INFOTACT SOLUTIONS

Analetizia Simono

Internship (Infotact Solutions)

NIDS Rule Creation & Testing Lab

28th September, 2025

Network Intrusion & Detection System (NIDS) Rule Creation & Testing Lab

Executive Summary

This project implements and validates a custom Snort detection rule to identify SSH brute-force attempts inside an isolated virtual lab. I built a reproducible environment (Ubuntu Server victim VM and Kali as attacker host and controller) and deployed Snort with a local rule that flags any IP that attempts five SSH connections to the server within 60 seconds. The lab demonstrates that Snort generates timely, actionable alerts during a Hydra-driven brute-force test (alert triggered after the configured threshold), proving the rule reliably reduces the mean-time-to-detect for this class of attack in a controlled network.

Introduction

Modern defenders need fast, reliable detection of common attack patterns so analysts can triage and respond before attackers succeed. The goal of Project 1 was to create and test a set of targeted NIDS rules that detect noisy, automated activity (brute-force logins) in real time, thereby reducing detection time and giving security teams a clear, machine-readable alert to investigate.
This is important because brute-force attacks are noisy and common; detecting them quickly prevents account compromise, stops lateral movement, and produces clear forensic artifacts (alerts, logs, etc) for investigation. A simple, well-tuned rule like this is low overhead and integrates easily into analyst workflows or a SIEM pipeline (TryHackMe).

Methodology

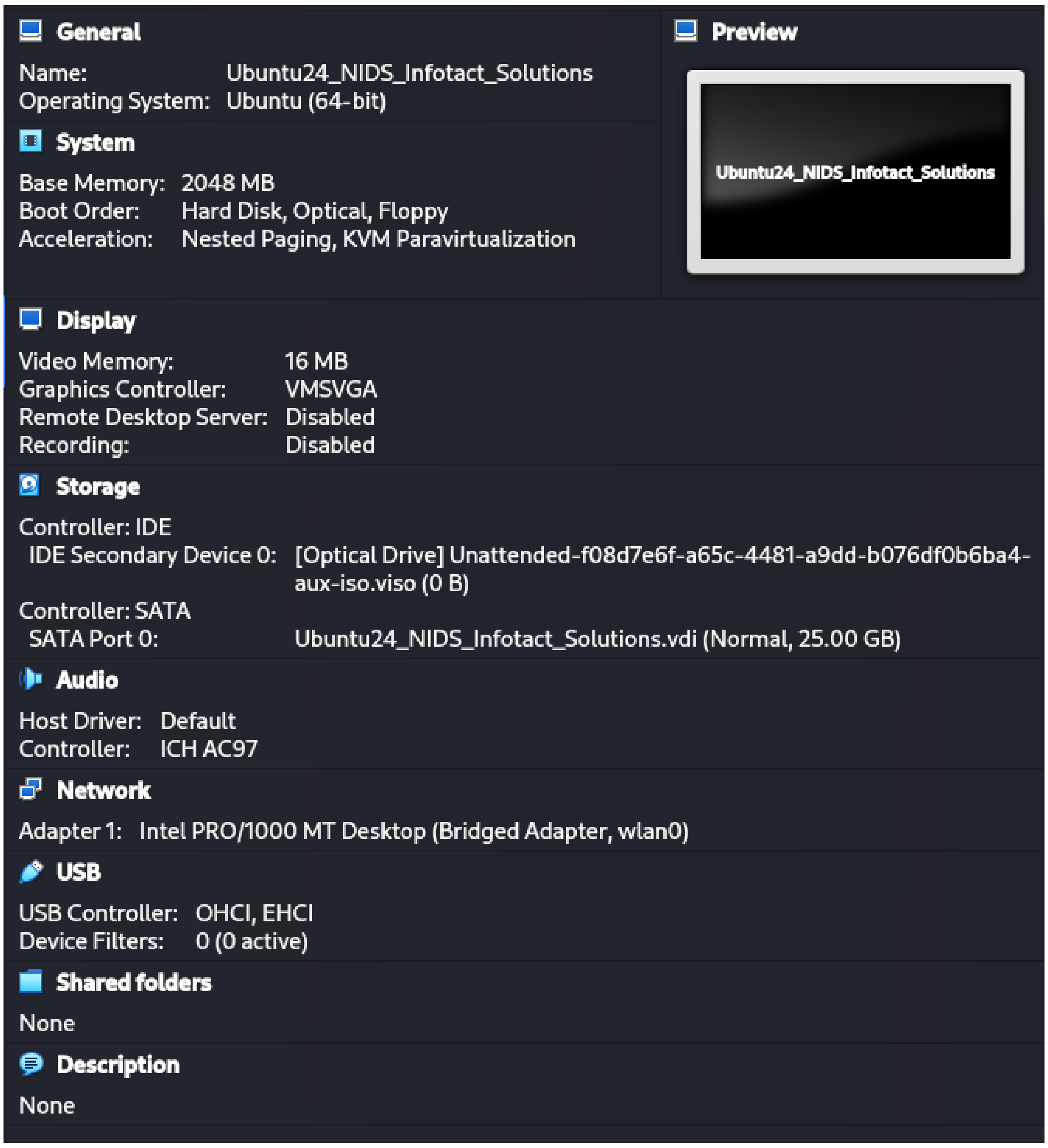
1. Prerequisites

- Enough system resources to handle both:
  - Ubuntu VM: 1 CPU, 2GB RAM, 25GB disk recommended.
  - Kali host: 2 CPUs, 2GB RAM, 40GB disk recommended.
- Stable network configuration (Ubuntu reachable with ping).
- Update all packages on both Linux devices before proceeding with the operations

2. Devices Setup

- **Host OS:** Kali Linux 2025.2 (used as the attacker, runs Hydra for brute-force simulation and is lab controller).
- **Hypervisor:** VirtualBox
- **Victim VM:** Ubuntu Server 24.04.3 live server, named Ubuntu24_NIDS_Infotact_Solutions. Runs sshd service and Snort IDS for monitoring incoming traffic.
- **Network Mode Setup:** Bridged Adapter (so VMs get IPs on the same LAN as host).

Figure 1: VM settings



General

Name: Ubuntu24_NIDS_Infotact_Solutions
Operating System: Ubuntu (64-bit)

System

Base Memory: 2048 MB
Boot Order: Hard Disk, Optical, Floppy
Acceleration: Nested Paging, KVM Paravirtualization

Preview

Display

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

Storage

Controller: IDE
IDE Secondary Device 0: [Optical Drive] Unattended-f08d7e6f-a65c-4481-a9dd-b076df0b6ba4-aux-iso.viso (0 B)
Controller: SATA
SATA Port 0: Ubuntu24_NIDS_Infotact_Solutions.vdi (Normal, 25.00 GB)

Audio

Host Driver: Default
Controller: ICH AC97

Network

Adapter 1: Intel PRO/1000 MT Desktop (Bridged Adapter, wlan0)

USB

USB Controller: OHCI, EHCI
Device Filters: 0 (0 active)

Shared folders

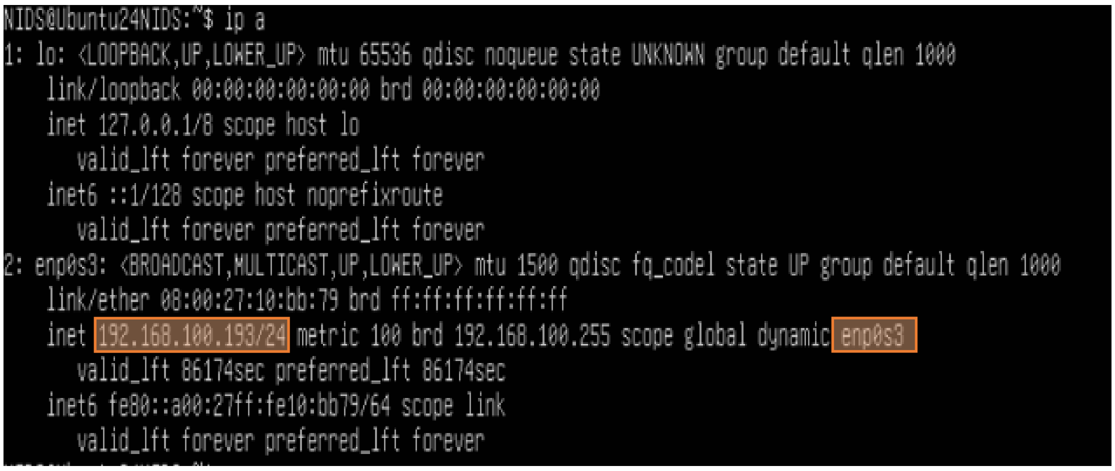
None

Description

None

Setting up Snort on Victim VM:

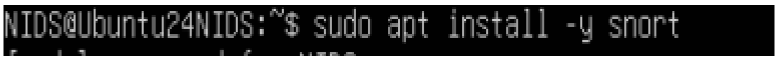
Figure 2: Checked the IPv4 address and interface for use on Snort setup



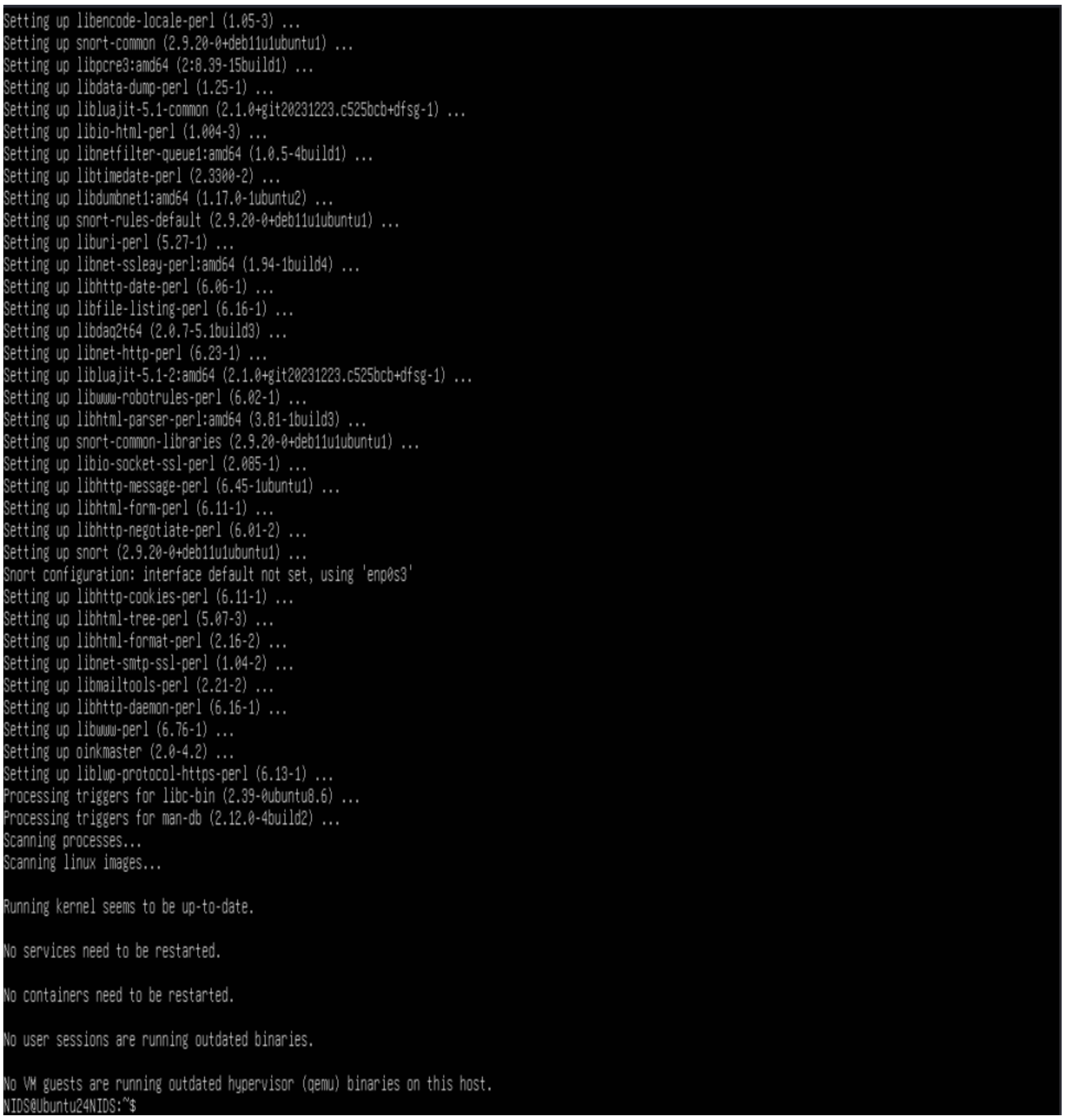
```
NIDS@Ubuntu24NIDS:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:10:bb:79 brd ff:ff:ff:ff:ff:ff
    inet 192.168.100.193/24 metric 100 brd 192.168.100.255 scope global dynamic enp0s3
       valid_lft 86174sec preferred_lft 86174sec
    inet6 fe80::a00:27ff:fe10:bb79/64 scope link
       valid_lft forever preferred_lft forever
```

Snort IDS installation: v2.9.20-0 (from Ubuntu package repositories).

Figure 3: Snort installation

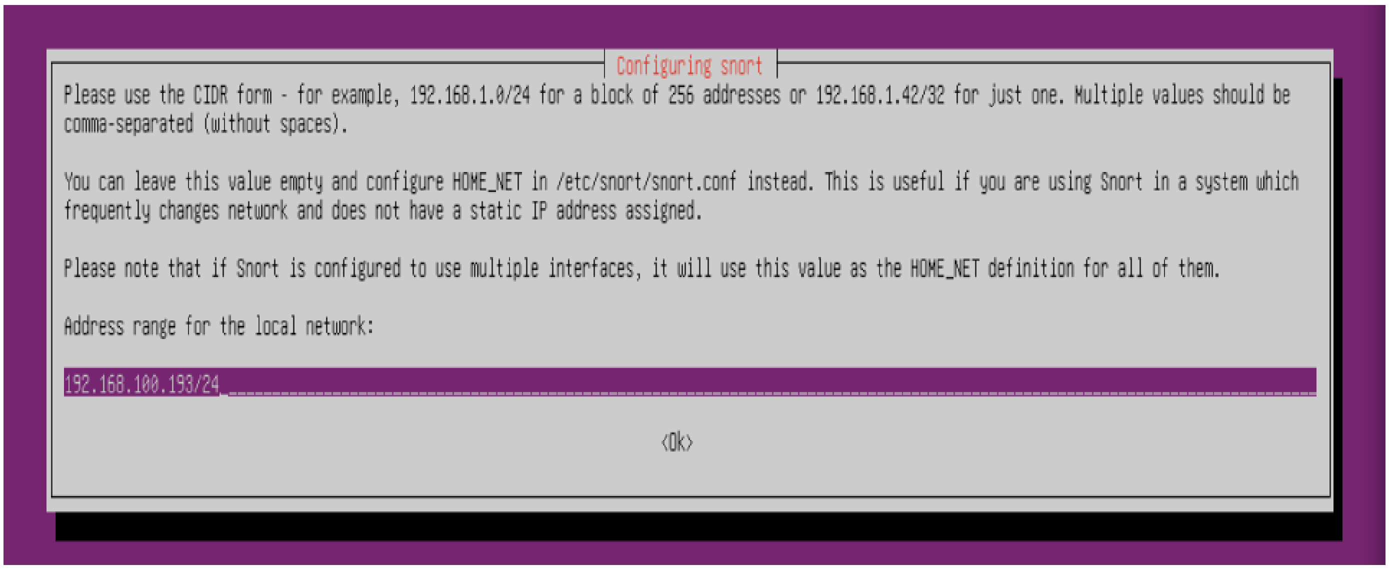


```
NIDS@Ubuntu24NIDS:~$ sudo apt install -y snort
```



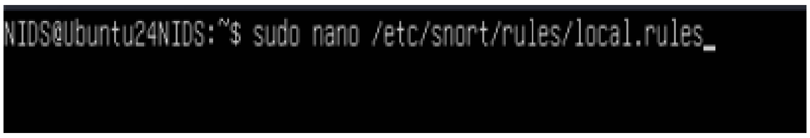
```
Setting up libencode-locale-perl (1.05-3) ...
Setting up snort-common (2.9.20-0+deb11u1ubuntu1) ...
Setting up libpcre3:amd64 (2:8.39-15build1) ...
Setting up libdata-dump-perl (1.25-1) ...
Setting up libluajit-5.1-common (2.1.0+git20231223.c525bcb+dfsg-1) ...
Setting up libio-html-perl (1.004-3) ...
Setting up libnetfilter-queue1:amd64 (1.0.5-4build1) ...
Setting up libtimedate-perl (2.3300-2) ...
Setting up libdumbnet1:amd64 (1.17.0-1ubuntu2) ...
Setting up snort-rules-default (2.9.20-0+deb11u1ubuntu1) ...
Setting up liburi-perl (5.27-1) ...
Setting up libnet-ssleay-perl:amd64 (1.94-1build4) ...
Setting up libhttp-date-perl (6.06-1) ...
Setting up libfile-listing-perl (6.16-1) ...
Setting up libdaq2t64 (2.0.7-5.1build3) ...
Setting up libnet-http-perl (6.23-1) ...
Setting up libluajit-5.1-2:amd64 (2.1.0+git20231223.c525bcb+dfsg-1) ...
Setting up libwww-robotrules-perl (6.02-1) ...
Setting up libhtml-parser-perl:amd64 (3.81-1build3) ...
Setting up snort-common-libraries (2.9.20-0+deb11u1ubuntu1) ...
Setting up libio-socket-ssl-perl (2.085-1) ...
Setting up libhttp-message-perl (6.45-1ubuntu1) ...
Setting up libhtml-form-perl (6.11-1) ...
Setting up libhttp-negotiate-perl (6.01-2) ...
Setting up snort (2.9.20-0+deb11u1ubuntu1) ...
Snort configuration: interface default not set, using 'enp0s3'
Setting up libhttp-cookies-perl (6.11-1) ...
Setting up libhtml-tree-perl (5.07-3) ...
Setting up libhtml-format-perl (2.16-2) ...
Setting up libnet-smtp-ssl-perl (1.04-2) ...
Setting up libmailtools-perl (2.21-2) ...
Setting up libhttp-daemon-perl (6.16-1) ...
Setting up libwww-perl (6.76-1) ...
Setting up oinkmaster (2.0-4.2) ...
Setting up liblwp-protocol-https-perl (6.13-1) ...
Processing triggers for libc-bin (2.39-0ubuntu8.6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
NIDS@Ubuntu24NIDS:~$
```

Figure 4: IP address on Snort



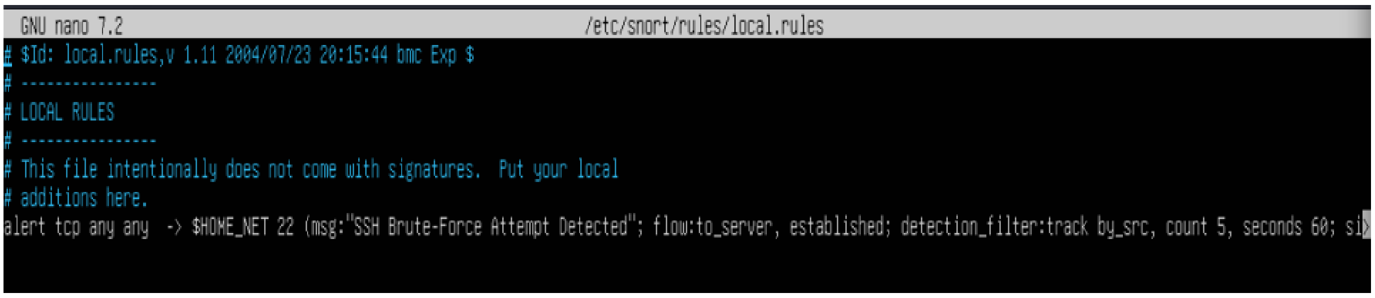
```
Configuring snort

Please use the CIDR form - for example, 192.168.1.0/24 for a block of 256 addresses or 192.168.1.42/32 for just one. Multiple values should be
comma-separated (without spaces).

You can leave this value empty and configure HOME_NET in /etc/snort/snort.conf instead. This is useful if you are using Snort in a system which
frequently changes network and does not have a static IP address assigned.

Please note that if Snort is configured to use multiple interfaces, it will use this value as the HOME_NET definition for all of them.

Address range for the local network:

192.168.100.193/24_

<Ok>
```

Figure 5: Opening the local.rules file with nano

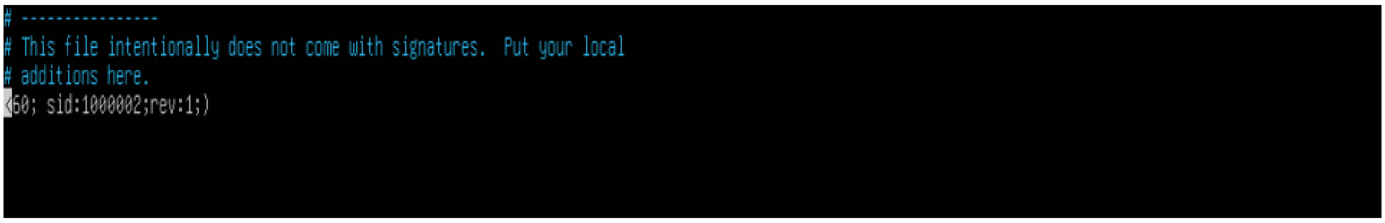


```
NIDS@Ubuntu24NIDS:~$ sudo nano /etc/snort/rules/local.rules_
```

Figure 6: Adding the rule which alert when a single source IP makes 5 TCP connections to a host's SSH port (22) within 60 seconds; rule ID 1000002 to avoid overwriting previous rules.

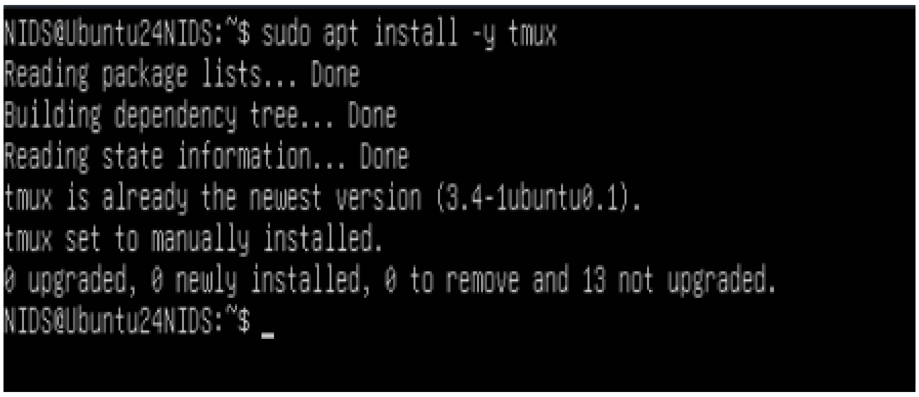


```
  GNU nano 7.2                                            /etc/snort/rules/local.rules
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
alert tcp any any  -> $HOME_NET 22 (msg:"SSH Brute-Force Attempt Detected"; flow:to_server, established; detection_filter:track by_src, count 5, seconds 60; si>
```

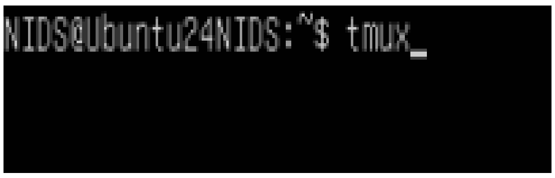


```
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
<60; sid:1000002;rev:1;)
```

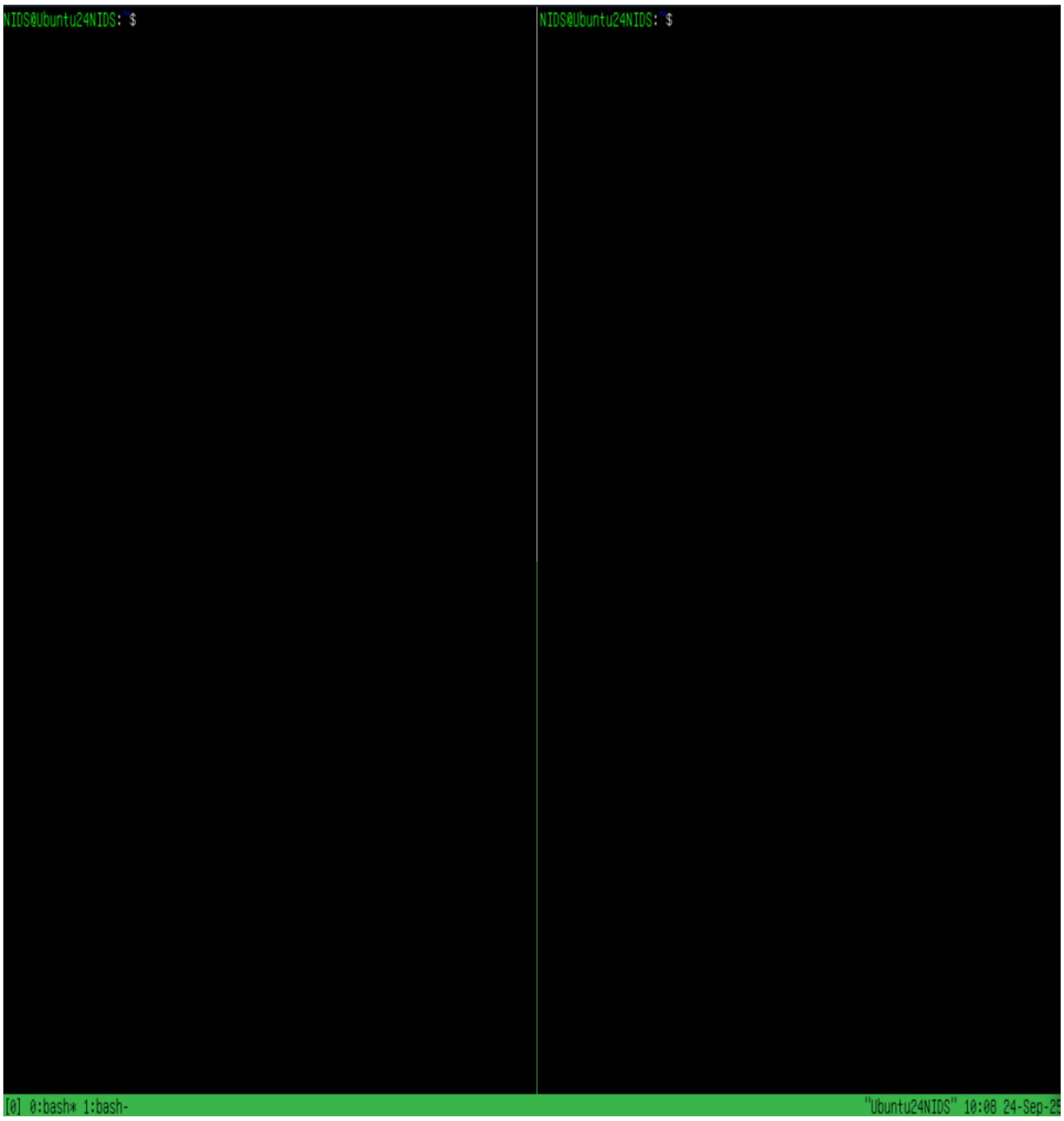
Figure 7: tmux setup for multi-window management in case of errors during execution



```
NIDS@Ubuntu24NIDS:~$ sudo apt install -y tmux
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
tmux is already the newest version (3.4-1ubuntu0.1).
tmux set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 13 not upgraded.
NIDS@Ubuntu24NIDS:~$ _
```

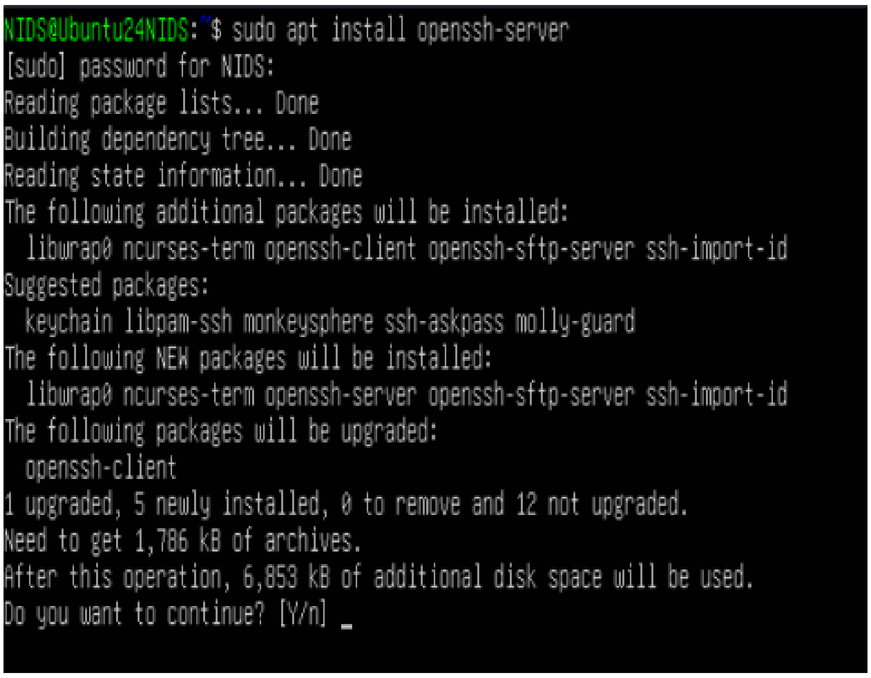


```
NIDS@Ubuntu24NIDS:~$ tmux_
```



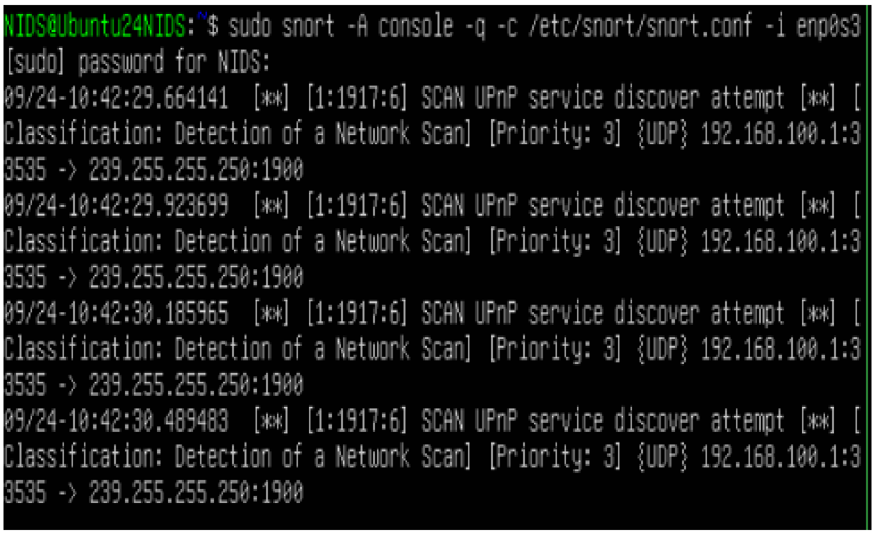
```
NIDS@Ubuntu24NIDS:~$                    │NIDS@Ubuntu24NIDS:~$
[0] 0:bash* 1:bash-                     "Ubuntu24NIDS" 10:08 24-Sep-25
```

Figure 8: Installing the openssh-server to attack



```
NIDS@Ubuntu24NIDS:~$ sudo apt install openssh-server
[sudo] password for NIDS:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libwrap0 ncurses-term openssh-client openssh-sftp-server ssh-import-id
Suggested packages:
  keychain libpam-ssh monkeysphere ssh-askpass molly-guard
The following NEW packages will be installed:
  libwrap0 ncurses-term openssh-server openssh-sftp-server ssh-import-id
The following packages will be upgraded:
  openssh-client
1 upgraded, 5 newly installed, 0 to remove and 12 not upgraded.
Need to get 1,786 kB of archives.
After this operation, 6,853 kB of additional disk space will be used.
Do you want to continue? [Y/n] _
```

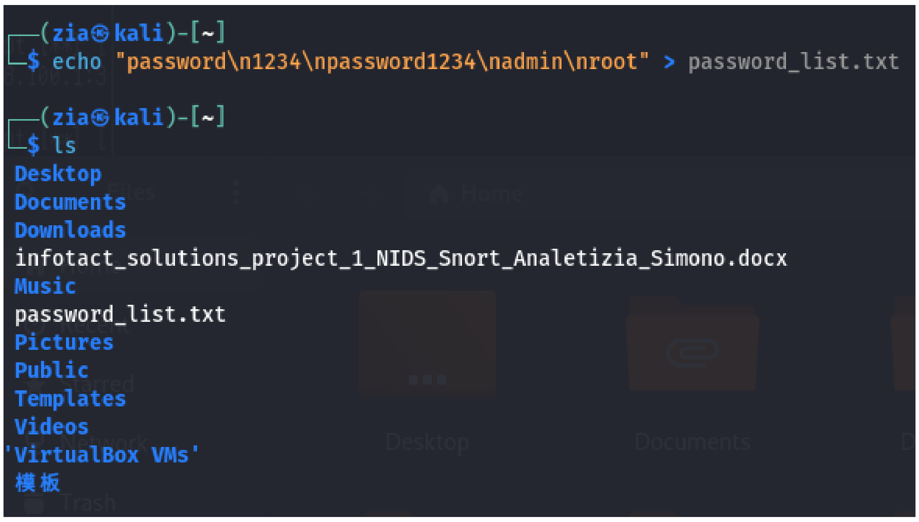
Figure 9: Running the snort command to start monitoring traffic



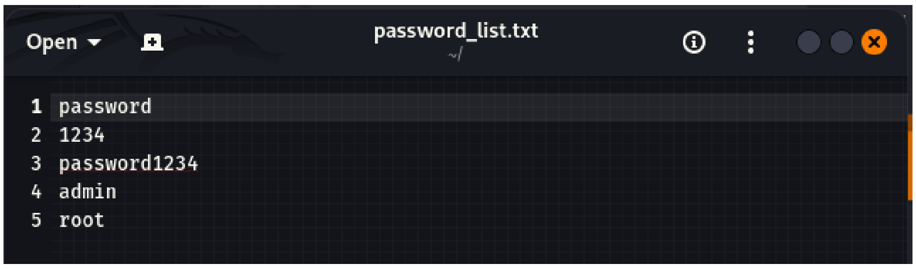
```
NIDS@Ubuntu24NIDS:~$ sudo snort -A console -q -c /etc/snort/snort.conf -i enp0s3
[sudo] password for NIDS:
09/24-10:42:29.664141  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:42:29.923699  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:42:30.185965  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:42:30.489483  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
```

On Kali host:

Figure 10: Making password file using 5 different, commonly used passwords

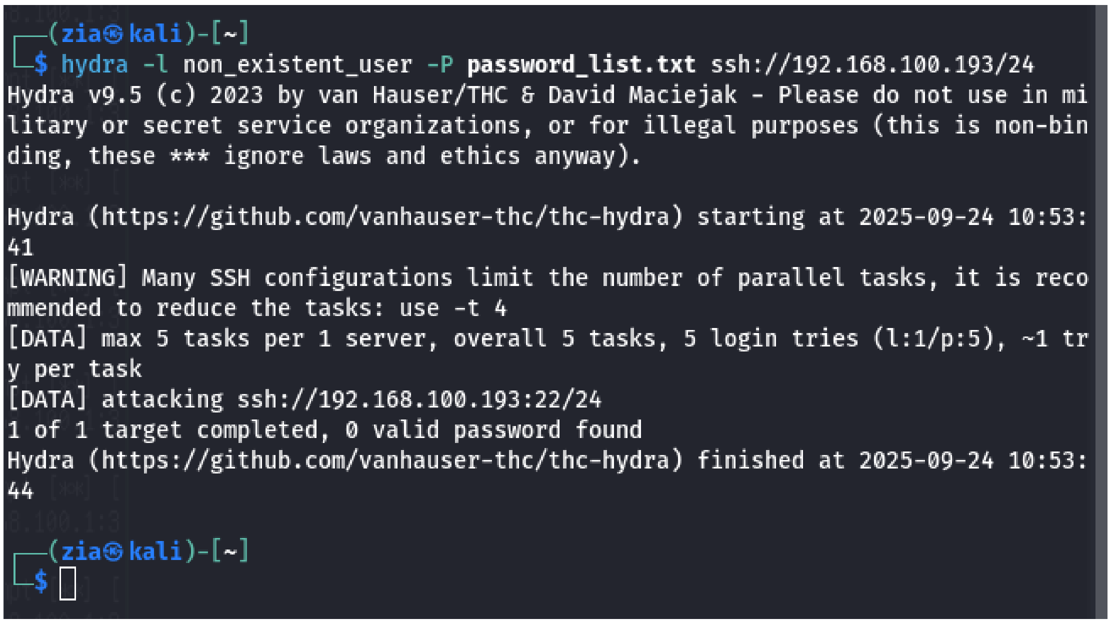


```
┌──(zia㉿kali)-[~]
└─$ echo "password\n1234\npassword1234\nadmin\nroot" > password_list.txt

┌──(zia㉿kali)-[~]
└─$ ls
Desktop
Documents
Downloads
infotact_solutions_project_1_NIDS_Snort_Analetizia_Simono.docx
Music
password_list.txt
Pictures
Public
Templates
Videos
'VirtualBox VMs'
模板
```



```
password
1234
password1234
admin
root
```

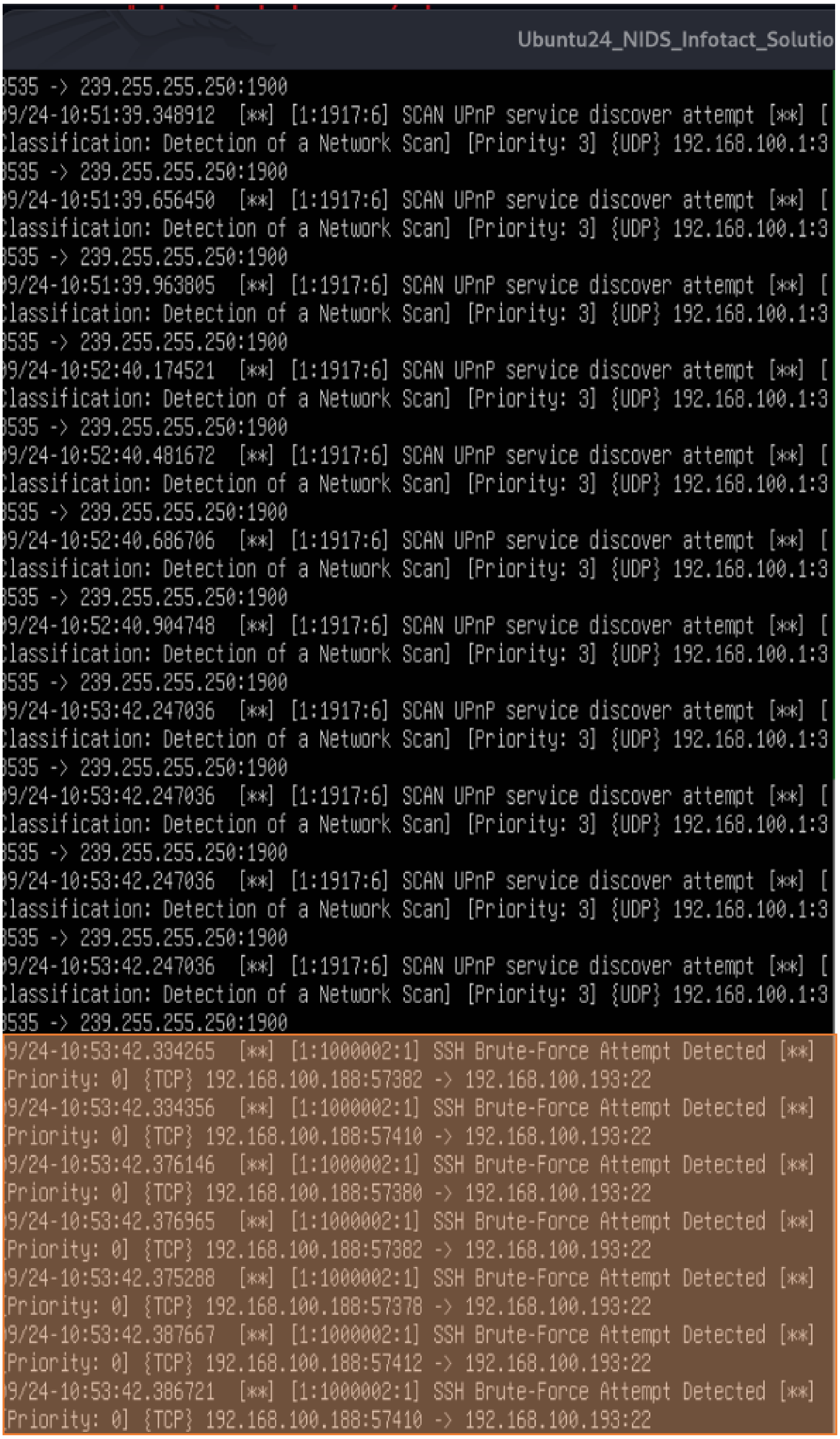
Figure 11: Running Hydra to attack our Ubuntu Server



```
┌──(zia㉿kali)-[~]
└─$ hydra -l non_existent_user -P password_list.txt ssh://192.168.100.193/24
Hydra v9.5 (c) 2023 by van Hauser/THC & David Maciejak - Please do not use in mi
litary or secret service organizations, or for illegal purposes (this is non-bin
ding, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2025-09-24 10:53:
41
[WARNING] Many SSH configurations limit the number of parallel tasks, it is reco
mmended to reduce the tasks: use -t 4
[DATA] max 5 tasks per 1 server, overall 5 tasks, 5 login tries (l:1/p:5), ~1 tr
y per task
[DATA] attacking ssh://192.168.100.193:22/24
1 of 1 target completed, 0 valid password found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2025-09-24 10:53:
44

┌──(zia㉿kali)-[~]
└─$ 
```

Findings

Figure 12: The custom rule appearing among the logs on the Ubuntu server



```
3535 -> 239.255.255.250:1900
09/24-10:51:39.348912  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:51:39.656450  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:51:39.963805  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:52:40.174521  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:52:40.481672  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:52:40.686706  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:52:40.904748  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:53:42.247036  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:53:42.247036  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:53:42.247036  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:53:42.247036  [**] [1:1917:6] SCAN UPnP service discover attempt [**] [
Classification: Detection of a Network Scan] [Priority: 3] {UDP} 192.168.100.1:3
3535 -> 239.255.255.250:1900
09/24-10:53:42.334265  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57382 -> 192.168.100.193:22
09/24-10:53:42.334356  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57410 -> 192.168.100.193:22
09/24-10:53:42.376146  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57380 -> 192.168.100.193:22
09/24-10:53:42.376965  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57382 -> 192.168.100.193:22
09/24-10:53:42.375288  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57378 -> 192.168.100.193:22
09/24-10:53:42.387667  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57412 -> 192.168.100.193:22
09/24-10:53:42.386721  [**] [1:1000002:1] SSH Brute-Force Attempt Detected [**]
[Priority: 0] {TCP} 192.168.100.188:57410 -> 192.168.100.193:22
```

The alerts generated provide a clear picture of the detected activity. Each alert begins with a timestamp in the format MM/DD-HH:MM:SS:MS, indicating precisely when the event occurred. Following this, we see the alert header, separated by [**].

- **Generator ID, SID, and Revision:** The first number is the generator ID, followed by the custom SID (Snort ID) from our local rule, and the revision number.
- **Message:** Next is the detection message we defined in the rule, confirming the type of activity Snort recognized.
- **Priority:** All alerts display a priority level of 0, which is the default since no manual priority was set in our rule configuration.
- **Protocol and IP Information:** The alerts identify the protocol used (TCP), the source IP (the attacker), and a source port number that changes with each attempt. The destination IP is consistently the Ubuntu server, and the destination port is always 22 (SSH).

The repetition of alerts with identical source and destination IPs, but varying source ports, confirms an automated brute-force attempt. This aligns with our test scenario, where Hydra was used to generate rapid login attempts against the SSH service. The attack lasted only 52 milliseconds, yet Snort successfully identified and logged multiple attempts.

Overall, the custom rule performed as intended: it flagged suspicious repeated SSH login attempts, demonstrating the effectiveness of using Snort for intrusion detection in real-world brute-force scenarios.

Discussion

Working through this lab was not a straight line — I made mistakes, learned from them, and picked up practical troubleshooting skills that go beyond just following instructions.

- **VM Setup Confusion**: At first, I mistakenly tried to make a bootable USB for Ubuntu Server, forgetting that virtual machines only require the ISO file. This reminded me to double-check requirements before diving in.
- **Installation Issues**: I initially struggled with downloading the minimal Ubuntu Server from Oracle's site. I plan to retry later, but it taught me the value of having backup sources and being flexible when tools don't cooperate.
- **Terminal Workflow**: On Ubuntu Server, I couldn't simply open new terminal windows like on Kali. This led me to discover and use tmux, which let me split one terminal into panes — running Snort on one side while installing and testing SSH on the other. This small but important workflow skill will definitely carry over into other projects.
- **Rule Writing & Syntax Errors**: Writing my first Snort rule was more challenging than expected. Tiny syntax issues, like misplaced semicolons (similar to Java errors I've faced before), broke my configuration. After trial-and-error and consulting online documentation (Cisco Talos Detection Response Team), I understood the proper formatting and successfully created a functioning rule.
- **Restarting Snort**: I also learned that changes to rules don't apply automatically — I had to restart Snort multiple times to load updates. This felt like rebooting a computer to fix an issue: sometimes, the simplest step is the solution.
- **Documentation Habits**: I realized the importance of taking screenshots before starting Snort in console mode, since scrolling can be difficult when Snort delivers alerts. In future labs, I'll be more mindful of capturing evidence at the right moment.

Finally, I want to emphasize that this entire exercise was conducted in a controlled lab environment with my own virtual machines. No public or unauthorized systems were involved.

This was purely an educational and professional project, simulating an attack and detection scenario safely.

Conclusion

This lab successfully implemented and validated a custom Snort rule to detect SSH brute-force attempts in an isolated, reproducible environment (Ubuntu Server victim VM; Kali attacker/controller). By tuning Snort to alert when a single source IP made five SSH connection attempts within 60 seconds and driving the test with Hydra, the NIDS produced timely, actionable alerts that clearly identified automated login attempts (repeated alerts with changing source ports and consistent destination port 22), demonstrating a measurable reduction in mean-time-to-detect for this attack class. The exercise also surfaced practical lessons—rule syntax must be exact, configuration changes require service restarts, and workflow tools like tmux improve visibility during testing—which improved my operational discipline and troubleshooting skills. Moving forward, I recommend refining rule thresholds to balance detection and false positives, integrating alerts into a SIEM for automated triage, and expanding test cases (different protocols and distributed attack sources) to validate the detection strategy across broader real-world scenarios.

References

Cisco Talos Detection Response Team. (n.d.). *detection_filter* — Snort 3 Rule Writing Guide. From https://docs.snort.org/rules/options/post/detection_filter

TryHackMe, et al. (n.d.). Pyramid of Pain. TryHackMe. Retrieved on 22 September, 2025 from https://tryhackme.com/room/pyramidofpainax