NİDS(Network Intrusion Detection System) REPORT

Prepared by: Vali Gasimli
date: 04.27.2025

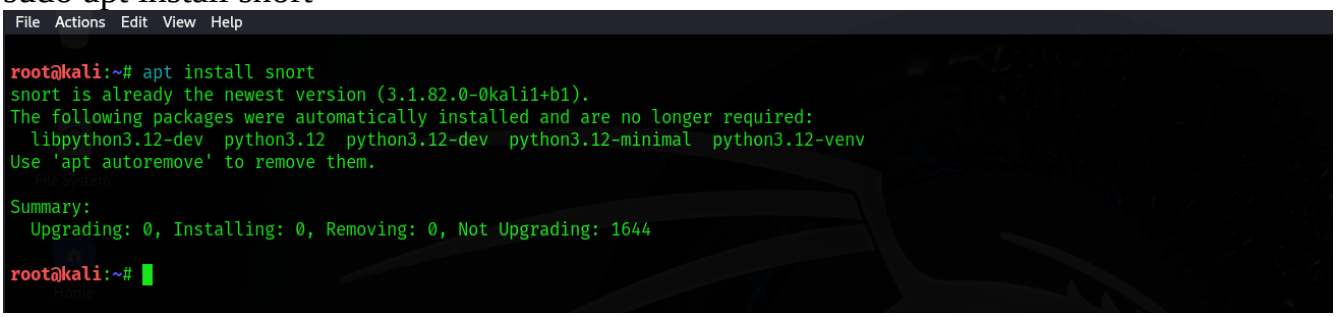
Content Headings:

1.introduction

This report provides an in-depth analysis of network traffic and security through the use of Snort++, a widely used intrusion detection and prevention system. The focus of this report is on the results of packet analysis, detection mechanisms, and the overall performance of Snort++ in a controlled testing environment. This analysis includes insights into different protocols and traffic patterns that were captured and analyzed during the testing period.

2.Operating Mechanism of Snort++ System

We will use snort to monitor malicious packets, etc. Download instructions:
sudo apt update
sudo apt install snort



Now we will write a **snort.lua** rule.

```
ips =
{
    variables = default_variables,
    rules = [[ /etc/snort/rules/local.rules ]]
}
```

Explanation:

- `ips`:
  This defines the configuration block for the Intrusion Prevention System (IPS) in Snort++. It tells Snort++ how to behave when inspecting network traffic for potential threats, and how to block malicious activities if needed.

- **`variables = default_variables`**:
  This line assigns a set of predefined variables called **`default_variables`**. These variables typically include important network information such as IP addresses, port numbers, and other network parameters (e.g., **`HOME_NET,`**

EXTERNAL_NET).
Using variables makes rule-writing easier, more flexible, and reusable.

- **rules = '/etc/snort/rules/ips.rules'**:
  This specifies the path to the file where the IPS rules are stored.
  Snort++ will load and apply the detection and prevention rules written in the **/etc/snort/rules/ips.rules** file to inspect the network traffic.

We are running snort. But one of the places we need to pay attention to is eth0 and wlan0.
sudo snort -c /etc/snort/snort.lua -i wlan0

We ping ourselves to monitor traffic and check sudo tail -f /var/log/snort/alert.

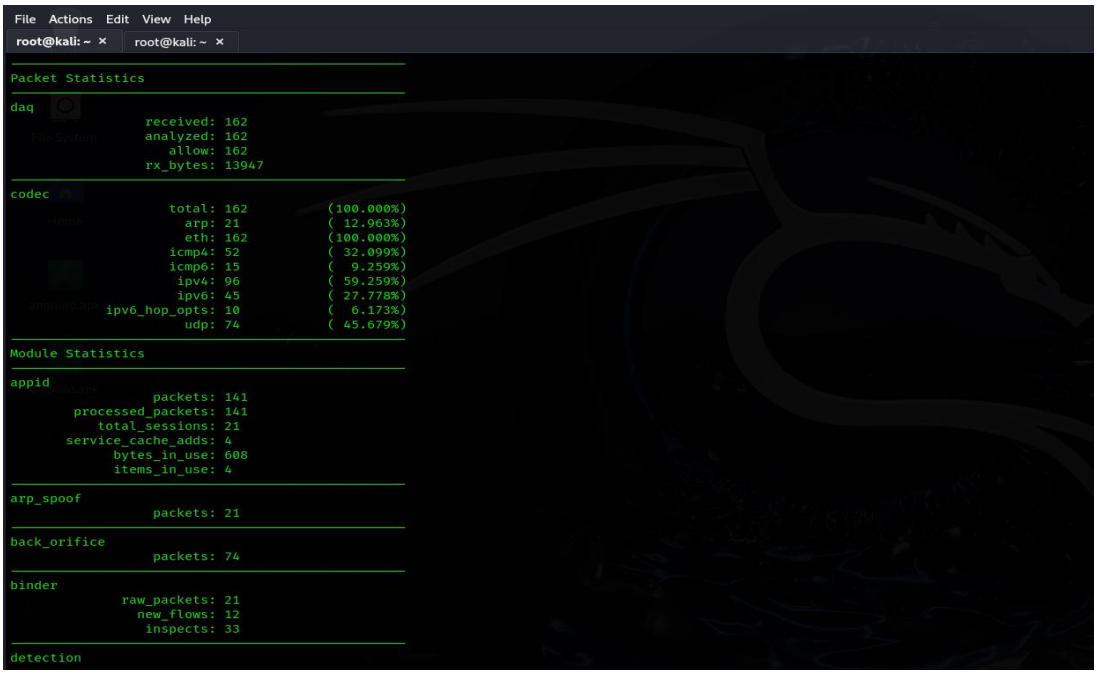
```
┌──(root㉿kali)-[~]
└─# tail -f /var/log/snort/alert

tail: cannot open '/var/log/snort/alert' for reading: No such file or directory
tail: no files remaining

┌──(root㉿kali)-[~]
└─# ping -f 192.168.100.99
PING 192.168.100.99 (192.168.100.99) 56(84) bytes of data.
.^
--- 192.168.100.99 ping statistics ---
136156 packets transmitted, 136156 received, 0% packet loss, time 1852ms
rtt min/avg/max/mdev = 0.002/0.003/0.696/0.006 ms, ipg/ewma 0.013/0.003 ms
```

3. Protocols and Modules Used

Snort++ supports various network protocols (ARP, ICMP, IPv4, IPv6, UDP, etc.) and analyzes traffic for these protocols. The system includes several modules and flow management mechanisms, such as:

- Stream Modules: Manages TCP, UDP, and ICMP flows.
- Port Scan: Detects port scanning activities on the network.
- ARP Spoofing: Prevents ARP spoofing attacks.
- Back Orifice: Monitors activities carried out by malware.

4. Analyzed Packets and Results



The following statistics provide an overview of the packets analyzed by Snort++:

Total Packets Received: 162

Packets Analyzed by Type:

- **ARP:** 21 packets
- **IPv4:** 96 packets
- **ICMP4:** 52 packets
- **UDP:** 74 packets

Network Flows: 12 new flows were created, and 9 flows experienced timeouts.

Detected Threats: ARP spoofing, Back Orifice, port scan.

5.Network Flow Management

The system has defined processes for managing network flows and detecting any anomalies in network traffic. The following statistics were recorded during the month:

- **Port Scan Analysis:** 141 packets were analyzed for port scanning activities.
- **ICMP Sessions:** 6 sessions were managed.
- **UDP Sessions:** 15 UDP sessions were created, and 9 UDP flows timed out.

6. Detected Applications and Services

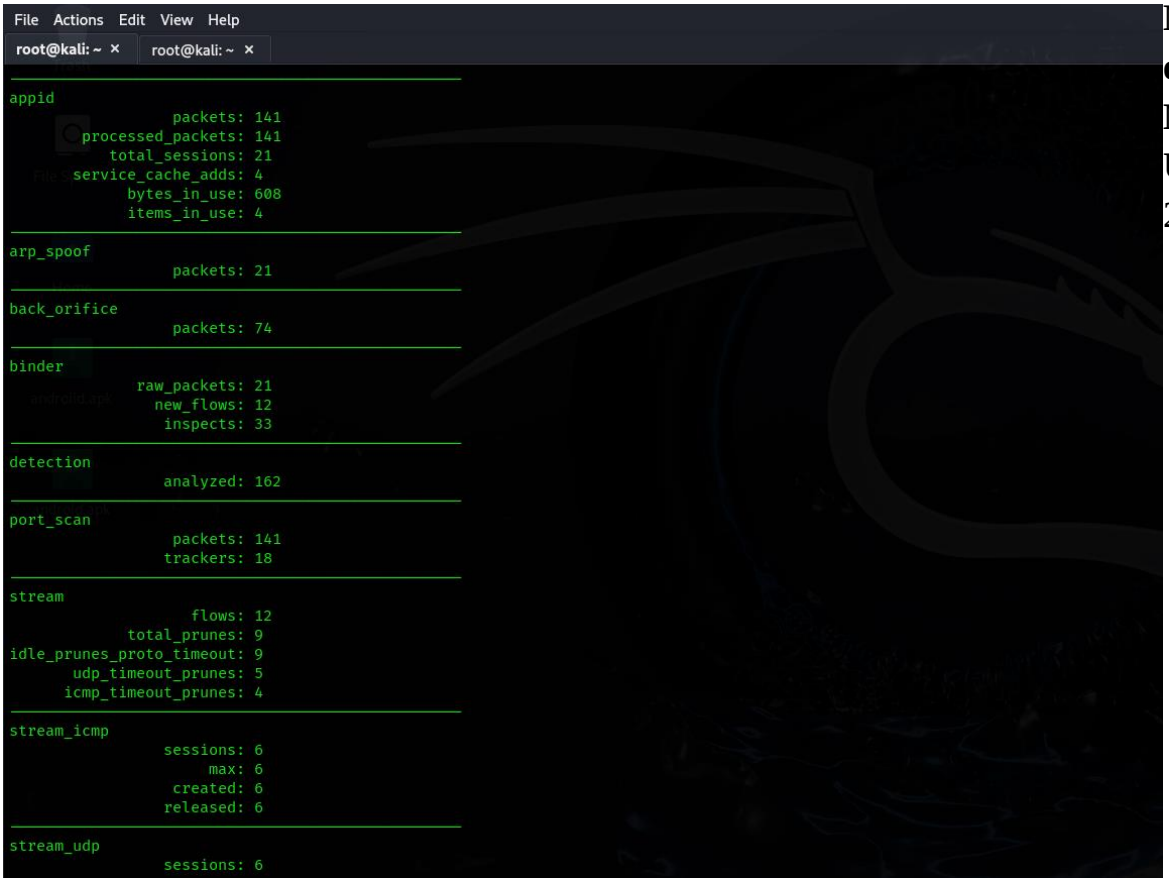
Snort++ analyzes the applications and services identified in network traffic and applies security measures accordingly. In this report, 13 applications and services were marked as unknown.

7. Resource Utilization

The system utilizes the following resources for operation:

Total Memory Usage: 608 bytes.

Main Memory Usage: 18.7 KB (Pattern Memory).



Flow Management Memory Usage: 22.3 KB.

8. Snort++ Performance

The performance of Snort++ during the analysis period is as follows:

Total Packets Analyzed: 162

Total Processing Time: 9 minutes and 46 seconds.

Elapsed Time: 586 seconds.

9. Results and Recommendations

Snort++ continues to operate effectively, ensuring network security by monitoring various protocols over the network.

Detected Threats: Port scan and ARP spoofing attacks have been identified, helping to prevent potential threats on the network.

Network Flow Timeouts: Some UDP sessions experienced timeouts, and flow management could be improved for better network performance.

Unidentified Applications: Some applications remain unidentified. Additional configurations or identification procedures could be implemented to improve detection accuracy.