Day 03 Task Report: Network Security Monitoring and Analysis

- **Prepared by:** Girija Shankar Sahoo
- **Date:** July 29, 2025

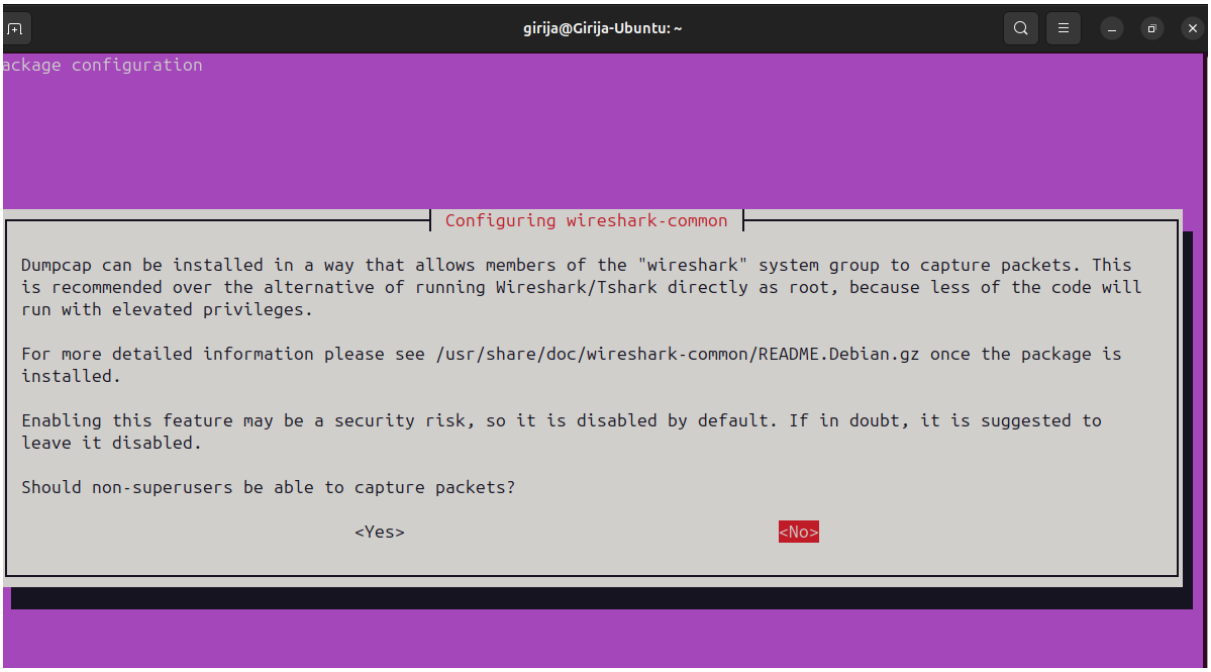
1. Executive Summary

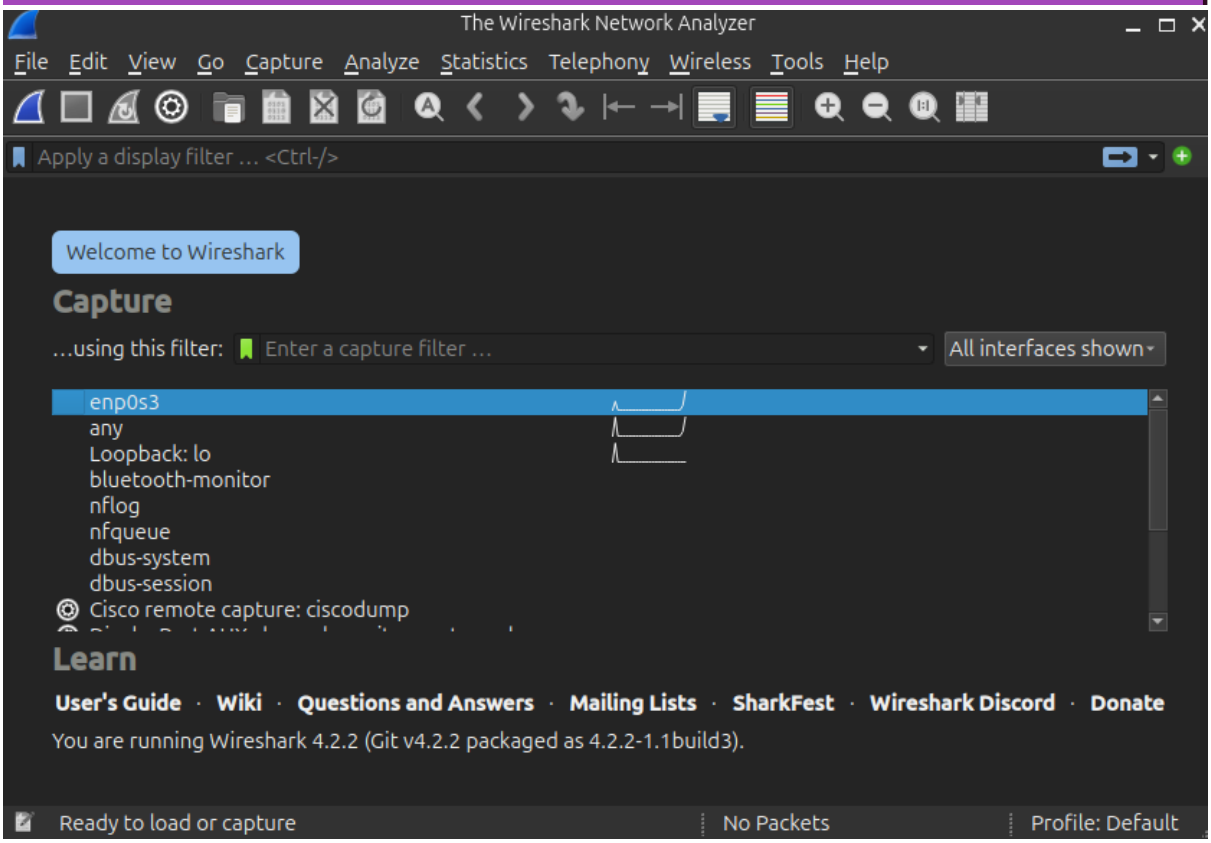
This report details the execution of a network security monitoring exercise involving three core components: passive traffic analysis with Wireshark, active intrusion detection with Snort, and custom packet generation with Scapy. The objective was to capture and analyze network behavior, configure an Intrusion Detection System (IDS) to detect suspicious patterns, and validate the system's effectiveness. All objectives were successfully met, demonstrating a complete workflow from analysis to detection and testing.

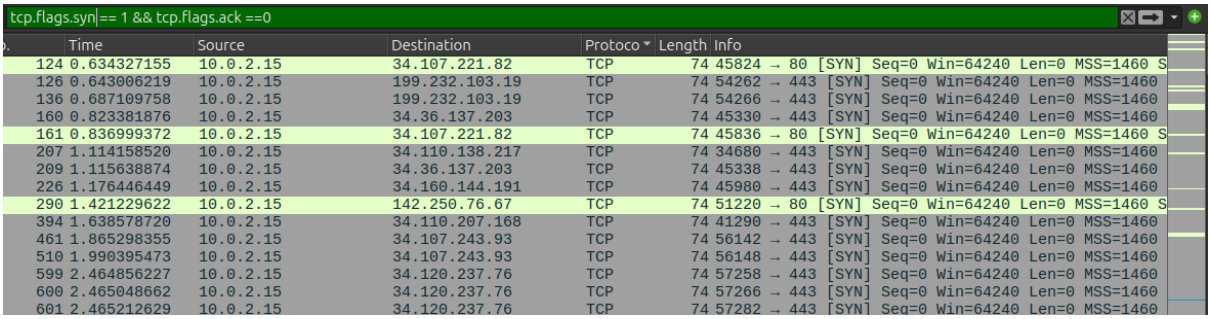
2. Task I: Network Traffic Analysis with Wireshark

- **Objective:** To passively capture and analyze live network traffic for 10 minutes to identify common protocols, communication patterns, and any potentially suspicious activity.
- **Methodology:** Wireshark was used on a Linux virtual machine to capture network traffic. Display filters were applied to isolate traffic by protocol (tcp, udp, icmp) and to investigate specific patterns, such as TCP SYN packets (tcp.flags.syn == 1 && tcp.flags.ack == 0).
- **Findings:** The traffic capture was dominated by TCP (HTTP/TLS) and UDP (DNS) packets, consistent with standard web Browse activity. Analysis of the conversations confirmed communication with expected public servers. No overtly malicious or anomalous traffic was identified during the passive monitoring phase.

```
girija@Girija-Ubuntu:~$ sudo apt install wireshark
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libb2-1 libbcg729-0 libdouble-conversion3 liblua5.2-0 libmd4c0 libminizip1t64 libnghttp3-3 libopencore-amrnb0
  libpcre2-16-0 libqt6core5compat6 libqt6core6t64 libqt6dbus6t64 libqt6gui6t64 libqt6multimedia6 libqt6network6t64
  libqt6opengl6t64 libqt6printsupport6t64 libqt6qml6 libqt6qmlmodels6 libqt6quick6 libqt6svg6 libqt6waylandclient6
  libqt6waylandcompositor6 libqt6waylandeglclienthwintegration6 libqt6waylandeglcompositorhwintegration6
  libqt6widgets6t64 libqt6wlshellintegration6 libsmi2t64 libspandsp2t64 libtsnet0t64 libwireshark-data libwireshark17t64
  libwiretap14t64 libwsutil15t64 qt6-gtk-platformtheme qt6-qpa-plugins qt6-translations-l10n qt6-wayland
  wireshark-common
Suggested packages:
  qt6-qmltooling-plugins snmp-mibs-downloader geoipupdate geoip-database geoip-database-extra libjs-leaflet
  libjs-leaflet.markercluster wireshark-doc
The following NEW packages will be installed:
  libb2-1 libbcg729-0 libdouble-conversion3 liblua5.2-0 libmd4c0 libminizip1t64 libnghttp3-3 libopencore-amrnb0
  libpcre2-16-0 libqt6core5compat6 libqt6core6t64 libqt6dbus6t64 libqt6gui6t64 libqt6multimedia6 libqt6network6t64
  libqt6opengl6t64 libqt6printsupport6t64 libqt6qml6 libqt6qmlmodels6 libqt6quick6 libqt6svg6 libqt6waylandclient6
```


girija@Girija-Ubuntu: ~
ackage configuration
Configuring wireshark-common
Dumpcap can be installed in a way that allows members of the "wireshark" system group to capture packets. This is recommended over the alternative of running Wireshark/Tshark directly as root, because less of the code will run with elevated privileges.
For more detailed information please see /usr/share/doc/wireshark-common/README.Debian.gz once the package is installed.
Enabling this feature may be a security risk, so it is disabled by default. If in doubt, it is suggested to leave it disabled.
Should non-superusers be able to capture packets?
<Yes>
<No>


The Wireshark Network Analyzer
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter … <Ctrl-/>
Welcome to Wireshark
Capture
…using this filter: Enter a capture filter …
All interfaces shown
enp0s3
any
Loopback: lo
bluetooth-monitor
nflog
nfqueue
dbus-system
dbus-session
Cisco remote capture: ciscodump
Learn
User's Guide · Wiki · Questions and Answers · Mailing Lists · SharkFest · Wireshark Discord · Donate
You are running Wireshark 4.2.2 (Git v4.2.2 packaged as 4.2.2-1.1build3).
Ready to load or capture
No Packets
Profile: Default

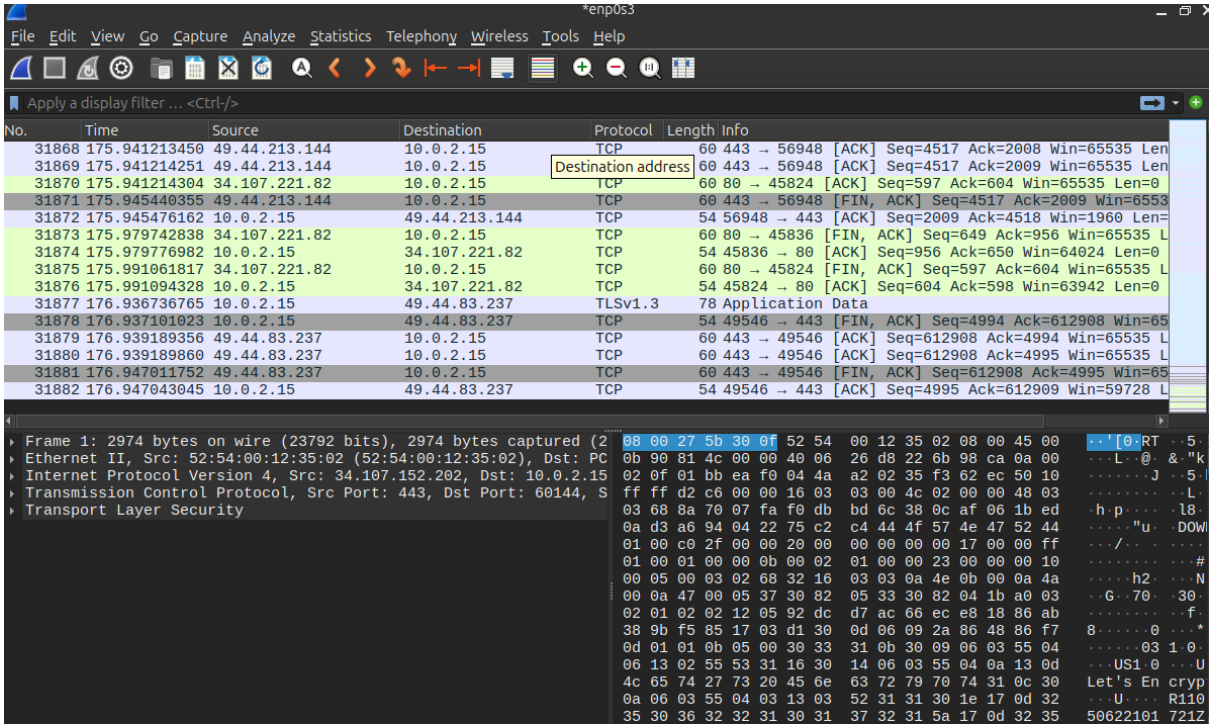


3. Task II: Intrusion Detection with Snort

- **Objective:** To configure Snort as a network Intrusion Detection System (IDS) and write custom rules to detect two specific events: the presence of a TCP SYN packet and a potential port scan.
- **Configuration:** Snort was installed and configured on a Linux VM. The primary configuration file (/etc/snort/snort.conf) was verified, and the following custom rules were added to /etc/snort/rules/local.rules:
- # Rule to detect any TCP SYN packet
- alert tcp any any -> $HOME_NET any (msg:"SYN Packet Detected"; sid:1000001; rev:1;)
-
- # Rule to detect traffic to ports 1-1024, indicating a potential port scan
- alert tcp any any -> $HOME_NET 1:1024 (msg:"Potential Port Scan"; sid:1000002; rev:1;)
- **Test Results:** The Snort IDS was run in console mode to monitor live traffic. Test traffic generated by the Scapy script successfully triggered both custom rules. Alerts for "SYN Packet Detected" and "Potential Port Scan" were observed in the console output, confirming the ruleset was functioning correctly.

```
girija@Girija-Ubuntu:~$ sudo snort -T -c /etc/snort/snort.conf -i eth0
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 3
02 4343 4848 5250 6988 7000:7001 7144:7145 7510 7777 7779 8000 8008 8014 8028 8080 8085 8088 8090 8118 8123 8180:8181 8
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
Tagged Packet Limit: 256
Loading dynamic engine /usr/lib/snort/snort_dynamicengine/libsf_engine.so... done
Loading all dynamic detection libs from /usr/lib/snort/snort_dynamicrules...
WARNING: No dynamic libraries found in directory /usr/lib/snort/snort_dynamicrules.
```

- 

4. Task III: Test Traffic Generation with Scapy

- **Objective:** To develop a Python script to generate specific network packets capable of testing the custom Snort rules.
- **Methodology:** A script named test_packets.py was created using the Scapy library. The script was designed with two functions: one to send a single TCP SYN packet to a target host, and another to simulate a port scan by sending a burst of TCP SYN packets to a range of ports (1-1024).
- **Outcome:** The script successfully generated the required test traffic. Executing its functions while Snort was active resulted in the immediate triggering of the corresponding IDS alerts, validating the end-to-end detection and testing workflow.

---

5. Conclusion and Key Learnings

This project provided comprehensive, hands-on experience in network monitoring and defense. The exercise demonstrated how passive analysis with a tool like Wireshark can establish a baseline of normal network behavior. Building on this, the project showed how an IDS like Snort can be configured to actively alert on deviations from this baseline or on known patterns of malicious activity. Finally, using Scapy to generate specific traffic underscored the importance of being able to test and validate security rules to ensure they are effective. The complete workflow represents a foundational element of modern cybersecurity operations.