



Network Traffic Analyzer and Firewall

Mohab Hassan	202001315
Mohamed Shawky	202000218
Omar Arafa	202000652
Seif Abdelrahman	202001473
Zeyad Elkahwagi	202000963

Presented to:

Dr. Ahmed Fathy Elnokrashy

TA. Mariam Wael Fathy

TA. Malak Magdy Abdelsattar

Table of Contents

Table of Contents

Table of Contents	2
Section 1: Abstract.....	4
Section 2: Introduction	4
Section 3: Literature review.....	5
3.1Network Fundamentals.....	5
3.2 Network Traffic Analyzer and Firewall.....	7
Section 4: Methodology	8
Section 5: Results and Analysis	13
Section 6: Conclusion.....	15
References.....	16

Section 1: Abstract

This paper presents the development of an advanced network traffic analyzer designed to enhance network security through real-time monitoring and automated firewall rule implementation. Utilizing `tshark` for packet analysis, the system extracts key metrics from network traffic, such as HTTP and HTTPS packet counts and the top source and destination IP addresses. By identifying anomalies and distinguishing between legitimate and suspicious traffic through a configurable threshold and whitelist, the analyzer effectively detects potential security threats. Upon detecting suspicious activity, the system automatically implements `iptables` firewall rules to block malicious IP addresses. Experimental results demonstrate the tool's efficacy in providing in-depth network insights and dynamically securing the network environment, representing a significant advancement in proactive network defense strategies.

Section 2: Introduction

The exponent necessitates network traffic due to IoT, cloud computing, and mobile applications necessitate advanced network monitoring and security measures. This paper introduces a network traffic analyzer that monitors, detects,

and mitigates suspicious activities by leveraging Wireshark's `tshark` for packet analysis. The system identifies anomalies based on traffic patterns and implements firewall rules using `iptables` to block malicious IP addresses. By integrating a configurable threshold and a whitelist of trusted IPs, it effectively reduces false positives and enhances network security. This research demonstrates the tool's ability to provide real-time insights and proactive defense, significantly improving network security management.

Section 3: Literature review

3.1 Network Fundamentals

A network is a group of interconnected devices that share resources and information, each identified by an IP address. IP addresses can be private (used within local networks and not routable on the internet) or public (unique across the internet and assigned by ISPs). Websites have unique public IP addresses; when you enter a URL, DNS translates it into the corresponding IP address, allowing your device to communicate with the website's server. A network's public IP, given by the ISP, enables external communication, while internal devices use private IPs assigned by the router for local communication. This ensures efficient interaction within the network and with the wider internet.

3.2 Network Traffic Analyzer and Firewall

Network traffic analyzers and firewalls are foundational components in cybersecurity, playing critical roles in monitoring and securing network environments. Network traffic analyzers, such as Wireshark and its command-line utility `tshark`, are widely used to capture, analyze, and visualize packet-level data. These tools provide comprehensive insights into network activity, facilitating the identification of various traffic patterns, detection of anomalies, and diagnosis of network issues. By examining packet headers and payloads, traffic analyzers can uncover detailed information about protocols, source and destination addresses, and data flows, enabling network administrators to make informed decisions about network management and security.

Firewalls, meanwhile, act as a vital barrier between trusted internal networks and untrusted external networks, enforcing security policies by controlling the flow of incoming and outgoing traffic. Tools like `iptables` in Linux environments provide robust mechanisms for defining and enforcing these security rules. Modern firewalls can dynamically adjust to detected threats, offering adaptive security measures in response to evolving attack vectors. This adaptability is crucial for maintaining network security in the face of sophisticated and persistent cyber threats.

The integration of network traffic analysis with automated firewall rule implementation has emerged as a powerful strategy in enhancing network security. By continuously monitoring network traffic, these integrated systems can promptly detect suspicious activities and anomalies that may indicate security breaches. Upon detection, they can automatically implement firewall rules to block malicious IP addresses, thereby providing a rapid response to potential threats. This approach not only improves the efficacy of threat detection and mitigation but also reduces the manual effort required for network security management.

Recent advancements in this field focus on enhancing the accuracy of anomaly detection algorithms to minimize false positives and improve the reliability of threat identification. The use of configurable thresholds and whitelists allows these systems to differentiate between legitimate high-volume traffic and actual security threats, ensuring that essential services remain uninterrupted while maintaining robust security. Furthermore, the integration of machine learning techniques into traffic analysis and firewall management is being explored to further enhance the predictive capabilities and responsiveness of these systems.

Section 4: Methodology

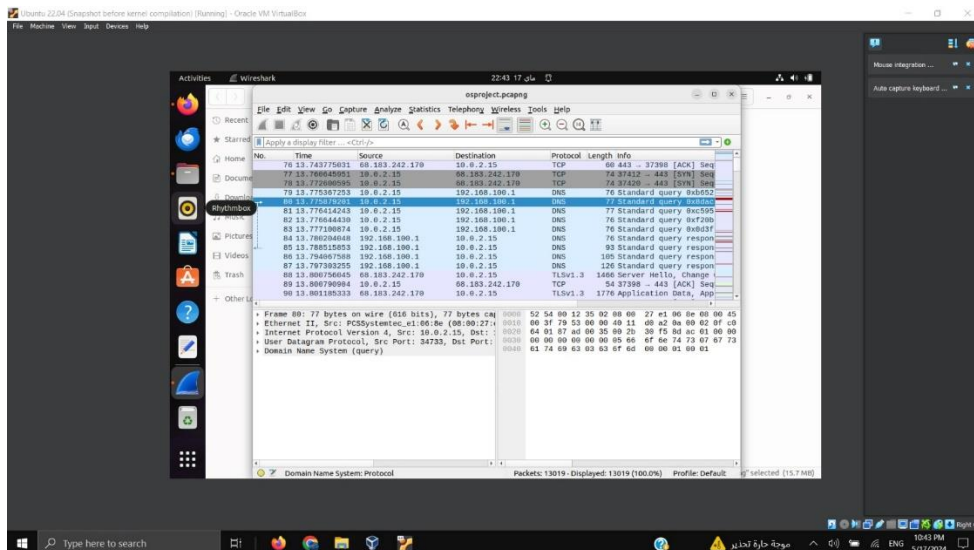
The methodology for developing and implementing the network traffic analyzer and firewall rule automation involves the following steps:

The methodology for developing and implementing the network traffic analyzer and firewall rule automation involves the following steps:

1. Wireshark Packet Capture:

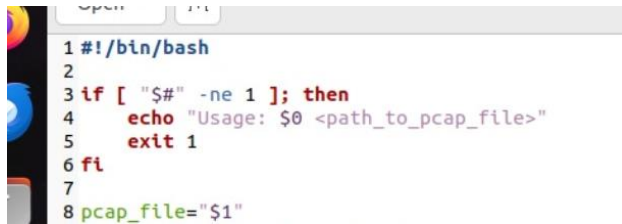
- The process begins with capturing network traffic using Wireshark, a powerful packet analyzer. Wireshark captures packets traversing the network interface and saves them to a pcap file.

- This pcap file serves as the input for subsequent analysis and anomaly detection.



2. Script Initialization:

- The script starts by verifying that exactly one argument (the path to a pcap file) is provided.

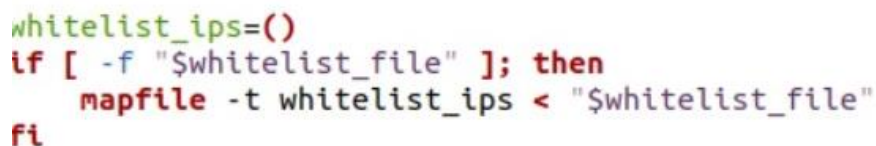


```
1#!/bin/bash
2
3if [ "$#" -ne 1 ]; then
4    echo "Usage: $0 <path_to_pcap_file>"
5    exit 1
6fi
7
8pcap_file="$1"
```

3. Loading Whitelist:

- The script checks for the existence of a `whitelist.txt` file, which contains IP addresses that should be neglected from being flagged as suspicious.

- If the file is found, it reads the IP addresses into an array (`whitelist_ips`).



```
whitelist_ips=()
if [ -f "$whitelist_file" ]; then
    mapfile -t whitelist_ips < "$whitelist_file"
fi
```

4. Whitelist Check Function:

- A function named `is whitelisted` is defined to check if a given IP address is in the whitelist array. It returns true if the IP is whitelisted and false otherwise.


```

9 whitelist_file="whitelist.txt"
0
1 whitelist_ips=()
2 if [ -f "$whitelist_file" ]; then
3     mapfile -t whitelist_ips < "$whitelist_file"
4 fi
5
6 is_whitelisted() {
7     local ip="$1"
8     for whitelist_ip in "${whitelist_ips[@]}; do
9         if [ "$ip" == "$whitelist_ip" ]; then
0             return 0
1         fi
2     done
3     return 1
4 }
5

```

5. Traffic Analysis:

- The 'analyze traffic' function uses 'tshark' to analyze the provided pcap file. It calculates:

- The total number of packets.
- The number of HTTP packets.
- The number of HTTPS packets.
- The top 10 source IP addresses by packet count.
- The top 10 destination IP addresses by packet count.
- The results are then printed in a formatted report.

```

analyze_traffic() {
    total_packets=$(tshark -r "$pcap_file" 2>/dev/null | wc -l)
    http_packets=$(tshark -r "$pcap_file" -Y "http" 2>/dev/null | wc -l)
    https_packets=$(tshark -r "$pcap_file" -Y "tls" 2>/dev/null | wc -l)

    top_src_ips=$(tshark -r "$pcap_file" -T fields -e ip.src 2>/dev/null | sort | uniq -c | sort -nr | head -n 10)
    top_dst_ips=$(tshark -r "$pcap_file" -T fields -e ip.dst 2>/dev/null | sort | uniq -c | sort -nr | head -n 10)

    echo "----- Network Traffic Analysis Report -----"
    echo "1. Total Packets: $total_packets"
    echo "2. Protocols:"
    echo "   - HTTP: $http_packets packets"
    echo "   - HTTPS/TLS: $https_packets packets"

    echo ""
    echo "3. Top 10 Source IP Addresses:"
    echo "$top_src_ips"
    echo ""
    echo "4. Top 10 Destination IP Addresses:"
    echo "$top_dst_ips"
    echo ""
    echo "----- End of Report -----"
}

```

6. Anomaly Detection:

- The 'detect_anomalies' function identifies IP addresses with an unusually high number of packets, using a threshold value (default is 1000 packets).
- It uses 'tshark' and 'awk' to find IP addresses exceeding this threshold and lists them as potential anomalies.
- Each identified IP address is checked against the whitelist. If not whitelisted, it is marked as suspicious.

```

~
9 detect_anomalies() {
0     local threshold=1000
1     echo "Checking for anomalies..."
2     anomalies=$(tshark -r "$pcap_file" -T fields -e ip.src 2>/dev/null | sort | uniq -c | awk -v threshold="$threshold" '{s1 >
threshold {print $2}}')
3
4     if [ -n "$anomalies" ]; then
5         echo "Suspicious patterns detected from the following IP addresses:"
6         for anomaly in $anomalies; do
7             if ! is_whitelisted "$anomaly"; then
8                 echo "$anomaly"
9             else
0                 echo "$anomaly is whitelisted."
1             fi
2         done
3         echo ""
4         implement_firewall_rules "$anomalies"
5     else
6         echo "No anomalies detected."
7     fi
8 }
~

```

7. Firewall Rule Implementation:

- The 'implement_firewall_rules' function takes the list of suspicious IP addresses and implements firewall rules to block them.
- For each suspicious IP that is not whitelisted, the script uses 'iptables' to add rules that drop both incoming and outgoing traffic to and from the IP address.
- The success or failure of rule implementation is logged.

```
implement_firewall_rules() {  
    local ips="$1"  
    for ip in $ips; do  
        if ! is_whitelisted "$ip"; then  
            echo "Blocking suspicious IP address: $ip"  
            if sudo iptables -A INPUT -s "$ip" -j DROP && sudo iptables -A OUTPUT -d "$ip" -j DROP; then  
                echo "Firewall rules implemented."  
            else  
                echo "Failed to implement firewall rules for IP address: $ip"  
            fi  
        fi  
    done  
}
```

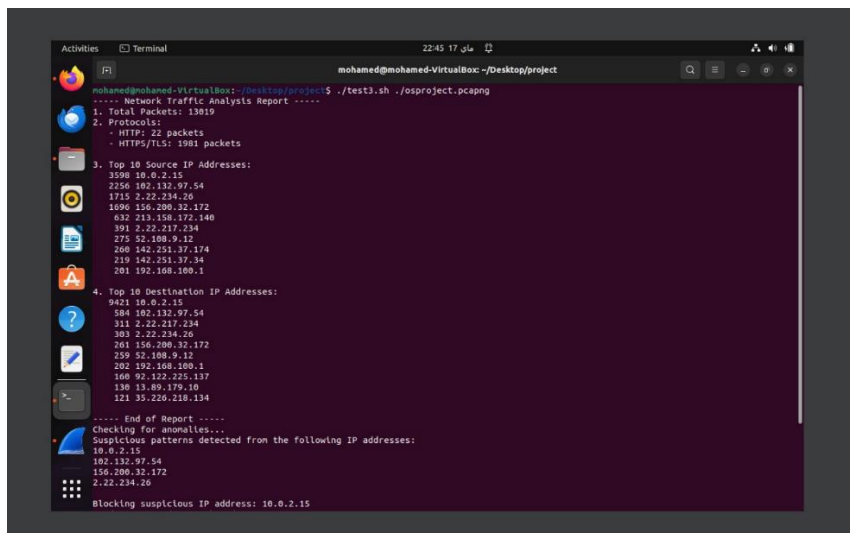
8. Execution Flow:

- The script first runs 'analyze_traffic' to generate the traffic analysis report.
- It then calls 'detect_anomalies' to check for suspicious traffic patterns.
- If anomalies are detected, Preventing Accidental Changes: Requiring 'sudo' prompts users to enter their password before executing privileged commands, adding an extra layer of protection against accidental or malicious modifications to

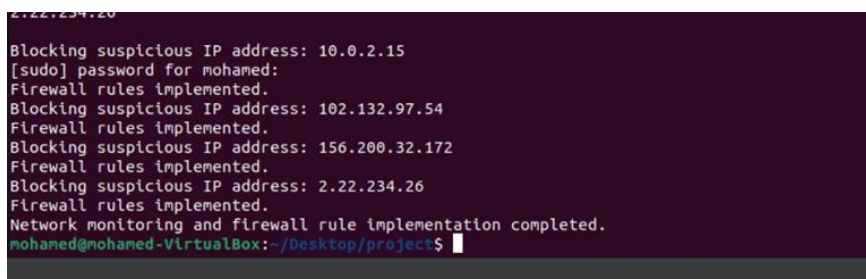
firewall rules, 'implement_firewall_rules' is invoked to block the identified suspicious IP addresses.

- Finally, the script outputs a completion message.

Section 5: Results and Analysis



```
mohamed@mohamed-VirtualBox: ~/Desktop/project
mohamed@mohamed-VirtualBox:~/Desktop/project$ ./test3.sh ./osproject.pcapng
----- Network Traffic Analysis Report -----
1. Total Packets: 13019
2. Protocols:
  - HTTP: 22 packets
  - HTTPS/TLS: 1991 packets
3. Top 10 Source IP Addresses:
  3590 10.0.2.15
  2256 102.132.97.54
  1715 2.22.234.26
  1696 156.200.32.172
  632 213.158.172.140
  391 2.22.217.234
  275 52.108.9.12
  260 142.251.37.174
  219 142.251.37.34
  201 192.168.100.1
4. Top 10 Destination IP Addresses:
  9411 10.0.2.15
  584 102.132.97.54
  311 2.22.217.234
  303 2.22.234.26
  261 156.200.32.172
  259 52.108.9.12
  202 192.168.100.1
  160 92.122.225.137
  130 13.89.179.10
  121 35.226.216.134
----- End of Report -----
Checking for anomalies...
Suspicious patterns detected from the following IP addresses:
10.0.2.15
102.132.97.54
156.200.32.172
2.22.234.26
Blocking suspicious IP address: 10.0.2.15
```



```
Blocking suspicious IP address: 10.0.2.15
[sudo] password for mohamed:
Firewall rules implemented.
Blocking suspicious IP address: 102.132.97.54
Firewall rules implemented.
Blocking suspicious IP address: 156.200.32.172
Firewall rules implemented.
Blocking suspicious IP address: 2.22.234.26
Firewall rules implemented.
Network monitoring and firewall rule implementation completed.
mohamed@mohamed-VirtualBox:~/Desktop/project$
```

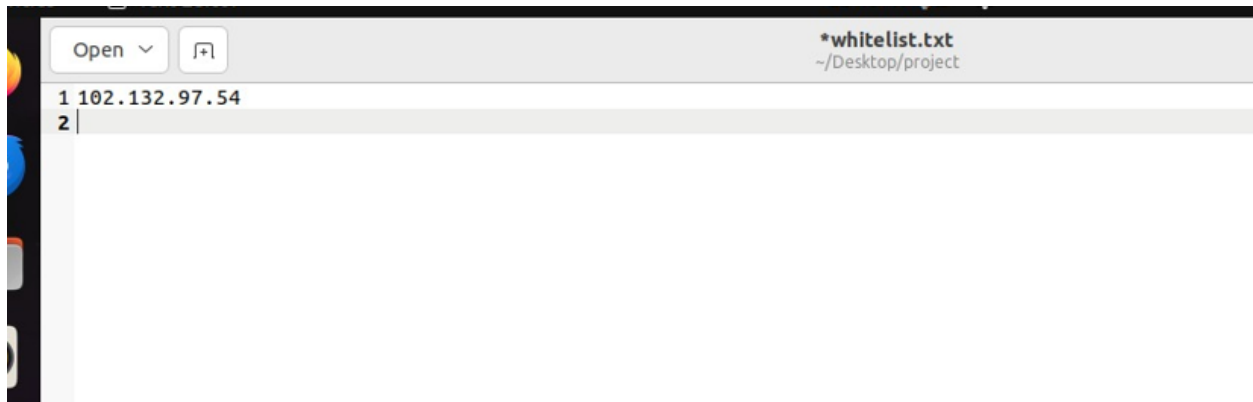
What happened step by step?

-The script takes a pcapng file as input, which contains captured network traffic data.

- Traffic Analysis: Using tshark, the script counts the total packets and counts HTTP and HTTPS packets. Identifies the top 10 source and destination IP addresses by packet count.

- Anomaly Detection: It identifies IP addresses with more than 1000 packets that are not on the whitelist.

- Firewall Rules: For each detected anomaly, the script uses iptables to block incoming and outgoing traffic for those IP addresses.



Here we putted an IP address which contain more than 1000 packets as an example in whitelist.txt

```
----- End of Report -----  
Checking for anomalies...  
Suspicious patterns detected from the following IP addresses:  
10.0.2.15  
102.132.97.54 is whitelisted.  
156.200.32.172  
2.22.234.26  
  
Blocking suspicious IP address: 10.0.2.15  
Firewall rules implemented.  
Blocking suspicious IP address: 156.200.32.172  
Firewall rules implemented.  
Blocking suspicious IP address: 2.22.234.26  
Firewall rules implemented.  
Network monitoring and firewall rule implementation completed.  
mohamed@mohamed-VirtualBox:~/Desktop/project$
```

When executed again it blocked all the suspicious Ip addresses except the Ip address we putted in whitelist.txt

Section 5: Conclusion

In conclusion, the developed network traffic analyzer, coupled with automated firewall rule implementation, represents a significant advancement in proactive network security. By leveraging packet analysis techniques and dynamic rule enforcement, the system provides real-time insights into network activity and effectively mitigates potential threats. The integration of anomaly detection algorithms and whitelist mechanisms enhances the accuracy of threat identification while minimizing false positives. This approach not only strengthens network defense but also streamlines security management processes.

Section 6: References

Wahid, A., Firdaus, M. E., & Parenreng, J. M. (2021). Implementation of wireshark and IP tables firewall.

<https://media.neliti.com/media/publications/558802-the-implementation-of-wireshark-and-ip-tables-firewall.pdf>

Brozek, M. (2020, March 19). How to use your firewall for network traffic analysis. Palo Alto Networks Blog.

<https://www.paloaltonetworks.com/blog/2020/02/cortex-network-traffic-analysis/>

Pandit, P. (21AD). study and analysis of Firewalls.

https://www.researchgate.net/publication/355040238_Study_and_Analysis_of_Firewalls

Lal, V. (n.d.). Network traffic analyzer and abstract. Scribd.

<https://www.scribd.com/document/82468646/Network-Traffic-Analyzer-and-Abstract-1>

What is network traffic analysis? | vmware. (n.d.-b).

<https://www.vmware.com/topics/glossary/content/network-traffic-analysis.html>