TITLE: A PROJECT REPORT ONNETWORK INTRUSION DETECTION SYSTEM USING SNORT

SUBMITTED BY:

Name: Sayuj Sur

COMPANY DETAILS

COMPANY: Infotact Solutions

Mentor Name: Vasudev Jha

DATE: 10.10.2025

Table of contents

Section Type	Title
	ABSTRACT
	ACKNOWLEDGEMENT
Chapter 1	**INTRODUCTION TO INTRUSION DETECTION SYSTEM (IDS)**
	1.1 Intrusion Detection System (IDS)
	1.2 Working of Intrusion Detection System (IDS)
	1.3 Classification of Intrusion Detection System (IDS)
Chapter 2	**INTRODUCTION TO SNORT**
	2.1 Features
	2.2 Types of Rules in SNORT
	2.3 Basic Usages
Chapter 3	**WORKING OF SNORT**
	3.1 Packet Sniffer Mode
	3.2 Packet Logging Mode
	3.3 NIDS (Network Intrusion Detection System) Mode
Chapter 4	**LAB ENVIRONMENT SCENARIO AND REQUIREMENTS FOR SNORT**
Chapter 5	**INSTALLATION & CONFIGURATION OF SNORT**
Chapter 6	**CONCLUSION AND FUTURE SCOPE**

ABSTRACT

This project report captures the installation and setup of a Network Intrusion Detection System (NIDS) using the open-source Snort tool. In response to the need for real-time threat examination in contemporary network defense, the main aim was to develop and deploy an operational NIDS that can observe network traffic for malicious behavior within a test lab environment. The experiment involved intensive examination of fundamental Intrusion Detection System principles, outlining Snort's three modes of operation (Sniffer, Logger, and NIDS), and providing step-by-step installation and configuration instructions. Success in the experiment validates the actual use of Snort to design, implement, and maintain tailored rule-sets for efficient, signature-based intrusion detection and subsequent alerting against emulated cyber threats, thus proving the system's capacity to strengthen network security posture.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who supported me throughout the successful completion of this project on the **Network Intrusion Detection System using Snort**.

First and foremost, I extend my deepest appreciation to **Infotact Solutions** for providing me with this invaluable internship opportunity and the necessary resources to conduct this research and implementation.

My profound thanks go to my dedicated mentor, **Vasudev Jha** for their expert guidance, constant encouragement, and willingness to share their extensive knowledge. Their insights were instrumental in overcoming technical challenges and shaping the final outcome of this report.

1 INTRODUCTION to Intrusion Detection System (IDS)

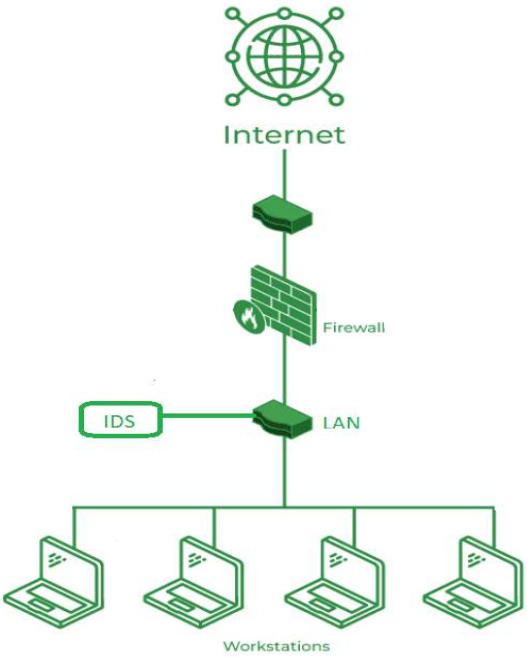
Intrusion Detection System (IDS)

Intrusion is when an attacker gets unauthorized access to a device, network, or system. Cyber criminals use advanced techniques to sneak into organizations without being detected.

Intrusion Detection System (IDS) observes network traffic for malicious transactions and sends immediate alerts when it is observed. It is software that checks a network or system for malicious activities or policy violations. Each illegal activity or violation is often recorded either centrally using an SIEM system or notified to an administration. IDS monitors a network or system for malicious activity and protects a computer network from unauthorized access from users, including perhaps insiders. The intrusion detector learning task is to build a predictive model (i.e. a classifier) capable of distinguishing between 'bad connections' (intrusion/attacks) and 'good (normal) connections'.

Working of Intrusion Detection System(IDS)

- An IDS (Intrusion Detection System) monitors the traffic on a computer network to detect any suspicious activity.
- It analyzes the data flowing through the network to look for patterns and signs of abnormal behavior.
- The IDS compares the network activity to a set of predefined rules and patterns to identify any activity that might indicate an attack or intrusion.
- If the IDS detects something that matches one of these rules or patterns, it sends an alert to the system administrator.
- The system administrator can then investigate the alert and take action to prevent any damage or further intrusion.



Classification of Intrusion Detection System(IDS)

Intrusion Detection System are classified into 5 types:

- **Network Intrusion Detection System (NIDS):** Network intrusion detection systems (NIDS) are set up at a planned point within the network to examine traffic from all devices on the network. It performs an observation of passing traffic on the entire subnet and matches the traffic that is passed on the subnets to the collection of known attacks. Once an attack is identified or abnormal behavior is observed, the alert can be sent to the administrator. An example of a NIDS is installing it on the subnet where firewalls are located in order to see if someone is trying to crack the firewall.
- **Host Intrusion Detection System (HIDS):** Host intrusion detection systems (HIDS) run on independent hosts or devices on the network. A HIDS monitors the incoming and outgoing packets from the device only and will alert the administrator if suspicious or malicious activity is detected. It takes a snapshot of existing system files and compares it with the previous snapshot. If the analytical system files were edited or deleted, an alert is sent to the administrator to investigate. An example of HIDS usage can be seen on mission-critical machines, which are not expected to change their layout.
- **Hybrid Intrusion Detection System:** Hybrid intrusion detection system is made by the combination of two or more approaches to the intrusion detection system. In the hybrid intrusion detection system, the host agent or system data is combined with network information to develop a complete view of the network system. The hybrid intrusion detection system is more effective in comparison to the other intrusion detection system. Prelude is an example of Hybrid IDS.
- **Application Protocol-Based Intrusion Detection System (APIDS):** An application Protocol-based Intrusion Detection System (APIDS) is a system or agent that generally resides within a group of servers. It identifies the intrusions by monitoring and interpreting the communication on application-specific protocols. For example, this would monitor the SQL protocol explicitly to the middleware as it transacts with the database in the web server.
- **Protocol-Based Intrusion Detection System (PIDS):** It comprises a system or agent that would consistently reside at the front end of a server, controlling and interpreting the protocol between a user/device and the server. It is trying to secure the web server by regularly monitoring the HTTPS protocol stream and accepting the related HTTP protocol. As HTTPS is unencrypted and before instantly entering its web presentation layer then this system would need to reside in this interface, between to use the HTTPS.
- **Signature-Based Detection:** Signature-based detection checks network packets for known patterns linked to specific threats. A signature-based IDS compares packets to a database of attack signatures and raises an alert if a match is found. Regular updates are needed to detect new threats, but unknown attacks without signatures can bypass this system

2 INTRODUCTION TO SNORT

SNORT is a network based intrusion detection system which is written in C programming language. It was developed in 1998 by Martin Roesch. Now it is developed by Cisco. It is free open-source software. It can also be used as a packet sniffer to monitor the system in real time. The network admin can use it to watch all the incoming packets and find the ones which are dangerous to the system. It is based on library packet capture tool. The rules are fairly easy to create and implement and it can be deployed in any kind of operating system and any kind of network environment. The main reason of the popularity of this IDS over others is that it is a free-to-use software and also open source because of which any user can be able to use it as the way he wants.

Features:

- Real-time traffic monitor
- Packet logging
- Analysis of protocol
- Content matching
- OS fingerprinting
- Can be installed in any network environment.
- Creates logs
- Open Source
- Rules are easy to implement

Types of Rules in SNORT:

There are 3 types of rules in SNORT, those are

1. **Alert Rules:** This uses the alert technique to produce notifications.
2. **Logging Rules:** It logs each individual alert as soon as it is generated.
3. **Pass Rules:** If the packet is deemed malicious, it is ignored and dropped.

Basic Usages:

- **Packet Sniffing:** The way traffic is being transmitted can be thoroughly examined by gathering the individual packets that travel to and from devices on the network.
- **Generates Alerts:** It generates warnings based on the configuration file's rules when it discovers unusual or malicious activity, the possibility of a vulnerability being exploited, or a network threat that compromises the organization's security policy.
- **Debug Traffic:** After the traffic has been logged, any malicious packets and configuration problems are checked.

There are currently 2 versions of Snort available:

1. Snort 2.X -De facto version of snort.
2. Snort 3.0 – Latest version of Snort that features improved efficiency, performance , scalability and usability over Snort 2.

In this Lab Report we're focusing on using Snort 2.x as it's the most widely implemented version and has extensive support, documentation & rule-sets.

Snort has 3 types of rules/rule-sets:

1. Community rules – Free rule-sets created by the Snort community.
2. Registered rules – Free rule-sets created by Talos. In order to use them, you must register for an account.
3. Subcription only rules – These rule-sets require an active paid subscription in order to be accessed and used.
4. Also we can write our own rules based on the requirements.

NOTE: Rules written for Snort 2 will need to be re-written or converted to the new Snort 3 format.

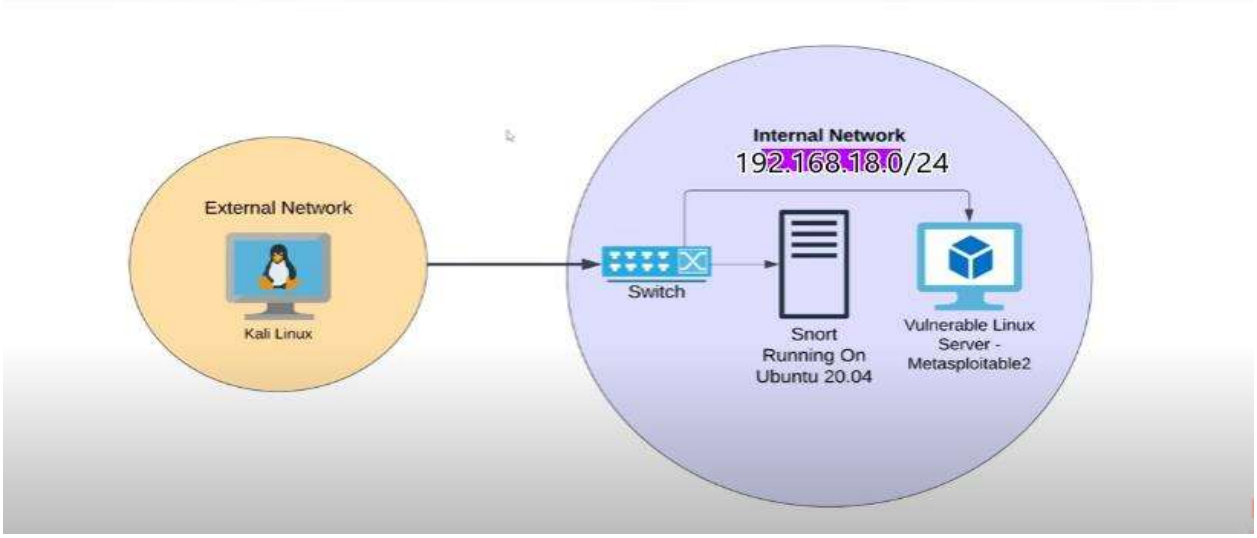
3 Working of Snort

The image illustrates three modes of operation for Snort, a popular open-source network intrusion detection system (NIDS):

1. **Packet Sniffer Mode**: In this mode, Snort functions like tcpdump, capturing and displaying network packets in real-time. It simply reads packet headers and payloads from the network interface and outputs them to the console. This mode is useful for basic network monitoring, troubleshooting, and understanding network traffic patterns. It's a passive observation mode, not actively analyzing for threats.

2. **Packet Logging Mode**: Here, Snort captures network packets and logs them to disk. Instead of just displaying them, it saves them in a format (often PCAP, which can be analyzed later with tools like Wireshark). This is crucial for forensic analysis, incident response, and long-term data collection. By logging all traffic or specific types of traffic, security analysts can review past events to understand how an attack unfolded or to identify persistent threats.
3. **NIDS (Network Intrusion Detection System) Mode**: This is Snort's primary and most powerful mode. In NIDS mode, Snort actively analyzes network traffic against a set of predefined rules to detect malicious activity, policy violations, and other suspicious behavior.
    - Rule-based Detection: Snort uses a comprehensive set of rules that describe various attack signatures, malware patterns, policy violations, and anomalous behaviors. These rules are highly customizable.
    - Real-time Analysis: As packets flow through the network, Snort inspects them in real-time, comparing them against its rule set.
    - Alerting: When a packet or a sequence of packets matches a rule, Snort generates an alert. These alerts can be logged, sent to a security information and event management (SIEM) system, or trigger other automated responses.
    - Preprocessors: Snort also uses preprocessors to normalize, decode, and analyze traffic before applying rules. These preprocessors can detect anomalies that simple rule matching might miss, like fragmented packets or malformed protocols.

4 LAB environment scenario and requiremnts for Snort



Scenario:

The lab environment simulates a simplified network where an **"External Network"** attempts to interact with an **"Internal Network."** The goal is to deploy Snort within the internal network to monitor traffic and detect potential malicious activities originating from the external network, specifically from a Kali Linux machine.

Components and their Roles:

- **External Network:**
    - **Kali Linux:** This machine acts as the attacker or the source of simulated external traffic. Kali Linux is a popular distribution for penetration testing and ethical hacking, containing a wide array of tools for network scanning, vulnerability exploitation, and other security assessments. In this scenario, it would be used to generate various types of traffic (e.g., port scans, exploit attempts) towards the Internal Network.
- **Internal Network (192.168.18.0/24):** This represents the protected network segment.
    - **Switch:** Connects the external network to the internal network components. It forwards traffic between the Kali Linux machine and the internal servers.
    - **Snort Running On Ubuntu 20.04:** This is the core of the intrusion detection system. Snort is deployed on an Ubuntu 20.04 server, positioned to monitor all traffic flowing between the External Network and the Vulnerable Linux Server. Its role is to analyze packets against configured rules and generate alerts when suspicious activity is detected. Snort needs to be configured in **NIDS mode** for this setup.
    - **Vulnerable Linux Server - Metasploitable2:** This machine serves as the target for attacks. Metasploitable2 is intentionally designed with numerous known security vulnerabilities, making it an ideal target for testing intrusion detection systems like Snort. Any attempt to exploit these vulnerabilities from the Kali Linux machine should ideally be detected by Snort.

Requirements for Snort Setup:

To successfully implement Snort in this lab environment, the following general requirements would apply:

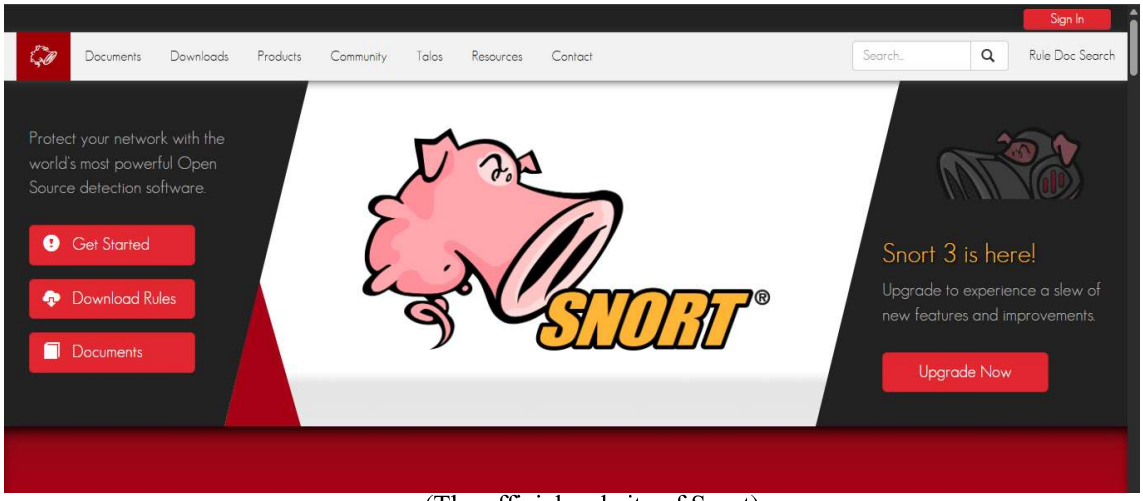
1. **Network Configuration:**
    - The Snort machine needs at least one network interface configured to be in **promiscuous mode**. This allows Snort to see all traffic passing through the switch, not just traffic destined for its own IP address.
    - Proper IP addressing within the 192.168.18.0/24 subnet for the Snort machine and the Metasploitable2 server.
2. **Operating System:**
    - **Ubuntu 20.04:** A clean installation of Ubuntu 20.04 is required to host Snort. All necessary system updates should be applied.
3. **Snort Installation and Configuration:**
    - **Installation:** Snort needs to be installed on the Ubuntu machine. This typically involves compiling from source or using package managers, along with its dependencies (e.g., libpcap, daq).
    - **Rule Sets:** Snort requires a set of rules to operate. This includes both community rules and potentially subscriber rules (if using a registered version). These rules must be regularly updated.
    - **Configuration Files:** The snort.conf file needs to be properly configured to:

        - Define the HOME_NET (the internal network range: 192.168.18.0/24).
        - Specify the network interface to listen on.
        - Enable preprocessors.
        - Include the relevant rule files.
        - Define output plugins (e.g., logging alerts to a file, database, or sending to a SIEM).
    - **Running Mode:** Snort must be started in NIDS mode (e.g., snort -A full -q -c /etc/snort/snort.conf -i eth0, replacing eth0 with the correct interface).
4. **Vulnerable Target:**
    - **Metasploitable2:** Requires installation and proper network configuration within the internal network. It should be accessible from the Kali Linux machine.
5. **Attacker Machine:**
    - **Kali Linux:** Needs to be able to reach the internal network, specifically the Metasploitable2 machine, to initiate simulated attacks.

By setting up this lab, users can observe Snort in action, generating alerts as the Kali Linux machine interacts with and attempts to compromise the Metasploitable2 server, thereby providing practical experience in network intrusion detection.

5 Installation & Configuration of Snort

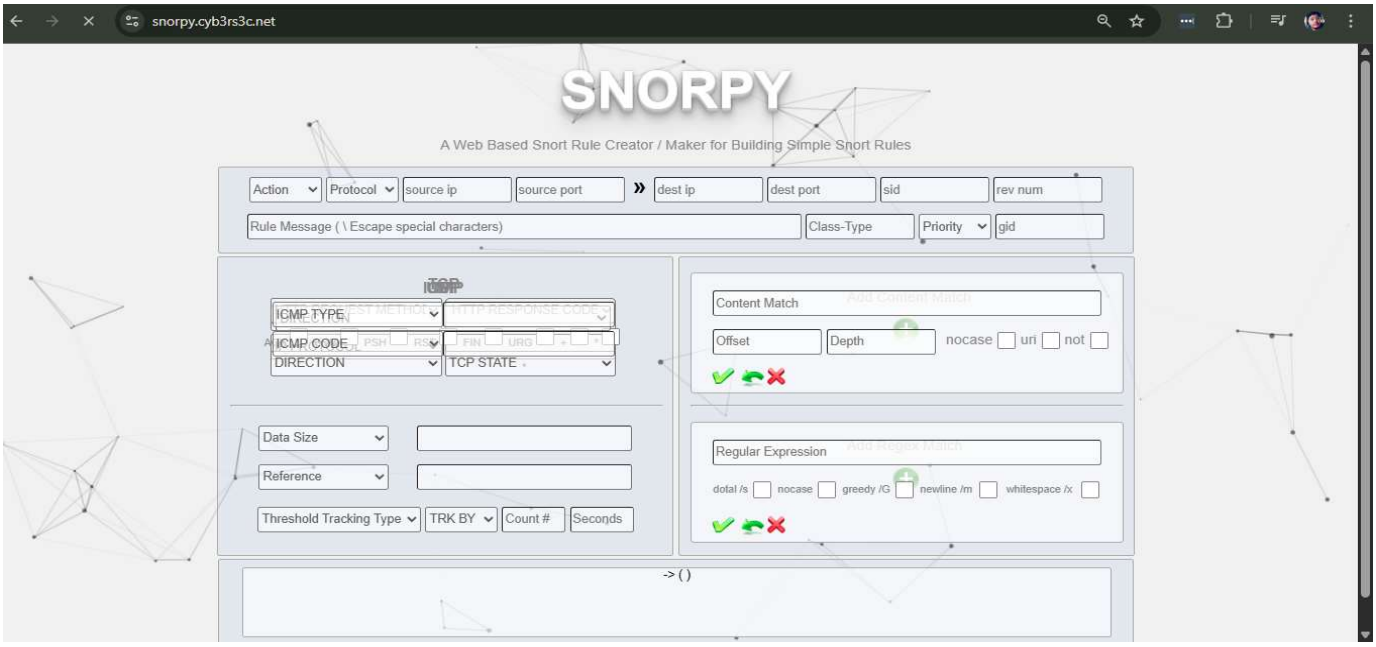
To download the Snort you simply need to **type its URL into your Ubuntu web browser's address bar and press Enter**. The official website for Snort is **snort.org**.



(The official website of Snort)

Once you navigate to **snort.org**, you'll be able to access information about Snort, download the software and rules, find documentation, and explore community resources(as shown in the above image)

Also we need Snorpy (or similar web-based Snort rule creators) because writing Snort rules directly can be complex and prone to errors, especially for new users or when dealing with intricate detection logic that's why open another tab in your browser and type Snorpy and go to the first website that appears in response.



(This is the snorpy web interface)

Benefits of using Snorpy:

1. **Simplifies Rule Creation:** Snort rules have a specific syntax that can be challenging to learn and remember. Snorpy provides a graphical user interface (GUI) that breaks down the rule components (action, protocol, source/destination IPs/ports, content match, etc.) into easy-to-use fields. This eliminates the need to memorize syntax and reduces the chances of typos or formatting errors.
2. **Accelerates Rule Development:** Instead of manually typing out each part of a rule and constantly referring to documentation, Snorpy allows you to quickly select options from dropdowns and input values. This significantly speeds up the process of creating and testing new detection rules.
3. **Reduces Errors:** By guiding the user through the rule-building process with predefined fields and validation, Snorpy helps to minimize common errors that can occur when writing rules manually, leading to more effective and reliable intrusion detection.

Now starting the actual installation part go to your Ubuntu machine's terminal and type **sudo apt-get update && sudo apt-get upgrade -y**

```
sayuj@Ubuntu24:~$ sudo apt-get update && sudo apt-get upgrade -y
[sudo] password for sayuj:
Reading package lists... Done
E: Could not get lock /var/lib/apt/lists/lock. It is held by process 3829 (aptd)
N: Be aware that removing the lock file is not a solution and may break your system.
E: Unable to lock directory /var/lib/apt/lists/
```

sudo apt-get update: This command refreshes the list of available packages from the repositories. It doesn't install or upgrade any software but updates the local index of what's available for installation or upgrade**.**

sudo apt-get upgrade -y: This command installs the newer versions of installed packages based on the updated package list. The -y flag automatically confirms any prompts, avoiding manual intervention.

This step is crucial for maintaining system health and security. It ensures that:

1. You have access to the latest software versions and security patches.
2. Any subsequent software installations (like Snort) fetch the most up-to-date and compatible dependencies.
3. Known vulnerabilities in existing software are addressed, improving overall system security.

After all dependencies and packages are updated download the snort by simply typing

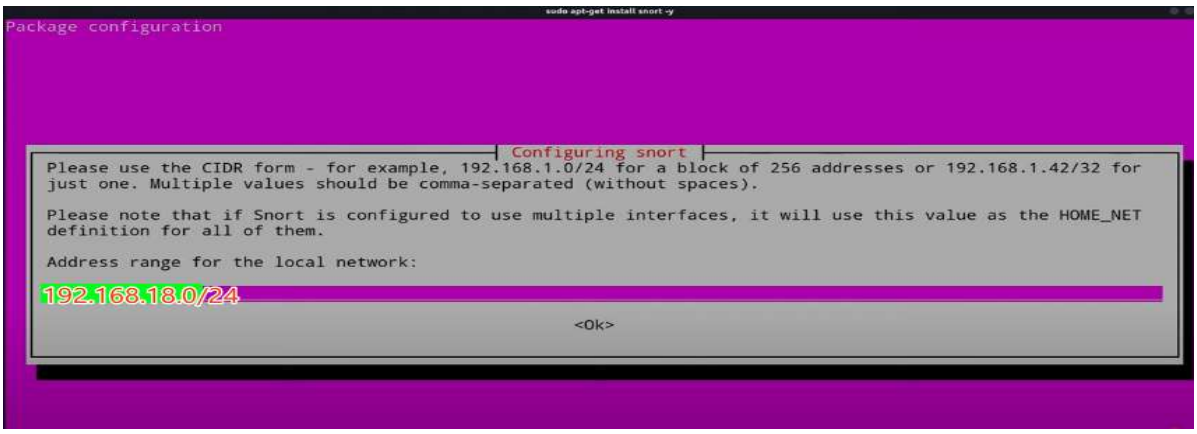
sudo apt-get install snort -y

This command **downloads and installs the Snort package** along with its necessary dependencies (e.g., libdaq2t64, libdumbnet1, oinkmaster, snort-common-libraries). The -y flag again automates the installation confirmation.

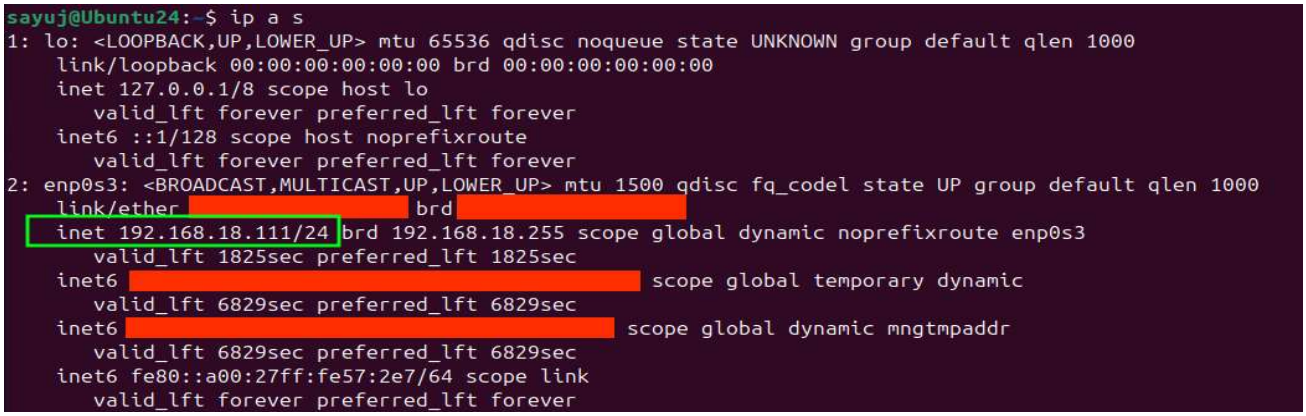
```
sayuj@Ubuntu24:~$ sudo apt-get install snort -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libdaq2t64 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common libnetfilter-queue1 libpcre3 oinkmaster snort-common
  snort-common-libraries snort-rules-default
Suggested packages:
  snort-doc
The following NEW packages will be installed:
  libdaq2t64 libdumbnet1 libluajit-5.1-2 libluajit-5.1-common libnetfilter-queue1 libpcre3 oinkmaster snort
  snort-common snort-common-libraries snort-rules-default
0 upgraded, 11 newly installed, 0 to remove and 8 not upgraded.
Need to get 0 B/2,666 kB of archives.
```

This command is **absolutely essential for deploying Snort**. It:

1. **Installs the core Snort application** itself, making it available on the system.
2. **Fetches all required libraries and components** that Snort needs to function correctly (like the Data Acquisition (DAQ) library for reading packets, and default Snort rules).
3. Sets up Snort to be manageable through the system's package management system, simplifying future updates or removal.



During the configuration an interface will pop-up to your screen(just like the above image).Also check your Ubuntu machine's interface and IP address on a another terminal before setting up the HOME_NET.



Then go back to the configuration terminal and write the IP address range along with the CIDR notation(as shown in the image in the previous page) also you don't have to specify the interface. Just change the address range and hit enter it will resume the installation.

Need for HOME_NET Configuration

Snort needs to distinguish between "internal" (home) traffic and "external" (outside) traffic to effectively detect intrusions. This configuration screen explicitly asks for the "Address range for the local network" using CIDR notation (e.g., 192.168.18.0/24).

By defining HOME_NET:

1. **Contextual Detection:** Snort rules often rely on whether traffic is coming from or going to the home network. For example, a rule might alert if a known attack signature is seen originating *from* the HOME_NET (indicating an internal compromise) or targeting *the HOME_NET* from outside.

2. **Reduced False Positives:** Properly defining HOME_NET helps Snort focus its detection efforts and reduces false positives by not alerting on legitimate internal network activity that might resemble an attack if the context (internal vs. external) wasn't understood.

3. **Rule Logic:** Many Snort rules are written with HOME_NET and EXTERNAL_NET (which is typically defined as any or !$HOME_NET) as variables. Setting HOME_NET correctly ensures these rules function as intended.

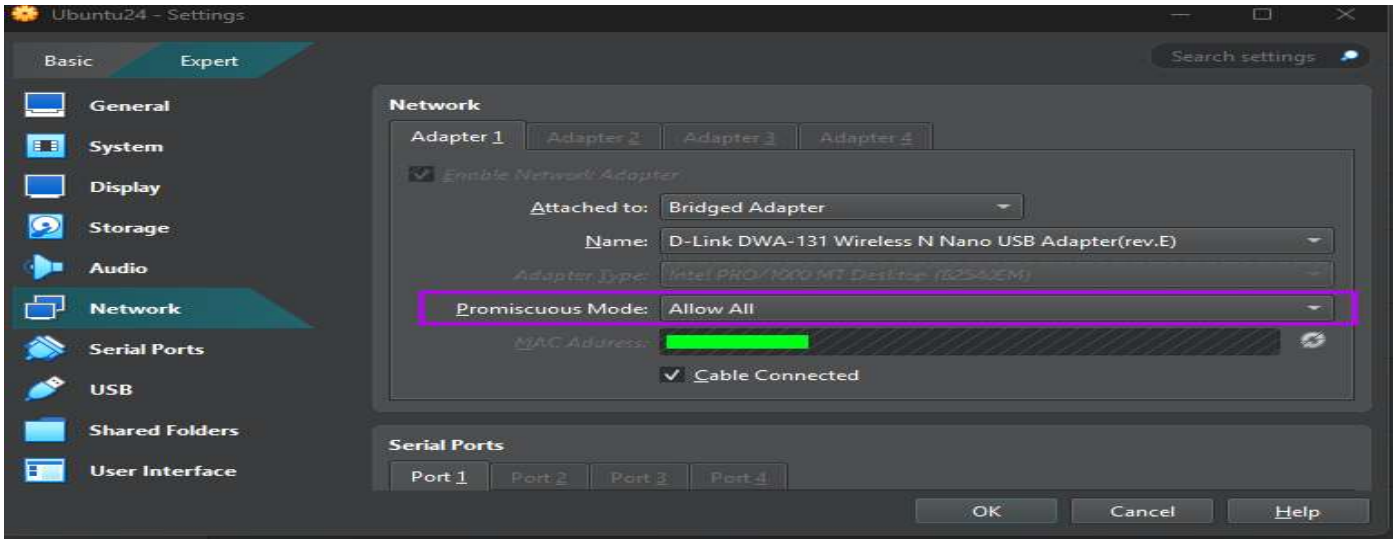
After installation, the **snort --version** command confirms the installed Snort version (e.g., **2.9.7.0 GRE**), verifying that Snort is successfully installed and ready for configuration and use.

Now allow all to the Promiscuous mode is crucial for Snort because it forces the network adapter to **capture all network traffic it sees on the segment**, regardless of whether that traffic is directly addressed to the Snort machine's own MAC address.

Need:

- **Comprehensive Monitoring:** Snort acts as a Network Intrusion Detection System (NIDS). To detect threats, it needs to inspect *all* packets flowing through the monitored network segment, not just those intended for itself.
- **Packet Capture:** Without promiscuous mode, the network adapter would typically drop packets not addressed to its MAC, preventing Snort from analyzing a large portion of the network's communication.

Go to Ubuntu machine's setting in virtualbox and check in the network section if the **Promiscuous mode** is set to allow all or not**.**



Or you can do that too from your terminal by simply typing **sudo ip link set enp0s3 promisc on**

Here instead of **enp0s3** you have to specify your default interface name.

```
> $ sudo ip link set enp0s3 promisc on
```

For Accessing Snort Documentation type **man snort ,** this command is used to access local documentation about Snort's usage, options, and configuration directly from the command line. It's an immediate way to get help without needing internet access.

```
sayuj@Ubuntu24:~$ man snort
sayuj@Ubuntu24:~$
```

After hitting enter it show you an manual page provides **comprehensive information** about how to use Snort. It details every available command-line flag, explains their purpose, and often includes examples. It's an invaluable resource for users to:

- Understand different **Snort modes of operation**.
- Learn how to **configure Snort** for specific tasks (e.g., promiscuous mode, logging, rule files).

- Troubleshoot issues by understanding the meaning of various options.

```
SNORT(8)                          System Manager's Manual                          SNORT(8)

NAME
       Snort - open source network intrusion detection system

SYNOPSIS
       snort  [-bCdDeEfHIMNOpqQsTUvVwWxXy?]  [-A alert-mode ] [-B address-conversion-mask ] [-c rules-file ] [-F bpf-
       file ] [-g group-name ] [-G id ] [-h home-net ] [-i interface ] [-k checksum-mode ]  [-K  logging-mode  ]  [-l
       log-dir  ]  [-L  bin-log-file ] [-m umask ] [-n packet-count ] [-P snap-length ] [-r tcpdump-file ] [-R name ]
       [-S variable=value ] [-t chroot_directory ] [-u user-name ] [-Z pathname ] [--logid id ] [--perfmon-file path-
       name ] [--pid-path pathname ] [--snaplen snap-length ] [--help ] [--version  ]  [--dynamic-engine-lib  file  ]
       [--dynamic-engine-lib-dir directory ] [--dynamic-detection-lib file ] [--dynamic-detection-lib-dir directory ]
       [--dump-dynamic-rules directory ] [--dynamic-preprocessor-lib file ] [--dynamic-preprocessor-lib-dir directory
       ]  [--dynamic-output-lib file ] [--dynamic-output-lib-dir directory ] [--alert-before-pass ] [--treat-drop-as-
       alert  ]  [--treat-drop-as-ignore  ]  [--process-all-events  ]  [--enable-inline-test  ]  [--create-pidfile  ]
       [--nolock-pidfile  ]  [--no-interface-pidfile  ] [--disable-attribute-reload-thread ] [--pcap-single= tcpdump-
       file ] [--pcap-filter= filter ] [--pcap-list= list ] [--pcap-dir= directory ] [--pcap-file= file ] [--pcap-no-
       filter ] [--pcap-reset ] [--pcap-reload ] [--pcap-show ] [--exit-check count ] [--conf-error-out ]  [--enable-
       mpls-multicast  ]  [--enable-mpls-overlapping-ip  ] [--max-mpls-labelchain-len ] [--mpls-payload-type ] [--re-
       quire-rule-sid ] [--daq type ] [--daq-mode mode ] [--daq-var name=value ] [--daq-dir dir ] [--daq-list [dir] ]
       [--dirty-pig ] [--cs-dir dir ] [--ha-peer ] [--ha-out file ] [--ha-in file ] expression
```

Now navigate to the **standard directory on Linux systems where Snort's main configuration files are stored** after installation.

Command: ls **-**l /etc/snort

```
sayuj@Ubuntu24:~$ ls -l /etc/snort
total 360
-rw-r--r-- 1 root root     1281 Apr 20  2022 attribute_table.dtd
-rw-r--r-- 1 root root     3757 Apr 20  2022 classification.config
-rw-r--r-- 1 root root    82469 Apr 19  2024 community-sid-msg.map
-rw-r--r-- 1 root root    23654 Apr 20  2022 file_magic.conf
-rw-r--r-- 1 root root    33339 Apr 20  2022 gen-msg.map
-rw-r--r-- 1 root root      687 Apr 20  2022 reference.config
drwxr-xr-x 2 root root     4096 Jun 25 05:55 rules
-rw-r----- 1 root snort   29773 Apr 19  2024 snort.conf
-rw------- 1 root root      807 Jun 25 05:55 snort.debian.conf
-rw-r--r-- 1 root root     2335 Apr 20  2022 threshold.conf
-rw-r--r-- 1 root root   160606 Apr 20  2022 unicode.map
```

The output displays various files and a directory within /etc/snort, including:

- snort.conf: The **primary configuration file** for Snort.
- rules/: A directory that typically **contains the Snort detection rules**.
- Other .conf and .map files: These are supporting configuration files that define things like classification, preprocessor settings, and Unicode mappings.

Now as you can see in the output we have a snort.conf file we have to edit snort.conf is **absolutely essential** for customizing and properly configuring Snort to perform network intrusion detection effectively within a specific environment. This file is Snort's brain, telling it:

1. **Which networks to monitor (HOME_NET):** As discussed previously, defining the internal network range is crucial for contextual threat detection.
2. **Where to find rules:** It specifies the paths to the Snort rule files that define what Snort should look for.
3. **Which preprocessors to use:** Preprocessors normalize and analyze traffic for anomalies before rules are applied (e.g., detecting fragmented packets).
4. **How to log alerts:** It dictates where Snort should output its alerts (e.g., to a log file, a database, or a SIEM system).

```
sayuj@Ubuntu24:~$ sudo vim /etc/snort/snort.conf
[sudo] password for sayuj:
sudo: vim: command not found
```

To open and edit the **snort.conf** file using the **sudo vim /etc/snort/snort.conf** command, which resulted in a "command not found" error for vim. No worries just open another terminal and run the following command: **sudo apt-get install vim** or **sudo apt-get install vim -y**

Then re-write the **sudo vim /etc/snort/snort.conf** command and it will open the configuration file in vim editor Normal Mode (Command Mode) , you **cannot type text** directly in this mode. it will looks like the image below.

```
#--------------------------------------------------
#   VRT Rule Packages Snort.conf
#
#   For more information visit us at:
#     http://www.snort.org                  Snort Website
#     http://vrt-blog.snort.org/    Sourcefire VRT Blog
#
#     Mailing list Contact:       snort-users@lists.snort.org
#     False Positive reports:     fp@sourcefire.com
#     Snort bugs:                 bugs@snort.org
#
#     Compatible with Snort Versions:
#     VERSIONS : 2.9.20
#
#     Snort build options:
#     OPTIONS : --enable-gre --enable-mpls --enable-targetbased --enable-ppm --enable-perfprofiling --enable-zlib --enab
le-active-response --enable-normalizer --enable-reload --enable-react --enable-flexresp3
#
#     Additional information:
#     This configuration file enables active response, to run snort in
#     test mode -T you are required to supply an interface -i <interface>
#     or test mode will fail to fully validate the configuration and
#     exit with a FATAL error
#--------------------------------------------------
```

Scroll down a little bit lower you will find the HOME_NET variable just as you can see in the image below.

```
  GNU nano 7.2                                  /etc/snort/snort.conf *
###################################################
# Step #1: Set the network variables.  For more information, see README.variables
###################################################

# 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 192.168.18.0/24

# 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

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

This line tells Snort which IP address range constitutes your "home" or **internal network**. This distinction is vital for Snort to:

1. **Contextualize Traffic:** Snort rules often differentiate between traffic originating from or destined for the HOME_NET versus the EXTERNAL_NET (everything else). This helps it understand if an attack is targeting your internal systems or if unusual activity is coming from within.
2. **Reduce False Positives:** By knowing the internal network, Snort can avoid alerting on benign internal traffic that might otherwise trigger a rule designed for external threats.

Now to change it to your home or internal network you have to shifted from Normal Mode to Insert Mode for that **Press i** you'll see -- INSERT -- or INSERT at the bottom of the screen when you are in Insert Mode.

Then change the ipvar HOME_NET any to you internal network range like in my case it's 192.168.18.0/24 just shown in the above image.

Also after completing you can see the other part of the configuration file scrolling down to lower you will find some customizable rule-sets for snort.

```
##################################################
# Step #7: Customize your rule set
# For more information, see Snort Manual, Writing Snort Rules
#
# NOTE: All categories are enabled in this conf file
##################################################

# Note to Debian users: The rules preinstalled in the system
# can be *very* out of date. For more information please read
# the /usr/share/doc/snort-rules-default/README.Debian file

#
# If you install the official VRT Sourcefire rules please review this
# configuration file and re-enable (remove the comment in the first line) those
# rules files that are available in your system (in the /etc/snort/rules
# directory)

# site specific rules
include $RULE_PATH/local.rules
```

After writing or changing the segment press Esc then **:wq** and press enter to save the changes.

So for validating configuration and preventing runtime failures type

sudo snort -T -i enp0s3 -c /etc/snort/snort.conf is used to test the Snort configuration file and rule set for errors before running Snort in a live environment.

```
sayuj@Ubuntu24:~$ sudo snort -T -i enp0s3 -c /etc/snort/snort.conf
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 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 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 37
02 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 82
43 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
    Split Any/Any group = enabled
    Search-Method-Optimizations = enabled
    Maximum pattern length = 20
```

After running it will produce a quiet long output along with some errors if you scroll down you will find those which will look like these now why these errors are showing

```
WARNING: /etc/snort/rules/community-web-php.rules(461) GID 1 SID 100000917 in rule duplicates previous rule. Ignorin
g old rule.

WARNING: /etc/snort/rules/community-web-php.rules(462) GID 1 SID 100000918 in rule duplicates previous rule. Ignorin
g old rule.

WARNING: /etc/snort/rules/community-web-php.rules(463) GID 1 SID 100000919 in rule duplicates previous rule. Ignorin
g old rule.

WARNING: /etc/snort/rules/community-web-php.rules(464) GID 1 SID 100000920 in rule duplicates previous rule. Ignorin
g old rule.

WARNING: /etc/snort/rules/community-web-php.rules(465) GID 1 SID 100000921 in rule duplicates previous rule. Ignorin
g old rule.

WARNING: /etc/snort/rules/community-web-php.rules(466) GID 1 SID 100000922 in rule duplicates previous rule. Ignorin
g old rule.
```

Snort's rule engine requires each rule to have a unique GID and SID combination. When it finds a duplicate, it simply **loads the first instance of that rule it encounters and ignores any subsequent rules with the identical GID:SID**.

Why it happens:

- **Rule Management Issues:** This often occurs when multiple rule sets are used, or when rules are copied or updated without proper de-duplication.
- **Default Behavior:** Snort handles this by prioritizing the first rule found, preventing conflicts but also potentially overlooking a newer or slightly modified version of a rule if it's placed later in the configuration.

In short you can ignore these errors also if you scroll down a lower you can find vital information about the number of rules loaded and a breakdown of rule types based on port and protocol.

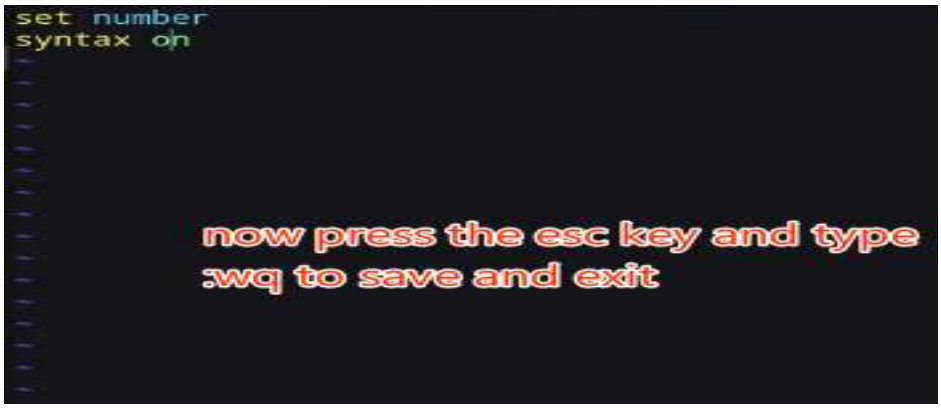
```
4057 Snort rules read
    3383 detection rules
    0 decoder rules
    0 preprocessor rules
3383 Option Chains linked into 949 Chain Headers
+++++++++++++++++++++++++++++++++++++++++++++++++++

+-------------------[Rule Port Counts]---------------------------------------
|               tcp      udp     icmp       ip
|      src      151       18        0        0
|      dst     3306      126        0        0
|      any      383       48       52       22
|       nc       27        8       15       20
|      s+d       12        5        0        0
+------------------------------------------------------------------------------

+-----------------------[detection-filter-config]--------------------------------
| memory-cap : 1048576 bytes
```

In order to remove those error you have comment the snort community rules to do that first you have to edit the **Vim configuration file (.vimrc)** for the root user.

Just type **sudo vim /root/.vimrc** this command on your terminal for customizing Vim Behavior for Root (e.g., enable syntax highlighting, set line numbers, configure tabs/spaces)

```
sudo vim /root/.vimrc
```

It will open the vim editor then you type the two lines as shown in the image for customize the Vim text editor for the root user by enabling line numbers and syntax highlighting, making it easier to read and edit configuration files (like Snort's snort.conf)



Now back to the snort configuration file now after the configuration file opens you can see the line

```
sayuj@Ubuntu24:~$ sudo vim /etc/snort/snort.conf
```

number assigned with each line of that file , so finally we want to commenting out specific rule include lines in snort.conf is necessary to **prevent Snort from loading redundant or conflicting rules that cause "duplicate rule" warnings**, thereby improving Snort's performance, streamlining rule management, and ensuring only the intended detection logic is active. This makes your Snort deployment more efficient and effective.

```
734
735 # dynamic library rules
736 # include $SO_RULE_PATH/bad-traffic.rules
737 # include $SO_RULE_PATH/chat.rules
738 # include $SO_RULE_PATH/dos.rules
739 # include $SO_RULE_PATH/exploit.rules
740 # include $SO_RULE_PATH/icmp.rules
741 # include $SO_RULE_PATH/imap.rules
742 # include $SO_RULE_PATH/misc.rules
743 # include $SO_RULE_PATH/multimedia.rules
744 # include $SO_RULE_PATH/netbios.rules
745 # include $SO_RULE_PATH/nntp.rules
746 # include $SO_RULE_PATH/p2p.rules
747 # include $SO_RULE_PATH/smtp.rules
748 # include $SO_RULE_PATH/snmp.rules
749 # include $SO_RULE_PATH/specific-threats.rules
750 # include $SO_RULE_PATH/web-activex.rules
751 # include $SO_RULE_PATH/web-client.rules
752 # include $SO_RULE_PATH/web-iis.rules
753 # include $SO_RULE_PATH/web-misc.rules
754
755 # Event thresholding or suppression commands. See threshold.conf
756 include threshold.conf
 :597,718s/^/#/
```

To comment out the line just press i , write **:597,718**/^/#/ and enter it will comment the lines now to save the changes press **esc** key write **:wq** and enter it will save the changes. If you run the command **sudo snort -T -i enp0s3 -c /etc/snort/snort.conf** command again it would not shows the warnings anymore.

```
sayuj@Ubuntu24:~$ sudo snort -T -i enp0s3 -c /etc/snort/snort.conf
[sudo] password for sayuj:
Running in Test mode

        --== Initializing Snort ==--
Initializing Output Plugins!
Initializing Preprocessors!
Initializing Plug-ins!
Parsing Rules file "/etc/snort/snort.conf"
PortVar 'HTTP_PORTS' defined :  [ 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 34443:34444 41080 50002 55555 ]
PortVar 'SHELLCODE_PORTS' defined :  [ 0:79 81:65535 ]
PortVar 'ORACLE_PORTS' defined :  [ 1024:65535 ]
PortVar 'SSH_PORTS' defined :  [ 22 ]
PortVar 'FTP_PORTS' defined :  [ 21 2100 3535 ]
PortVar 'SIP_PORTS' defined :  [ 5060:5061 5600 ]
PortVar 'FILE_DATA_PORTS' defined :  [ 80:81 110 143 311 383 591 593 901 1220 1414 1741 1830 2301 2381 2809 3037 3128 37
02 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 82
43 8280 8300 8800 8888 8899 9000 9060 9080 9090:9091 9443 9999 11371 34443:34444 41080 50002 55555 ]
PortVar 'GTP_PORTS' defined :  [ 2123 2152 3386 ]
Detection:
   Search-Method = AC-Full-Q
```

Finally the installation and configuration part is done it’s time for the testing phase.

6 Testing Snort

In order to **verifying Snort's installation version** and then **accessing the local.rules file to add or modify custom Snort detection rules**, which is a fundamental part of tailoring Snort for specific security monitoring needs.

TEST no. 1 ICMP Ping Alert

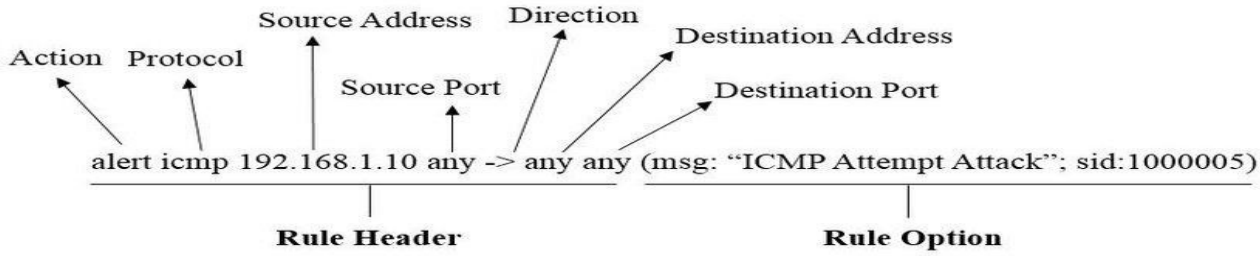
Now type **sudo vim /etc/snort/rules/local.rules** command to create specific rules tailored to their unique network environment, applications, or to detect novel threats. This command allows for the direct creation or modification of these custom detection rules.

```
root@Ubuntu24:/home/sayuj# snort --version

   ,,_     -*> 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

root@Ubuntu24:/home/sayuj# sudo vim /etc/snort/rules/local.rules
```

Snort Rule Format

Snort offers its user to write their own rule for generating logs of Incoming/Outgoing network packets. Only they need to follow the snort rule format where packets must meet the threshold conditions. Always bear in mind that the snort rule can be written by combining two main parts "**the Header**" and "**the Options**" segment.

The header part contains information such as the action, protocol, the source IP and port, the network packet Direction operator towards the destination IP and port, the remaining will be considered in the options part.



Syntax: Action Protocol Source IP Source port -> Destination IP Destination port (options)

Header Fields:-

Action: It informs Snort what kind of action to be performed when it discovers a packet that matches the rule description. There are five existing default job actions in Snort: alert, log, pass, activate, and dynamic are keyword use to define the action of rules. You can also go with additional options which include drop, reject, and sdrop.

Protocol: After deciding the option for action in the rule, you need to describe specific Protocol (IP, TCP, UDP, ICMP, any) on which this rule will be applicable.

Source IP: This part of header describes the sender network interface from which traffic is coming.

Source Port: This part of header describes the source Port from which traffic is coming.

Direction operator ("->", "<>"): It denotes the direction of traffic flow between sender and receiver networks.

Destination IP: This part of header describes the destination network interface in which traffic is coming for establishing the connection.

Destination Port: This part of header describes the destination Port on which traffic is coming for establishing the connection.

Option Fields:

The body for rule option is usually written between circular brackets "()" that contains keywords with their argument and separated by semicolon ";" from another keyword.

There are four major categories of rule options.

General: These options contains metadata that offers information with reference to them.

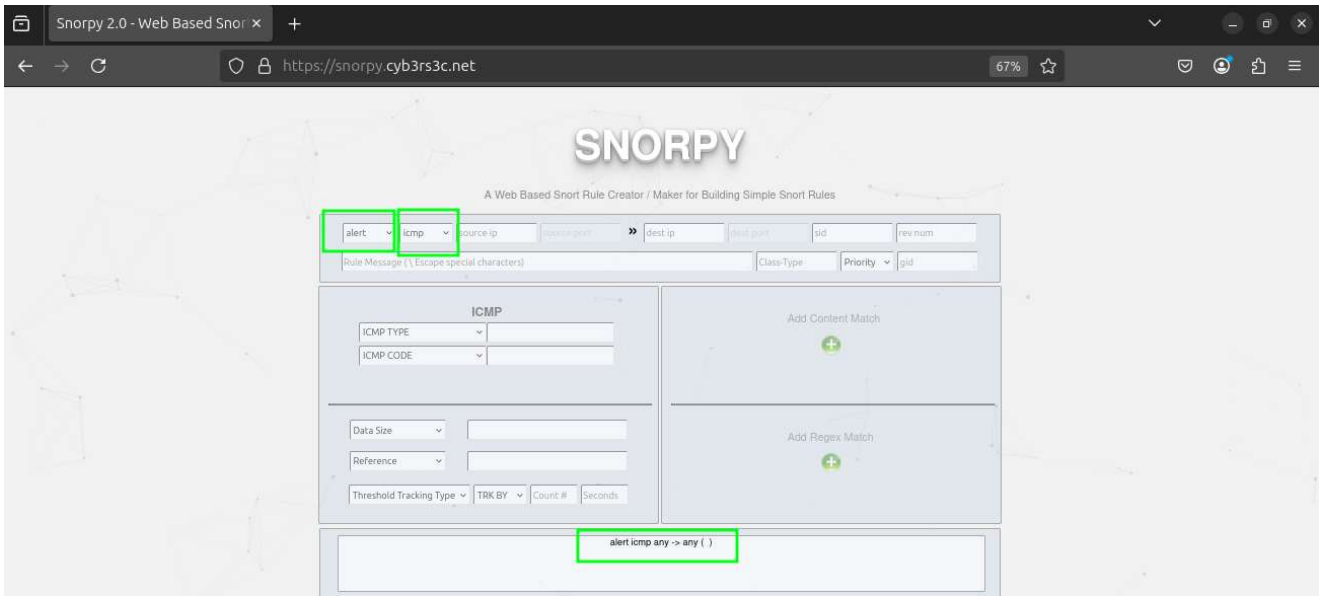
Payload: These options all come across for data contained by the packet payload and can be interconnected.

Non-payload: These options come across for non-payload data.

Post-detection: These options are rule specific triggers that happen after a rule has fired."

General Rule Options (Metadata)

In this article are going to explore more about general rule option for beginners so that they can easily write a basic rule in snort rule file and able to analyst packet of their network. Metadata is part of the optional rule which basically contains additional information of about snort rule that is written with the help of some keywords and with their argument details.



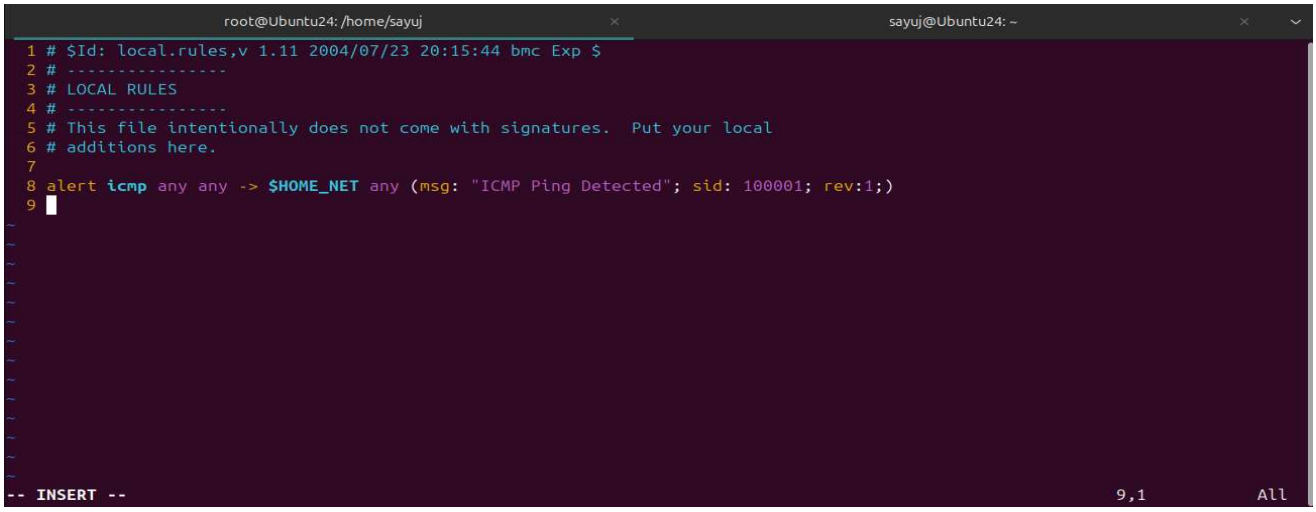
Now go to the main interface of **Snorpy**, alert action being selected (indicating Snort should generate an alert when the rule is matched).

- The icmp protocol being selected (meaning the rule will apply to ICMP traffic).

- A generated rule snippet at the bottom: alert icmp any -> any (). This shows the basic structure of the rule being formed based on the selections made in the GUI.

As shown in the above image, now go back to the snort local rules and type (for typing press i and then write the rule)

alert icmp any any -> $HOME_NET any (msg: "ICMP Ping Detected"; sid: 1000001; rev:1;)

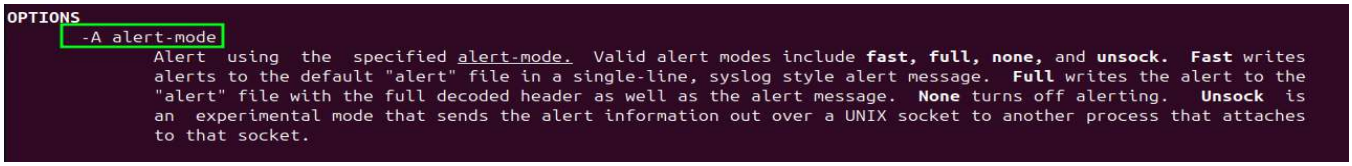


Now this is an creation of a **custom Snort rule** in local.rules to **detect ICMP ping activity** directed at your internal network, primarily serving as a straightforward way to **test and confirm Snort's core detection and alerting functionality**.

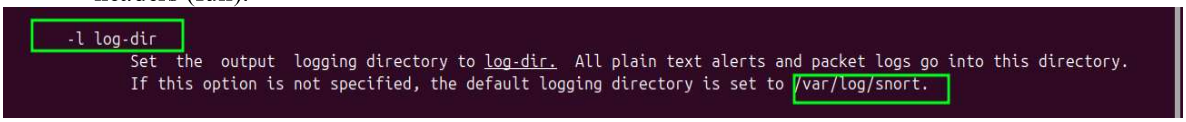
To save this press **esc** key then **:wq** to save the changes.

Now we have to see the manual of Snort for choosing the correct flags which are recommended in our testing or in order to run the Snort to do so type man snort as shown in the image.

It will shows the arious command-line options and flags available for running Snort, providing detailed information on how to control its behavior from there you have to scroll down a little bit



-A alert-mode option : This is needed to specify the desired level of detail in Snort's alerts, allowing you to choose between concise single-line alerts (fast) or more verbose alerts with full decoded headers (full).



-l log-dir option: This option is crucial to define where Snort will store its generated alerts and logged packets. Without it, you wouldn't know where to find the output of Snort's detection efforts.

-i interface option: This is fundamental. Snort must know which network card to listen on to capture and analyze traffic. For instance, in your lab setup, this would be enp0s3.

```
-q     Quiet  operation.  Don't display banner and initialization information. In daemon mode, banner and initializa-
       tion information is not logged to syslog.

-Q     Enable inline mode operation.
```

-q option: This flag is used to suppress verbose startup messages and banners, making Snort's output cleaner, especially when running it as a background daemon or when only alert output is desired.

-Q option: stands for **"Enable inline mode operation"**. This is a critical distinction from -q. When Snort is run in inline mode, it acts as an **Intrusion Prevention System (IPS)** rather than just an Intrusion Detection System (IDS).

To run snort as an NIDS to actively monitoring your network for threats we have to specify some arguments like the image below.

```
root@Ubuntu24:/home/sayuj# sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
```

Now go to your attacker machine to check whether snort is implemented properly or not in our scenario we're using Kali Linux for demonstation now open kali machine and go to the terminal and type ifconfig to clarifies this machine's IP(inet 192.168.18.82).

```
┌──[sayuj@sayuj-virtualbox]─[~/Desktop]
└─$ ifconfig
enp0s3: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.18.82  netmask 255.255.255.0  broadcast 192.168.18.255
        inet6 [redacted]  prefixlen 64  scopeid 0x20<link>
        inet6 [redacted]  prefixlen 64  scopeid 0x0<
global>
        ether 08:00:27:82:b9:e8  txqueuelen 1000  (Ethernet)
        RX packets 53856  bytes 44892389 (42.8 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 4624 (4.5 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Now to detect the ICMP attack launch an ICMP ping requests and replies that's gonna detected by snort.

Command: **ping 192.168.18.111** now this is IP address is for the Ubuntu machine where you configured the Snort after generating the attack go back to the Ubuntu machine.

```
┌──[sayuj@sayuj-virtualbox]─[~/Desktop]
└─$ ping 192.168.18.111
PING 192.168.18.111 (192.168.18.111) 56(84) bytes of data.
64 bytes from 192.168.18.111: icmp_seq=1 ttl=64 time=0.944 ms
64 bytes from 192.168.18.111: icmp_seq=2 ttl=64 time=0.553 ms
64 bytes from 192.168.18.111: icmp_seq=3 ttl=64 time=0.499 ms
64 bytes from 192.168.18.111: icmp_seq=4 ttl=64 time=0.963 ms
64 bytes from 192.168.18.111: icmp_seq=5 ttl=64 time=0.628 ms
64 bytes from 192.168.18.111: icmp_seq=6 ttl=64 time=0.438 ms
64 bytes from 192.168.18.111: icmp_seq=7 ttl=64 time=0.367 ms
64 bytes from 192.168.18.111: icmp_seq=8 ttl=64 time=0.510 ms
64 bytes from 192.168.18.111: icmp_seq=9 ttl=64 time=0.435 ms
64 bytes from 192.168.18.111: icmp_seq=10 ttl=64 time=0.394 ms
64 bytes from 192.168.18.111: icmp_seq=11 ttl=64 time=0.640 ms
64 bytes from 192.168.18.111: icmp_seq=12 ttl=64 time=1.04 ms
64 bytes from 192.168.18.111: icmp_seq=13 ttl=64 time=1.13 ms
64 bytes from 192.168.18.111: icmp_seq=14 ttl=64 time=0.434 ms
64 bytes from 192.168.18.111: icmp_seq=15 ttl=64 time=0.440 ms
64 bytes from 192.168.18.111: icmp_seq=16 ttl=64 time=0.501 ms
64 bytes from 192.168.18.111: icmp_seq=17 ttl=64 time=0.530 ms
64 bytes from 192.168.18.111: icmp_seq=18 ttl=64 time=0.766 ms
64 bytes from 192.168.18.111: icmp_seq=19 ttl=64 time=0.460 ms
64 bytes from 192.168.18.111: icmp_seq=20 ttl=64 time=0.426 ms
64 bytes from 192.168.18.111: icmp_seq=21 ttl=64 time=0.564 ms
```

Now to stop the attack press **ctrl+c** and see the Snort console on your Ubuntu machine.

```
6/27-04:49:29.053755  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:29.053770  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:30.056842  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:30.056871  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:31.072622  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:31.072645  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:32.100892  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:32.100924  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:33.101983  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:33.102010  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:34.109638  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
6/27-04:49:34.109659  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.111 -> 192.168.18.82
6/27-04:49:35.121682  [**] [1:100001:1] ICMP Ping Detected [**] [Priority: 0] {ICMP} 192.168.18.82 -> 192.168.18.111
```

Gotcha!!! Snort's console output, displaying "ICMP Ping Detected" alerts (SID 1000001) for traffic between 192.168.18.82 (Ubuntu/Snort) and 192.168.18.111 (ping target).

This demonstrates Snort's successful detection of the ping traffic, triggering the custom rule and confirming Snort's core functionality.

Now to stop monitoring press **ctrl+c .**

TEST no. 2 SSH Authentication Attempts

Let’s try SSH attemp testing to do that you have to create another rule type open the local rules file again

Command: sudo vim /etc/snort/rules/local.rules

```
root@Ubuntu24:/home/sayuj# sudo vim /etc/snort/rules/local.rules
```

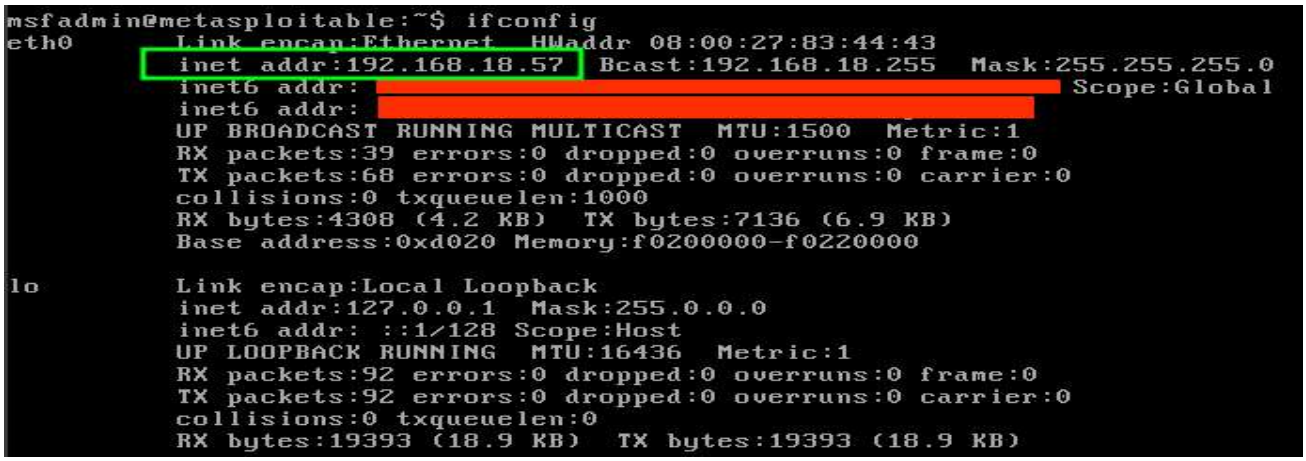
```
root@Ubuntu24: /home/sayuj                                  sayuj@Ubuntu24: ~
 1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
 2 # ----------------
 3 # LOCAL RULES
 4 # ----------------
 5 # This file intentionally does not come with signatures.  Put your local
 6 # additions here.
 7
 8 alert icmp any any -> $HOME_NET any (msg: "ICMP Ping Detected"; sid: 100001; rev:1;)
 9 alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
10
11
```

After opening the local rule file press i and write the rule after writing to save it press esc type :wq and enter.

alert tcp any any -> $HOME_NET 22 (msg: "SSH Authentication Attempt"; sid: 1000002; rev:1;)

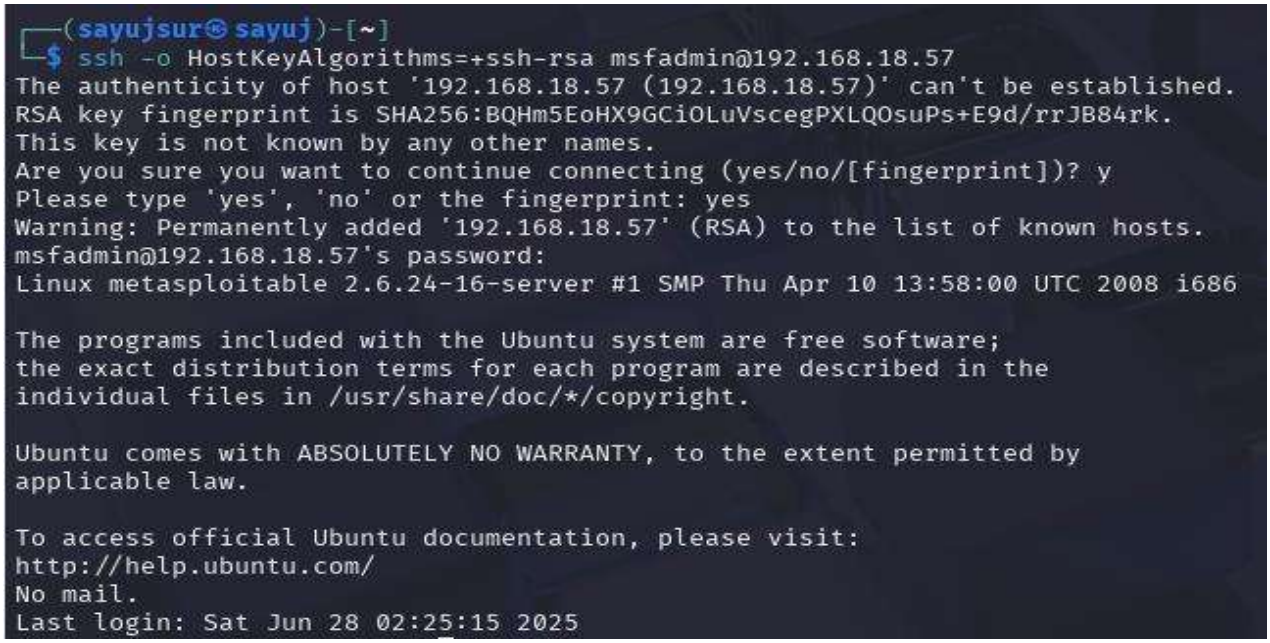
This rule will generate an alert for any TCP traffic destined for port 22 (SSH) on the $HOME_NET, labeled as an **"SSH Authentication Attempt".** These custom rules allow the Snort administrator to tailor the IDS to detect specific activities relevant to their network, beyond the default rule sets.

Now to demonstrate it open your metasploitable machine and check it’s IP address by typing the command **ifconfig** shows that the network interface eth0 on this Metasploitable2 machine as shown in the below image.



inet addr:192.168.18.57: This shows that the network interface eth0 on this Metasploitable2 machine has been assigned the IPv4 address 192.168.18.57.

Now open your Kali machine terminal again for SSH Connection to a Vulnerable Server type the command as shown in the image



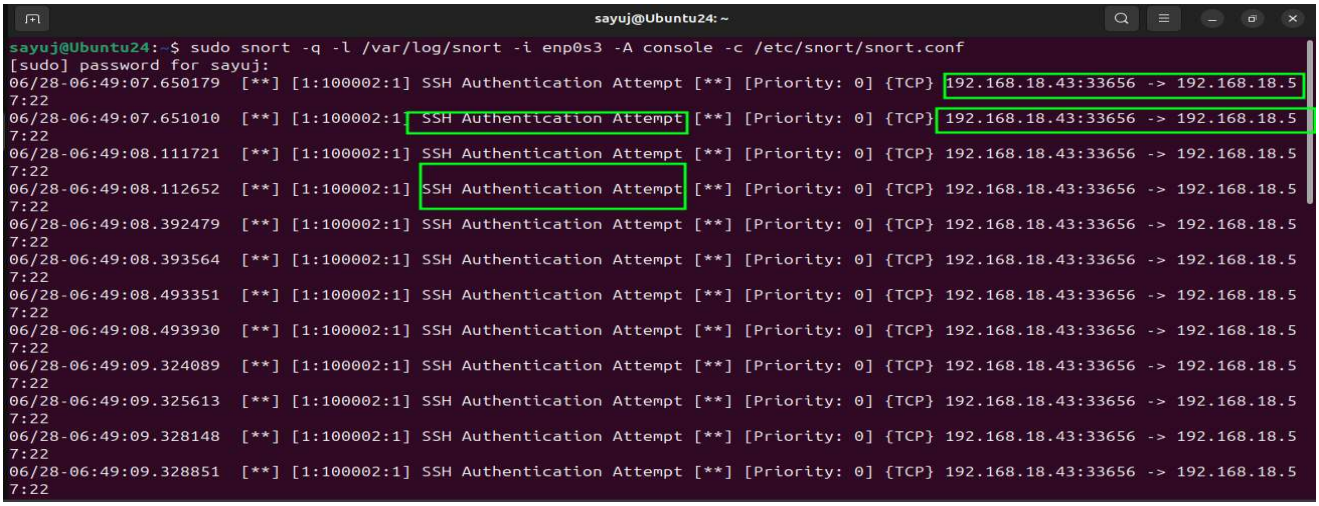
Command: ssh -o HostKeyAlgorithms=+ssh-rsa msfadmin@192.168.18.57

It's used to establish a **Secure Shell (SSH) connection** to the host with IP address 192.168.18.57 using the username msfadmin.

After typing **ssh -o HostKeyAlgorithms=+ssh-rsa msfadmin@192.168.18.57** this command type **y** and the output indicates that the host's authenticity is being verified, and the user accepts it by typing **yes**

Then after hitting enter provide your Metasploitable machine's username and password which is by default **msfadmin** for both.

Go back to the Snort console again in Ubuntu machine, you will see the Snort Alerts for SSH Authentication Attempts as shown in the below image

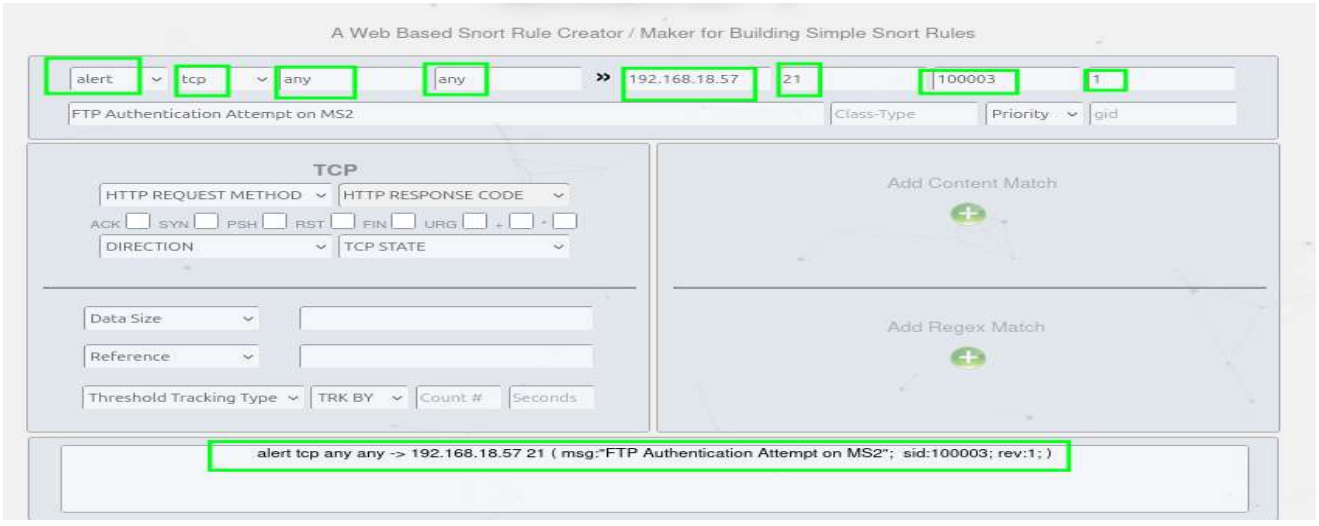


This image is crucial because it **demonstrates Snort successfully detecting and alerting on SSH connection attempts** to the HOME_NET (the protected internal network). This is a critical step in validating the IDS's capability to detect potential malicious activities or policy violations.

To stop Snort monitoring press **ctrl+c** it will stop the monitoring.

TEST no. 3 FTP Authentication Attempt

Now previous two tests are detected successfully now go to the web interface of **Snorpy** for creating an FTP Authentication Attempt Snort Rule



This image demonstrates the process of **graphically constructing a custom Snort rule to detect FTP authentication attempts** specifically targeting the Metasploitable2 server. This rule, once added to local.rules, would enable Snort to generate an alert whenever TCP traffic is observed going to port 21 of 192.168.18.57, helping a network administrator monitor potential brute-force attacks or unauthorized access attempts against the FTP service on that vulnerable machine.

After seleting the parameters you can copy the corresponding Snort rule at the bottom:

alert tcp any any -> 192.168.18.57 21 (msg:"FTP Authentication Attempt on MS2"; sid:1000003; rev:1;).

Also you can write it manually in the snort's custom rule file opens Snort's local.rules file using the vim text editor

Command: sudo vim /etc/snort/rules/local.rules

```
sayuj@Ubuntu24:~$ sudo vim /etc/snort/rules/local.rules
```

```
1 # $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
2 # ----------------
3 # LOCAL RULES
4 # ----------------
5 # This file intentionally does not come with signatures.  Put your local
6 # additions here.
7
8 alert icmp any any -> $HOME_NET any (msg: "ICMP Ping Detected"; sid: 100001; rev:1;)
9 alert tcp any any -> $HOME_NET 22 (msg:"SSH Authentication Attempt"; sid:100002; rev:1;)
10
11 alert tcp any any -> 192.168.18.57 21 ( msg:"FTP Authentication Attempt on MS2"; sid:100003; rev:1; )
```

alert tcp any any -> 192.168.18.57 21 (msg:"FTP Authentication Attempt on MS2"; sid:1000003; rev:1;).

Type the new rule, alert tcp any any -> 192.168.18.57 21 (msg:"FTP Authentication Attempt on MS2"; sid:1000003; rev:1;), is highlighted. This rule is designed to trigger an alert whenever TCP traffic is directed to port 21 (standard for FTP) on the IP address 192.168.18.57 (the Metasploitable2 server).

It will detect and alert on any connection attempts to the FTP service on the target server, allowing for monitoring of potential brute-force or unauthorized access.

To save the changes press **esc** key type **:wq** and hit enter to save it.

Go back to your Ubuntu terminal, to start the Snort once again type

```
sayuj@Ubuntu24:~$ sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf
06/28-06:59:48.065670  [**] [1:100002:1] SSH Authentication Attempt [**] [Priority: 0] {TCP} 192.168.18.43:58198 -> 192.168.18.5
7:22
```

sudo snort -q -l /var/log/snort -i enp0s3 -A console -c /etc/snort/snort.conf is visible, indicating Snort is running as a Network Intrusion Detection System (NIDS) and outputting alerts to the console.

Now to perform it go to your attaker machine or Kali Machine terminal, to the end of an SSH session to 192.168.18.57 (the Metasploitable2 server). The exit command is used, resulting in "Connection to 192.168.18.57 closed."

Then establish an **FTP connection** with the command:

ftp 192.168.18.57

It shows a successful connection to the vsFTPd 2.3.4 service on port 21, a prompt for username and password, and ultimately 230 Login successful (as shown in the below image).

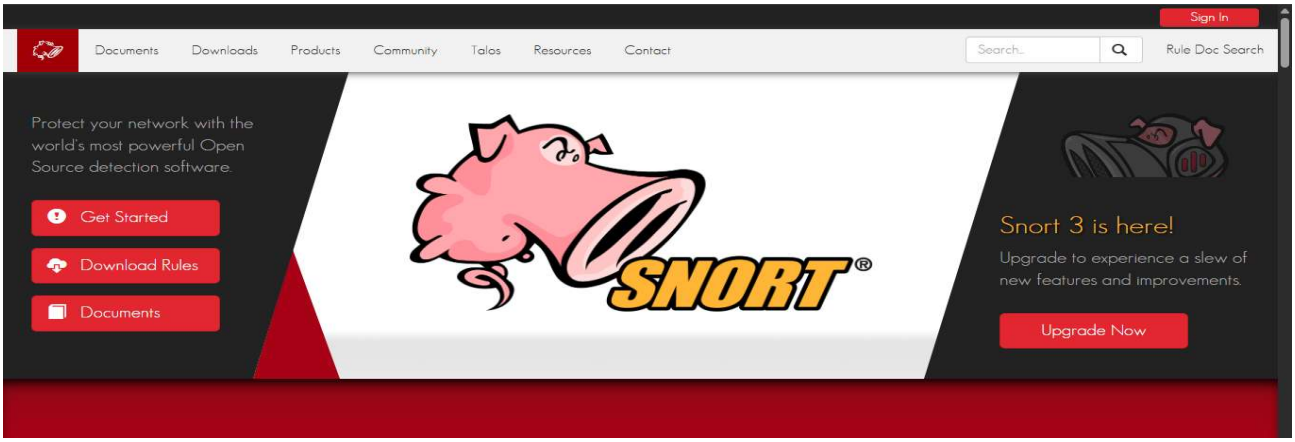
```
┌──(sayujsur㉿sayuj)-[~]
└─$ ssh -o HostKeyAlgorithms=+ssh-rsa msfadmin@192.168.18.57
msfadmin@192.168.18.57's password:
Linux metasploitable 2.6.24-16-server #1 SMP Thu Apr 10 13:58:00 UTC 2008 i686

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To access official Ubuntu documentation, please visit:
http://help.ubuntu.com/
No mail.
Last login: Sat Jun 28 02:33:25 2025 from 192.168.18.43
msfadmin@metasploitable:~$ exit
logout
Connection to 192.168.18.57 closed.

┌──(sayujsur㉿sayuj)-[~]
└─$ ftp 192.168.18.57 21
Connected to 192.168.18.57.
220 (vsFTPd 2.3.4)
Name (192.168.18.57:sayujsur): msfadmin
331 Please specify the password.
Password:
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp>
```

Now the connection is successfully established check the Snort console, in your Ubuntu machine,

```
06/28-07:00:31.490014  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:31.491106  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:31.514285  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:35.931308  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:35.932457  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.147799  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.161016  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.161407  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.162328  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.163790  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
06/28-07:00:40.207625  [**] [1:100003:1] FTP Authentication Attempt on MS2 [**] [Priority: 0] {TCP} 192.168.18.43:57320 -> 192.1
68.18.57:21
```

Success!! This image serves as direct **confirmation that Snort is successfully detecting and alerting on FTP authentication attempts** directed at the vulnerable server, based on the custom rule (SID 1000003) previously added to local.rules. It verifies that Snort is actively monitoring for and identifying specific service-related interactions within the protected network, which is critical for intrusion detection and security monitoring.

Finally Snort works properly to detect and mitigate the attempts properly also you can download predefined rules from the Snort's official website.



6 CONCLUSION AND FUTURE SCOPE

6.1 Conclusion

This project successfully achieved its objective of implementing and configuring a fully operational Network Intrusion Detection System (NIDS) using Snort within a controlled lab environment. Through detailed setup and rigorous testing, Snort proved to be an effective, flexible, and powerful open-source tool capable of performing real-time traffic analysis, packet logging, and signature-based intrusion detection. The ability to define and deploy custom rule sets allowed for precise identification and alerting on simulated malicious network activities, validating the system's core functionality as a proactive security layer. The project demonstrated critical skills in network monitoring, threat analysis, and the practical application of industry-standard security tools, confirming that Snort is a vital component in enhancing an organization’s network security posture.

6.2 Future Scope

To further expand upon the foundation laid by this project, the following areas are recommended for future research and development:

1. **Integration with SIEM:** Integrate the deployed Snort instance with a Security Information and Event Management (SIEM) system (e.g., Splunk or ELK Stack). This integration would centralize log data, provide robust data visualization, and enable security analysts to perform advanced correlation and analysis of threat events across the network.
2. **Conversion to IPS:** Upgrade the current NIDS setup to a Network Intrusion Prevention System (NIPS) by configuring Snort with an inline capability (often achieved using a tool like Snort-Adept or Suricata). This would allow the system to actively block or drop malicious packets, rather than just alerting on them.
3. **Performance Optimization:** Investigate and implement techniques for performance optimization, such as refining the rule set to minimize false positives/negatives and leveraging hardware acceleration (e.g., multi-threading) to ensure high throughput monitoring in high-volume production network environments.
4. **Anomaly-Based Detection:** Explore and integrate anomaly detection methods alongside the current signature-based approach. This involves utilizing machine learning models to baseline normal network behavior, allowing the NIDS to detect zero-day threats or sophisticated intrusions that do not match existing signatures.