

Snort Intrusion Detection System Implementation

-VENUKAMATCHI P

Introduction:

In the realm of cyber security, understanding and implementing Intrusion Detection Systems (IDS) is highly essential for safeguarding networks against numerous threats. This project will give a detailed understanding of Snort, an open-source IDS, on an Ubuntu host machine, with the objective of detecting and responding to attacks initiated from a Kali Linux attacker machine. The project commences with the meticulous configuration of both the Ubuntu host machine and the Kali Linux attacker machine. This project is a balanced one where it is focused on both defending and attacking. The main goal of the project is to give awareness that we have to break the intruder's nose by using most powerful tool snort and secure our organisation. Let's dive deep into it!

Setting Up Ubuntu and Kali

Step 1: Download VirtualBox:

Download VirtualBox from the official website:

<https://download.virtualbox.org/virtualbox/7.0.18/VirtualBox-7.0.18-162988-Win.exe>

Step 2: Download Ubuntu and Kali:

1. Download Ubuntu from the official website: <https://ubuntu.com/download/desktop> (Kindly visit the website for updated version)
2. Download Kali from the official website: <https://cdimage.kali.org/kali-2024.1/kali-linux-2024.1-installer-amd64.iso> (Kindly visit the website for updated version)

Step 3: Install VirtualBox:

1. Run the VirtualBox installer that you downloaded in step 1 and follow the installation instructions. Once installed, launch VirtualBox.
2. Do it one by one

Step 4: Create a New Virtual Machine:

1. Click on the "New" button in the VirtualBox interface to create a new virtual machine.
2. Enter a name for your virtual machine (both Ubuntu and Kali), choose the type as Linux, and select **Ubuntu (64-bit)** as the version and for Kali select **Debian (64-bit)** as the version.
3. Choose the folder where you like.
4. Make sure you select correct iso file (ex: **.iso**)

Step 5: Allocate Memory:

1. Choose the amount of RAM to allocate to the virtual machine.
2. It's recommended to allocate **10GB** for *Ubuntu* and **7GB** for *Kali*.

Step 6: Allocate Hardware:

1. It's highly recommended to allocate 11GB for Base Memory for both VM's.
2. Eventually you have to set the Processor to 4 CPU which is minimum for both VM's.

NOTE: - If you face freeze issue in both VM means increase the CPU

Step 7: Allocate Hard Disk:

1. Minimum you have to give 20GB hard disk size for both VM's.
2. The hard disk file type should be VDI (Virtual Disk Image) for both VM's.

Step 8: Ubuntu & Kali Setup:

Follow the on-screen prompts to set up Ubuntu & Kali, including creating a user account and configuring network settings if necessary.

NOTE: - Highly Recommended Video <https://youtu.be/wX75Z-4MEoM?feature=shared>

Setting Up Snort in Ubuntu

Step 1: Installation:

*NOTE: - If you are new to Linux means kindly execute the commands in **Terminal***

1. Need to do before every Installation:

```
venukamatchi@ubuntu: $ sudo apt-get update  
venukamatchi@ubuntu: $ sudo apt-get upgrade
```

2. Now Install **Snort**:

```
venukamatchi@ubuntu: $ sudo apt-get snort
```

Just press enter to all

Step 2: Configuration:

1. **Vim** installation

```
venukamatchi@ubuntu: $ sudo apt-get vim
```

2. Go into the **Terminal** and type

```
venukamatchi@ubuntu: $ sudo vim /etc/snort/snort.conf
```

NOTE: - Every downloaded project will be there in **etc** folder only

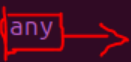
Step 3: Vim Configuration:

NOTE: - New to vim means refer to this <https://youtu.be/XguBRi4TDNc?feature=shared>

```
# 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 overridden 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 any
# 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
```



Remove any and insert your IP with Subnet Mask

To know your IP with Subnet Mask follow these steps,

```
venukamatchi@ubuntu: $ ip addr
```

You will find your IP with subnet mask (ex: **196.145.202.112/24**)

Rule Configuration & Attack Execution

Rule Configuration: Effective IDS operation hinges on precise rule crafting and implementation. In this project, meticulous attention is given to configuring rules for optimizing Snort's detection capabilities. Leveraging Snort's rule syntax, custom rules encapsulate specific attack signatures. Each rule serves as a vigilant guardian, triggering alerts upon detecting anomalous network activity. Tuned to the network's nuances, they strike a balance between sensitivity and specificity, aiming to minimize false positives while maximizing detection efficacy.

Attack Execution: Once the rule framework is established, focus shifts to executing attack tactics to test network defences. The Kali Linux attacker machine plays a central role, launching various simulated attacks to breach the Ubuntu host's defences. These attacks range from basic exploits to sophisticated evasion tactics, aiming to evade detection and compromise network integrity. Each attack serves as a litmus test, assessing the IDS solution's robustness and the efficacy of configured rules in thwarting threats.

Interplaying rule configuration and attack execution provides first-hand insights into offense-defence dynamics in cyber security. Snort's rule framework and adversary tactics illustrate the cat-and-mouse game in network security. Navigating attack vectors and detection mechanisms hones analytical skills, deepening understanding of modern cyber threats. Let's move on into the topic

Some of the Default Commands before Every Rule:-

Adding or updating rules commands,

```
venukamatchi@ubuntu: $ sudo vim /etc/snort/rules/local.rules
```

Command to see the Output,

```
venukamatchi@ubuntu: $ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
```

Code Explanation:-

- ❖ sudo: Runs the command with administrative or super privileges.
- ❖ snort: The command to run Snort.
- ❖ -q: Makes Snort run quietly, showing less output.
- ❖ -i enp0s3: Tells Snort to monitor network traffic on a specific network interface (enp0s3).
- ❖ -l /var/log/snort: Sets the directory where Snort will save its log files (in this case, /var/log/snort).
- ❖ -A console: Specifies that Snort should show alerts on the console or terminal.
- ❖ -c /etc/snort/snort.conf: Points to the configuration file (/etc/snort/snort.conf) that contains all the settings and rules for Snort.

TCP Port

In Ubuntu:

1. `alert tcp any any -> $HOME_NET any`

- ❖ This part of the rule triggers an alert (`alert`) for TCP traffic (`tcp`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and any destination port (`any`).

2. `(msg: "TCP Port Scan Detected"; flags:S; threshold: type threshold, track by_src, count 5, seconds 60; sid:100001;)`

- ❖ This segment provides additional information about the alert.
- ❖ `msg: "TCP Port Scan Detected";` Specifies the message displayed when the alert is triggered, indicating a TCP port scan detection.
- ❖ `flags:S%;` Specifies that the SYN flag should be set in the TCP header, with `%` acting as a wildcard for any other flags.
- ❖ `threshold: type threshold, track by_src, count 5, seconds 60;;` Sets a threshold for triggering the alert. It activates when 5 packets with the same source IP address are seen within a 60-second window.
- ❖ `sid:100001;;` Assigns a unique ID (100001) to the rule, allowing for easy identification and management within Snort.

In Kali:

```
(venukamatchi@kali)-[~]  
$ nmap 192.168.237.151  
Starting Nmap 7.94 ( https://nmap.org ) at 2024-05-11 13:19 MDT  
Note: Host seems down. If it is really up, but blocking our ping probes, try  
-Pn  
Nmap done: 1 IP address (0 hosts up) scanned in 3.04 seconds
```

Now, the **Snort** started to block all type of ping from other network. So, our kali machine can't get connected with **Ubuntu**.

Log by Snort:

```
venukamatchi@venukamatchi:~$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/12-00:49:47.827319  ** [1:100001:0] TCP Port Scan Detected  ** [Priority: 0] {TCP} 192.168.237.74:18383 -> 192.168
.237.151:443
05/12-00:49:48.827232  ** [1:100001:0] TCP Port Scan Detected  ** [Priority: 0] {TCP} 192.168.237.74:18385 -> 192.168
.237.151:80
05/12-00:49:49.831199  ** [1:100001:0] TCP Port Scan Detected  ** [Priority: 0] {TCP} 192.168.237.74:18385 -> 192.168
.237.151:80
05/12-00:49:50.834023  ** [1:100001:0] TCP Port Scan Detected  ** [Priority: 0] {TCP} 192.168.237.74:18384 -> 192.168
.237.151:443
Activate Windows
Go to Settings to activate Win
```

Event Timestamp:

The date and time (05/12-00:49:47.827319) when the event was detected.

Alert Information:

- ❖ A TCP port scan attack has been detected.
- ❖ The alert has a unique ID (1:100001:0).
- ❖ It's labelled "TCP Port Scan Detected" with a priority level of 0.

Source and Destination Details:

- ❖ Source IP address and port: 192.168.237.74:18383.
- ❖ Destination IP addresses and ports: 192.168.237.151:443 and 192.168.237.151:80.
- ❖ These show which devices and ports were involved in the port scan.

Repetition:

- ❖ Similar lines follow, indicating the same attack pattern but with different source ports used in the port scan.

SSH

In Ubuntu:

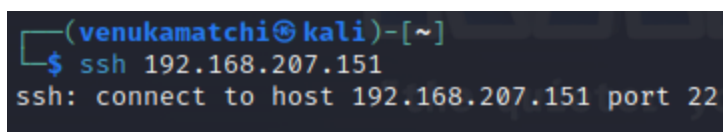
1. `alert tcp any any -> $HOME_NET 22`

- ❖ This part of the rule triggers an alert (`alert`) for TCP traffic (`tcp`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and 22 is the destination port.

2. `(msg: "SSH Connection Detected"; sid: 10110102; rev:1;)`

- ❖ `msg: "SSH Connection Detected";`: Specifies the message displayed when the alert is triggered, indicating a TCP port scan detection.
- ❖ `sid: 10110102;`: Assigns a unique ID (10110102) to the rule, allowing for easy identification and management within Snort.
- ❖ `rev:1;`: The revision number of the rule.

In Kali:



```
(venukamatchi@kali)-[~]  
$ ssh 192.168.207.151  
ssh: connect to host 192.168.207.151 port 22
```

Now, we got the **SSH** connection to the target machine from attacker machine.

Log by Snort:

```
venukamatchi@venukamatchi:/etc/snort$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/13-21:04:04.096323  [**] [1:10110102:1] SSH Connection Detected [**] [Priority: 0] {TCP} 192.168.207.162:43132 -> 192
.168.207.151:22
05/13-21:04:12.792731  [**] [1:10110102:1] SSH Connection Detected [**] [Priority: 0] {TCP} 192.168.207.162:43012 -> 192
.168.207.151:22
```

1. Timestamp:

- ❖ Shows when the event occurred: May 13th, at 9:04:04 PM.

2. Alert Message:

- ❖ Indicates an SSH (Secure Shell) connection was detected.
- ❖ Includes a unique ID (1:10110102:1) for the Snort rule, which can be used for more info.
- ❖ Specifies it's using the TCP protocol.

3. Source and Destination Details:

- ❖ Source IP: 192.168.207.162, Source Port: 43132.
- ❖ Destination IP: 192.168.207.151, Destination Port: 22 (default SSH port).

4. Summary:

- ❖ Snort detected two SSH connection events.
- ❖ Both connections are from 192.168.207.162 to 192.168.207.151.

FTP

In Ubuntu:

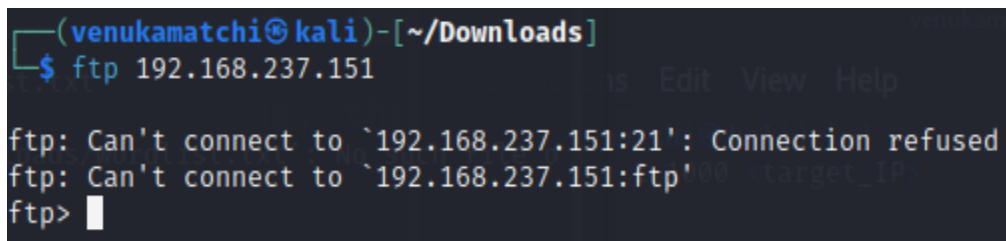
1. `alert tcp any any -> $HOME_NET 21`

- ❖ This part of the rule triggers an alert (`alert`) for TCP traffic (`tcp`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and 21 is the destination port.

2. `(msg: "FTP Connection Detected"; sid: 1000004; rev:1;)`

- ❖ `msg: "FTP Connection Detected";`: Specifies the message displayed when the alert is triggered, indicating a TCP port scan detection.
- ❖ `sid: 1000004;`: Assigns a unique ID (1000004) to the rule, allowing for easy identification and management within Snort.
- ❖ `rev:1;`: The revision number of the rule.

In Kali:



```
(venukamatchi@kali)-[~/Downloads]
$ ftp 192.168.237.151
ftp: Can't connect to `192.168.237.151:21': Connection refused
ftp: Can't connect to `192.168.237.151:ftp': Connection refused
ftp>
```

Now, the **Snort** starts to block the connection.

Log by Snort:

```
venukamatchi@venukamatchi:~$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
[sudo] password for venukamatchi:
05/12-02:04:23.427891  ** [1:1000004:0] FTP Connection Attempt ** [Priority: 0] {TCP} 192.168.237.74:49270 -> 192.168.237.151:21
05/12-02:04:23.929117  ** [1:1000004:0] FTP Connection Attempt ** [Priority: 0] {TCP} 192.168.237.74:49270 -> 192.168.237.151:21
05/12-02:04:24.430319  ** [1:1000004:0] FTP Connection Attempt ** [Priority: 0] {TCP} 192.168.237.74:49270 -> 192.168.237.151:21
05/12-02:04:24.932398  ** [1:1000004:0] FTP Connection Attempt ** [Priority: 0] {TCP} 192.168.237.74:49270 -> 192.168.237.151:21
05/12-02:04:25.435434  ** [1:1000004:0] FTP Connection Attempt ** [Priority: 0] {TCP} 192.168.237.74:49270 -> 192.168.237.151:21
```

1. Timestamp:

- ❖ Shows when the event occurred: May 12th, at 2:04:23 AM.

2. Alert Message:

- ❖ Indicates an FTP (File Transfer) connection was detected.
- ❖ Includes a unique ID (1:1000004:0) for the Snort rule, which can be used for more info.
- ❖ Specifies it's using the TCP protocol.

3. Source and Destination Details:

- ❖ Source IP: 192.168.237.74, Source Port: 49270.
- ❖ Destination IP: 192.168.237.151, Destination Port: 21 (default FTP port).

4. Summary:

- ❖ Snort detected two SSH connection events.
- ❖ Both connections are from 192.168.237.162 to 192.168.237.151.

SQL

In Ubuntu:

1. `alert tcp any any -> any any`

- ❖ This part of the rule triggers an alert (`alert`) for TCP traffic (`tcp`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`any`) and any is the destination port (`any`).

2. `(msg: "SQL Injection Detected"; sid: 100001; rev:1;)`

- ❖ `msg:"SQL Injection Detected"` : This is the message that will be displayed when the rule is triggered. `sid: 100001;`
- ❖ `content: "SELECT"; nocase;` : This is looking for the string "SELECT" in the network traffic, the `nocase` keyword means it's case insensitive.
- ❖ `content: "FROM"; nocase;` : This is looking for the string "FROM" in the network traffic, again case insensitive.
- ❖ `sid:100001`: This is the unique identifier for the rule.

In Kali:

```
(venukamatchi@kali)-[~]
$ echo "SELECT * FROM users" | nc 192.168.207.151 80
HTTP/1.1 400 Bad Request
Date: Mon, 13 May 2024 20:24:26 GMT
Server: Apache/2.4.58 (Ubuntu)
Content-Length: 301
Connection: close
Content-Type: text/html; charset=iso-8859-1

<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>400 Bad Request</title>
</head><body>
<h1>Bad Request</h1>
<p>Your browser sent a request that this server could not understand.<br />
</p>
<hr>
<address>Apache/2.4.58 (Ubuntu) Server at 127.0.1.1 Port 80</address>
</body></html>
```

Log by Snort:

```
venukamatchi@venukamatchi:/etc/snort$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/14-01:54:26.155423  ** [1:100001:0] SQL Injection Detected ** [Priority: 0] {TCP} 192.168.207.162:59292 -> 192.168.207.151:80
```

1. Event Timestamp:

- ❖ Shows when the event occurred: May 14th, at 01:54:26 (1:54 AM).

2. Alert Information:

- ❖ Indicates an SQL Injection attack was detected.
- ❖ Includes a unique ID (1:100001:0) for the Snort rule, useful for identifying the alert.
- ❖ Labelled "SQL Injection Detected" with a priority level of 0.

3. Source and Destination Details:

- ❖ Source IP: 192.168.207.162, Source Port: 59292.
- ❖ Destination IP: 192.168.207.151, Destination Port: 80.
- ❖ Indicates the source is attempting to inject SQL commands into the web server running on the destination IP and port.

4. Repetition:

- ❖ No similar lines follow, but if there were, it would suggest the same attack pattern from the same source IP address.

NMAP

In Ubuntu:

3. `alert tcp any any -> $HOME_NET any`

- ❖ This part of the rule triggers an alert (`alert`) for TCP traffic (`tcp`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and `any` is the destination port.

4. `(msg: "Scan Detected"; sid: 10002001; rev:1;)`

- ❖ `msg: "Scan Detected";`: Specifies the message displayed when the alert is triggered, indicating a TCP port scan detection.
- ❖ `sid: 10002001;`: Assigns a unique ID (10002001) to the rule, allowing for easy identification and management within Snort.
- ❖ `rev:1;`: The revision number of the rule.

In Kali:

```
(venukamatchi@kali)-[~/Desktop]
$ sudo nmap 192.168.225.151
Starting Nmap 7.94 ( https://nmap.org ) at 2024-05-14 19:05:00
Nmap scan report for 192.168.225.151
Host is up (0.0011s latency).
Not shown: 999 filtered tcp ports (no-response)
PORT      STATE SERVICE
12345/tcp  closed netbus
MAC Address: 08:00:27:AC:A7:15 (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 17.38 seconds
```

Log by Snort:

```
venukamatchi@venukamatchi:/etc/snort$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/11-21:01:54.293236  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35013 -> 192.168.145.76:443
05/11-21:01:59.295985  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35012 -> 192.168.145.76:443
05/11-21:02:02.150646  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35015 -> 65.109.127.181:3333
05/11-21:02:04.326327  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:720d:6574:9ae0:35029 -> 2606:4700:83b5:c0b6:59fd:797:b2f4:efeb:443
05/11-21:02:07.295744  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35012 -> 192.168.145.76:443
05/11-21:02:15.143437  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35004 -> 171.79.46.70:7680
05/11-21:02:28.952867  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35044 -> 34.107.243.93:443
05/11-21:02:39.611685  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35047 -> 65.109.127.181:3333
05/11-21:02:48.720342  ** [1:10002000:0] Scan Detected ** [Priority: 0] {TCP} 192.168.237.74:35049 -> 65.109.127.181:3333
```

1. Timestamp:

- ❖ Indicates when the event occurred: May 11th, at 21:01:54 (9:01 PM).

2. Alert Message:

- ❖ "Scan Detected" message alerts to scanning activity.
- ❖ Signature ID (1:10002000:0) corresponds to a specific rule in Snort's rule set.
- ❖ Priority level is 0, the lowest priority.

3. Protocol Used:

- ❖ Detected activity used the Transmission Control Protocol (TCP).

4. Source and Destination Details:

- ❖ Source IP: 192.168.237.74, Source Port: 35013.
- ❖ Destination IP: 192.168.145.76, Destination Port: 443.
- ❖ Shows the source and destination of the scanning activity.

ICMP

In Ubuntu:

1. `alert icmp any any -> $HOME_NET any`

- ❖ This part of the rule triggers an alert (`alert`) for ICMP traffic (`ICMP`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and `any` is the destination port.

2. `(msg: "ICMP Attack Detected"; sid: 10001; rev:1;)`

- ❖ `msg: "ICMP Attack Detected";`: Specifies the message displayed when the alert is triggered, indicating a ICMP port scan detection.
- ❖ `sid: 10001;`: Assigns a unique ID (10001) to the rule, allowing for easy identification and management within Snort.
- ❖ `rev:1;`: The revision number of the rule

In kali:

```
(venukamatchi@kali)-[~]  
└─$ ping 192.168.145.76  
PING 192.168.145.76 (192.168.145.76) 56(84) bytes of data.
```

Log by Snort:

```
05/11-20:57:49.840153 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:50.869497 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:51.887329 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:52.916778 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:53.936222 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:54.971378 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:55.982589 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:57.006229 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:58.031513 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:57:59.054294 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:00.077556 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:01.101996 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:02.125785 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:03.163634 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:04.174198 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:05.197833 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:06.238237 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:07.245582 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:08.106782 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {IPV6-ICMP} fe80::f824:e8ff:fec5:4ed1 ->
2401:4900:3600:d8c1:cd8e:720d:6574:9ae0
05/11-20:58:08.106784 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {IPV6-ICMP} 2401:4900:3600:d8c1:cd8e:720
d:6574:9ae0 -> fe80::f824:e8ff:fec5:4ed1
05/11-20:58:08.268959 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:09.282073 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {IPV6-ICMP} fe80::ffff:ffff:ffff -> ff02
::2
05/11-20:58:09.293147 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 192.168.237.74 -> 192.168.145.76
05/11-20:58:09.318952 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {IPV6-ICMP} fe80::8000:f227:d7ae:34f ->
fe80::ffff:ffff:ffff
05/11-20:58:09.332863 [**] [1:10001:1] ICMP Attack detected [**] [Priority: 0] {ICMP} 103.233.216.18 -> 192.168.237.74
Go to Settings to activate Win
```

1. Event Timestamp:

- ❖ The attacks were detected within the time frame from 20:57:49 to 20:58:09 on May 11.

2. Alert Information:

- ❖ The log shows a series of ICMP (Internet Control Message Protocol) attacks detected by the Snort IDS.

3. Source and Destination Details:

- ❖ Source IP address of the attacks: 192.168.237.74.
- ❖ Target IP address: 192.168.145.76.

4. Repetition:

- ❖ Multiple ICMP attacks were detected within the specified time frame.

HTTP REQUEST

In Ubuntu:

1. `alert tcp any any -> any 80`

- ❖ This part of the rule triggers an alert (`alert`) for ICMP traffic (`ICMP`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and any is the destination port.

2. `(msg: "HTTP Request Detected"; content: "GET"; http_method; sid: 10000002; rev:1;)`

- ❖ `msg: "HTTP Request Detected";` This is the message that will be displayed when the rule matches.
- ❖ `sid: 10000002;` Assigns a unique ID (10000002) to the rule, allowing for easy identification and management within Snort.
- ❖ `http_method:` This option specifies that the rule should inspect the HTTP method (e.g., GET, POST, PUT, etc.) in the packet.
- ❖ `sid: 10000002:` This is the rule ID, which is a unique identifier for the rule.
- ❖ `rev:1;` The revision number of the rule

In Kali:-

```
(venukamatchi@kali)-[~]
$ for i in {1..10}; do curl http://192.168.207.151; done
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3
.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
  <!--
    Modified from the Debian original for Ubuntu
    Last updated: 2022-03-22
    See: https://launchpad.net/bugs/1966004
  -->
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <title>Apache2 Ubuntu Default Page: It works</title>
    <style type="text/css" media="screen">
  * {
    margin: 0px 0px 0px 0px;
    padding: 0px 0px 0px 0px;
  }
```

Log by Snort:

```
venukamatchi@venukamatchi:/etc/snort$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/13-21:25:32.492817  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:58712 -> 192.1
68.207.151:80
05/13-21:27:46.398886  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40884 -> 192.1
68.207.151:80
05/13-21:27:46.419555  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40900 -> 192.1
68.207.151:80
05/13-21:27:46.440367  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40904 -> 192.1
68.207.151:80
05/13-21:27:46.458769  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40920 -> 192.1
68.207.151:80
05/13-21:27:46.479191  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40932 -> 192.1
68.207.151:80
05/13-21:27:46.516724  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40938 -> 192.1
68.207.151:80
05/13-21:27:46.536640  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40948 -> 192.1
68.207.151:80
05/13-21:27:46.555696  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40954 -> 192.1
68.207.151:80
05/13-21:27:46.572397  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40956 -> 192.1
68.207.151:80
05/13-21:27:46.611336  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:40970 -> 192.1
68.207.151:80
05/13-21:28:05.397125  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:43176 -> 192.1
68.207.151:80
05/13-21:28:05.413314  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:43178 -> 192.1
68.207.151:80
05/13-21:28:05.431372  [**] [1:10000002:1] HTTP Request Detected [**] [Priority: 0] {TCP} 192.168.207.162:43188 -> 192.1
68.207.151:80
```

1. Event Timestamp:

- ❖ Date and Time: 05/13-21:25:32.492817, representing the date and time of the event in day/month-hour:minute:second.microseconds format.

2. Alert Information:

- ❖ SID and REV: SID:REV represents the Snort rule ID and revision number, respectively. In this case, SID is 1 and REV is 10000002.
- ❖ Event description: HTTP Request Detected, indicating the event.
- ❖ Priority: [Priority: 0], with 0 being the lowest priority.
- ❖ Protocol: {TCP}, indicating the event is related to TCP.

3. Source and Destination Details:

- ❖ Source IP: Source_IP, Source Port: Source_Port.
- ❖ Destination IP: Destination_IP, Destination Port: Destination_Port.

4. Repetition:

- ❖ The log shows a series of HTTP requests detected by Snort, all with a priority level of 0. The requests are coming from the same source IP address (192.168.207.162) and are destined for the same destination IP address (192.168.207.151) on port 80 (the default HTTP port).

HTTP

In Ubuntu:

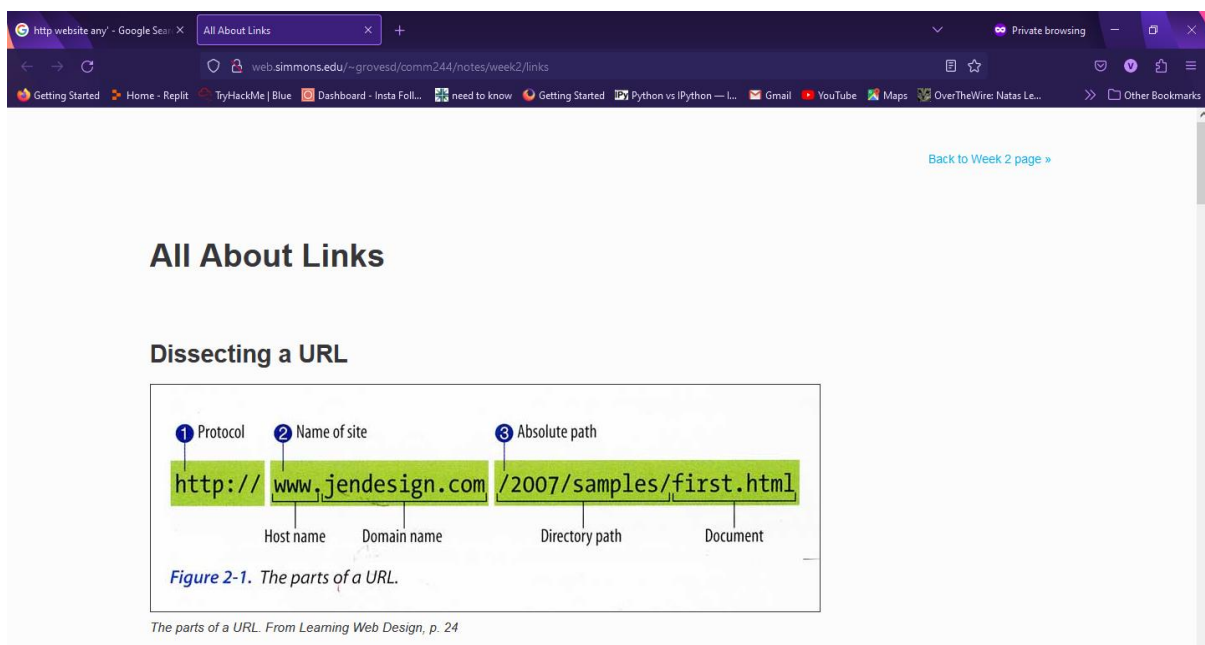
1. `alert tcp $HOME_NET any -> 80 any`

- ❖ This part of the rule triggers an alert (`alert`) for ICMP traffic (`ICMP`).
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`$HOME_NET`) and `any` is the destination port.

2. `(msg: "HTTP Connection Detected"; sid: 100001010101;)`

- ❖ `msg: "HTTP Connected Detected";` Specifies the message displayed when the alert is triggered, indicating a 80 port scan detection.
- ❖ `sid: 100001010101;` Assigns a unique ID (100001010101) to the rule, allowing for easy identification and management within Snort.

In Ubuntu:



When we try to open http website the snort will generate an alert or block.

Log by Snort:

```
720d:6574:9ae0:27205 -> 2600:140f:2c00::684d:ad69:80
05/11-22:31:24.535487  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27205 -> 2600:140f:2c00::684d:ad69:80
05/11-22:31:24.616480  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27205 -> 2600:140f:2c00::684d:ad69:80
05/11-22:32:07.048859  [**] [1:254:4] DNS SPOOF query response with TTL of 1 min. and no authority [**] [Classification:
Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.237.62:53 -> 192.168.237.76:38954
05/11-22:32:10.679142  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network S
can] [Priority: 3] {UDP} 192.168.237.74:59671 -> 239.255.255.250:1900
05/11-22:32:13.679993  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network S
can] [Priority: 3] {UDP} 192.168.237.74:59671 -> 239.255.255.250:1900
05/11-22:32:16.680849  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network S
can] [Priority: 3] {UDP} 192.168.237.74:59671 -> 239.255.255.250:1900
05/11-22:32:24.575766  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27205 -> 2600:140f:2c00::684d:ad69:80
05/11-22:32:24.575767  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 192.168.237.74:27204 -> 1
52.195.38.76:80
05/11-22:32:24.575771  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 192.168.237.74:27203 -> 1
52.195.38.76:80
05/11-22:32:24.575771  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27202 -> 2600:140f:2400:1a1::1b01:80
05/11-22:32:24.607030  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 192.168.237.74:27203 -> 1
52.195.38.76:80
05/11-22:32:24.619232  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 192.168.237.74:27204 -> 1
52.195.38.76:80
05/11-22:32:24.619234  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27205 -> 2600:140f:2c00::684d:ad69:80
05/11-22:32:24.619234  [**] [1:1216762293:0] HTTP Connection Detected [**] [Priority: 0] {TCP} 2401:4900:3600:d8c1:cd8e:
720d:6574:9ae0:27202 -> 2600:140f:2400:1a1::1b01:80
Activate Windows
Go to Settings to activate Win
```

1. Event Timestamp:

- ❖ Each line in the log contains a timestamp in the format May 11th at 10:31:24.535487 PM, indicating the date and time of the event.

2. Alert Information:

- ❖ SID represents the Snort rule ID and revision number. In this case, the SID is 100001010101.
- ❖ Event description: HTTP Request Detected.
- ❖ Priority: Priority level of the event, with 0 being the lowest priority.

3. Repetition:

- ❖ The log shows a series of HTTP requests detected by Snort, all with a priority level of 0. The requests are coming from the same source IP address and are destined for the same destination IP address on port 80 (the default HTTP port).

Remote Code Execution

In Ubuntu:

1. `alert tcp any any -> any any`

- ❖ This part of the rule triggers an alert (`alert`)
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`any`) and `any` is the destination port.

2. `(msg: "Remote Code Execution Attempt"; content:"exec("; nocase; sid: 100000000256;)`

- ❖ `msg: " Remote Code Execution Attempt ";` Specifies the message displayed when the alert is triggered.
- ❖ `content: "exec(";` This specifies the content to search for in the packet payload. In this case, it's looking for the string "exec(" (note the opening parenthesis).
- ❖ `sid: 100000000256;` Assigns a unique ID (100000000256) to the rule, allowing for easy identification and management within Snort.

In Kali:

```
1 import socket
2
3 target_ip = '192.168.207.151'
4 target_port = # Choose any available port
5 payload = b'exec('
6
7 s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8 s.connect((target_ip, target_port))
9 s.sendall(payload)
10 s.close()
11
```

You can use `sudo` also to make it even more effective

```
(venukamatchi@kali)-[~/Desktop]
$ python3 script.py

(venukamatchi@kali)-[~/Desktop]
$
```

Log by Snort:

```
venukamatchi@venukamatchi:/etc/snort$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/14-02:21:53.565821  [**] [1:1215752448:0] Remote Code Execution Attempt [**] [Priority: 0] {TCP} 192.168.207.162:4829
4 -> 192.168.207.151:80
```

1. Event Timestamp:

- ❖ Timestamp of when the event occurred: May 14th, 2:21:53 AM, including the microsecond (565821) when the packet was captured.

2. Alert Information:

- ❖ Snort rule ID (SID) and revision number (0): SID is 1215752448, corresponding to a rule detecting a "Remote Code Execution Attempt."
- ❖ Description of the alert: "Remote Code Execution Attempt."
- ❖ Priority level of the alert: 0, indicating the lowest priority.
- ❖ Transport layer protocol used: TCP (Transmission Control Protocol).

3. Source and Destination Details:

- ❖ Source IP: 192.168.207.162, Source Port: 4829.
- ❖ Destination IP: 192.168.207.151, Destination Port: 80.
- ❖ Arrow -> indicates the direction of communication from source to destination.

4. Repetition:

- ❖ This log entry describes a single event and does not indicate repetition.

C&C

In Ubuntu:

1. `alert tcp any any -> any any`

- ❖ This part of the rule triggers an alert (`alert`)
- ❖ It applies to any source IP address (`any`) and any source port (`any`).
- ❖ It's directed towards any destination IP address (`any`) and `any` is the destination port.

2. `(msg: "Remote Code Execution Attempt"; content:"exec("; nocase; sid: 1000000000000006;)`

- ❖ `msg: " Command Injection Attempt ";` Specifies the message displayed when the alert is triggered.
- ❖ `content: "?"`: This is the actual rule that looks for a specific pattern in the network traffic. In this case, the pattern is a single question mark character (?). The `B` keyword indicates that this is a byte-level match, meaning that Suricata will look for an exact match of the specified pattern.
- ❖ `sid: 1000000000000006;` Assigns a unique ID (1000000000000006) to the rule, allowing for easy identification and management within Snort.

In Kali:

```
1 from scapy.all import *
2
3 # Craft TCP packet with payload containing ";"
4 packet = IP(dst="192.168.1.1") / TCP(dport=24) / Raw(load=";")
5
6 # Send the packet
7 send(packet)
8
```

```
(venukamatchi@kali)-[~/Desktop]
$ sudo python3 script_1.py
[sudo] password for venukamatchi:
.
Sent 1 packets.
```

Log by Snort:

```
venukamatchi@venukamatchi:~$ sudo snort -q -i enp0s3 -l /var/log/snort -A console -c /etc/snort/snort.conf
05/14-03:25:21.189083  ** [1:276447238:0] Command Injection Attempt ** [Priority: 0] {TCP} 157.240.16.52:443 -> 192.168.145.151:37390
05/14-03:25:21.416639  ** [1:276447238:0] Command Injection Attempt ** [Priority: 0] {TCP} 2a03:2880:f22f:c5:face:b00c:0:167:443 -> 2402:3a80:1bbe:4611:891f:8936:3a49:9e4b:45414
```

1. Event Timestamp:

- ❖ Timestamp indicating when the event occurred: 14th of May at 03:25:21.189883 AM.

2. Alert Information:

- ❖ Signature ID (SID): 1:276447238:0, triggering the alert for "Command Injection Attempt."
- ❖ Description of the alert: "Command Injection Attempt."
- ❖ Priority level of the alert: 0, denoting the lowest priority.

3. Source and Destination Details:

- ❖ Source IP: 157.240.16.52, Source Port: 443.
- ❖ Destination IP: 192.168.145.151, Destination Port: 37390.
- ❖ Protocol used: TCP (Transmission Control Protocol).

4. Repetition:

- ❖ The log includes multiple entries, each indicating a separate event, but no repetition within the given context.

In conclusion, this project has been a testament to unwavering dedication and tireless commitment. Countless hours of effort and numerous sleepless nights have been poured into this endeavour. The meticulous configuration, comprehensive understanding, and relentless pursuit of cyber security excellence have culminated in a robust defence mechanism against potential threats. As we close this chapter, it's evident that safeguarding our networks requires continuous vigilance and adaptation to the ever-changing landscape of cyber threats. With the implementation of Snort and the steadfast determination to fortify our organization's security, we stride confidently into the future, knowing that we've taken significant steps towards ensuring a safer digital environment for all. As of the current time, 00:50.