

## **Snort IDS Project Report**

**Title:** Installing and Configuring Snort on Ubuntu Server (VirtualBox Lab)

**Author:** Rishav K Thapa

**Date:** 7/6/2025

## **Introduction**

This project demonstrates how to install and configure Snort, an open-source Intrusion Detection System (IDS), on an Ubuntu Server using VirtualBox. Snort monitors traffic for suspicious activity and logs alerts based on predefined rules. This is part of a blue team security lab where real attacks are simulated using Nmap to test detection capabilities.

## **Objective**

- Learn how to install and configure Snort on a Linux server.
- Simulate real attacks using tools like Nmap.
- Detect and analyze traffic with Snort alerts.
- Build a reusable IDS lab environment for future learning and experimentation.

## **Tools Used**

- Ubuntu Server 22.04 (VirtualBox)
- Snort 2.9.15.1
- Nmap (for scanning and attack simulation)
- nano (for editing config files)
- VirtualBox (for creating virtual lab)

## Lab Deployment Overview

In this demonstration, we set up Snort IDS on an Ubuntu Server running inside VirtualBox. The server was configured with a static IP, and Snort was installed, tuned with the correct HOME\_NET, and then tested using basic Nmap scanning techniques from a Kali Linux machine. The goal was to verify if Snort could detect and log malicious or suspicious network traffic in a controlled lab environment.

### Step 1: Install Snort

```
user@user:~$ sudo apt install snort_
```

- sudo apt update
- sudo apt install snort -y

### Step 2: Check Snort Version

```
user@user:~$ snort --version

  __  __  _
 o"/  )~
  '    '

-*> Snort! <*-
Version 2.9.15.1 GRE (Build 15125)
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team
Copyright (C) 2014-2019 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.10.1 (with TPACKET_V3)
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

user@user:~$
```

- snort --version

### Step 3: Verify Snort Files

```
user@user:~$ cd /etc/snort/
user@user:/etc/snort$ ls
attribute_table.dtd  file_magic.conf  rules              threshold.conf
classification.config  gen-msg.map     snort.conf         unicode.map
community-sid-msg.map  reference.config snort.debian.conf
user@user:/etc/snort$ nano snort.conf
```

- cd /etc/snort
- ls

Important files and directories:

- ✓ snort.conf
- ✓ classification.config
- ✓ reference.config
- ✓ threshold.conf
- ✓ rules/
- ✓ unicode.map
- ✓ snort.debian.conf

### Step 4: Configure HOME\_NET in snort.conf

- sudo nano /etc/snort/snort.conf
- ipvar HOME\_NET 192.168.1.111

```
GNU nano 6.2                                snort.conf *
# 6) Configure output plugins
# 7) Customize your rule set
# 8) Customize preprocessor and decoder rule set
# 9) Customize shared object rule set
#####

#####
# Step #0: (Debian specific) Create a configuration
#           for a specific interface
#####
#
# If you want to run Snort in Debian using different
# instances each handling a different interface and
# a different configuration you can copy this file to
# /etc/snort/snort.$interface.conf (where '$interface' is the name of your
# network interface) and adjust the value there.
#
# The Debian init.d script is defined in such a way
# that you can run multiple instances.

#####
# 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 192.168.1.111_

^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^N Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-F Redo
```

## Step 5: Start Snort and Monitor Traffic

### On Ubuntu server:

- `cd /var/log/snort`
- `ls`
- `tail -f snort.alert.fastnmap -sS 192.168.1.111`

### On kali machine:

`nmap -sS 192.168.1.111`

```
kali@kali:~$ cd /etc/snort
kali@kali:~/etc/snort$ cat 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 overridden when starting
# up the Snort daemon through the init.d script by the
# value of $DEBIAN_SNORT_HOME_NET defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.1.111

user@user:/etc/snort$ cd /var/log/snort
user@user:/var/log/snort$ ls
snort.alert  snort.alert.fast  snort.log
user@user:/var/log/snort$ tail -f snort.alert.fast
07/06-11:24:56.675396 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] [UDP] 0.0.0.0:0:0 -> 255.255.255.255:67
07/06-11:24:56.689773 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] [IPV6-ICMP] :: -> ff02::16
07/06-11:24:56.817771 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] [IPV6-ICMP] :: -> ff02::16
07/06-11:24:56.950329 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] [UDP] 0.0.0.0:0:0 -> 255.255.255.255:67
07/06-11:24:56.822434 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] [UDP] 0.0.0.0:0:0 -> 255.255.255.255:67
07/06-11:25:17.957886 [**] [1:1461:11] SNMP request tcp [**] [Classification: Attempted Information Leak] [Priority: 2] [TCP] 192.168.1.67:45315 -> 192.168.1.111:161
07/06-11:25:18.102151 [**] [1:1461:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] [TCP] 192.168.1.67:45315 -> 192.168.1.111:705
```

Snort successfully captured and logged various types of network activity, including suspicious traffic triggered by simulated Nmap scans. The alerts were generated in real time and stored in the snort.alert.fast log file, confirming that Snort was actively monitoring the interface and applying its detection rules. This validates that our configuration is working correctly, and the IDS is functioning as expected within the virtual lab environment.



## **Results & Observations**

- Snort successfully detected simulated attacks from Nmap.
- Alerts for "BAD-TRAFFIC", SNMP scanning, and UDP broadcasts were captured.
- Confirmed Snort is properly monitoring the interface and logging alerts based on traffic patterns.

## **Problems Faced**

- Snort failed to start initially due to an incorrect interface name.
- Need to manually fix the HOME\_NET variable.
- Required enabling the correct NIC (eth0) in VirtualBox settings.
- Rules needed to be adjusted for fine-tuned detection.

## **Future Enhancements**

- Add custom Snort rules for known threats.
- Integrate Snort with ELK Stack or Wazuh for log visualization.
- Set up email alerts or syslog forwarding.
- Automate rule updates and configuration backups.

## **Conclusion**

This project helped me understand how Snort detects network intrusions and how to set up a working IDS environment inside a virtual lab. The process also involved troubleshooting configurations and simulating real-world attacks. This forms a strong foundation for further work in blue teaming and network security analysis.

## References

- <https://www.snort.org/>
- <https://ubuntu.com>
- <https://nmap.org>
- <https://www.kali.org>