391087  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:52862 -> 192.168.56.104:1720
06/18-13:24:07.392272  [**] [1:100003:1] Port Scan Detected [**] [Priority: 0] {TCP} 192.168.56.103:52862 -> 192.168.56.104:8080
```

Alert file showing logged alerts captured by Snort

Setup Issues & Error and Fixes

- Snort Config Test Error: Undefined `$HOME_NET`
  - Issue: `sudo snort -T` failed due to undefined `$HOME_NET` in `local.rules` or `snort.conf`.
  - Fix: Set `ipvar HOME_NET 192.168.56.0/24` in `/etc/snort/snort.conf`.