

**AMERICAN UNIVERSITY OF SCIENCE & TECHNOLOGY**

FACULTY OF ENGINEERING AND COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

Linux Kernel Customization & Shell Programming Lab

ICT350L

Final Project Report

**TCPDUMP Configuration Menu**

Prepared by

Ralph Achkar 12210343

Presented to

Dr. Alexandre Sayegh

May 28, 2024

Achrafieh Campus

**ABSTRACT**

*This report presents a comprehensive guide on configuring and utilizing TCPDUMP on a Linux system. TCPDUMP is a powerful network packet analyzer that allows users to capture and inspect network traffic in real time. The report begins with an overview of TCPDUMP, highlighting its significance in network diagnostics and security analysis. It then details the installation process on various Linux distributions, ensuring accessibility for different user environments. The core of the report focuses on the configuration of TCPDUMP, including essential command-line options, filtering expressions, and practical examples for capturing specific types of traffic. Advanced usage scenarios are explored, such as remote packet capturing, analyzing packet data with Wireshark, and integrating TCPDUMP with other network monitoring tools. Additionally, the report addresses common challenges and troubleshooting techniques to optimize TCPDUMP performance. The contents of this report are what we used throughout the project to have a thorough understanding of how to configure and use TCPDUMP effectively for network analysis on Linux systems, leveraging automation through customized scripting to enhance their network monitoring capabilities.*

List of Figures

[Figure 1. Flowchart 5](#_Toc167549603)

[Figure 2. tcp\_functions\_script Part 1 6](#_Toc167549604)

[Figure 3. tcp\_functions\_script Part 2 7](#_Toc167549605)

[Figure 4. tcp\_functions\_script Part 3 8](#_Toc167549606)

[Figure 5. tcp\_code of menu.sh 9](#_Toc167549607)

[Figure 6. tcp main menu 10](#_Toc167549608)

[Figure 7. Capturing packets 10](#_Toc167549609)

[Figure 8. Ending the capturing 11](#_Toc167549610)

[Figure 9. Displaying pcap files 11](#_Toc167549611)

[Figure 10. reading a pcap file 12](#_Toc167549612)

[Figure 11. Login user option 12](#_Toc167549613)

[Figure 12. saved log file 13](#_Toc167549614)

[Figure 13. users per port 13](#_Toc167549615)

[Figure 14. Capturing Packet for a specific IP 14](#_Toc167549616)

[Figure 15. Number of online users 14](#_Toc167549617)

[Figure 16. Filter for specific IP using browser 15](#_Toc167549618)

[Figure 17. Backup and delete old file 15](#_Toc167549619)

[Figure 18. The backup file 16](#_Toc167549620)

[Figure 18. Exiting 16](#_Toc167549621)

Table of Contents

[I. Introduction 4](#_Toc167549641)

[Overview of TCPDUMP 4](#_Toc167549642)

[Importance in Network Diagnostics and Security Analysis 4](#_Toc167549643)

[II. Installation 4](#_Toc167549644)

[Installing TCPDUMP on Various Linux Distributions 4](#_Toc167549645)

[Configuration and Usage 4](#_Toc167549646)

[Basic Command-Line Options 4](#_Toc167549647)

[Filtering Expressions 5](#_Toc167549648)

[III. Design and Procedure 5](#_Toc167549649)

[Code Explanation 6](#_Toc167549650)

[Outputs 10](#_Toc167549651)

[Capture Network Traffic 11](#_Toc167549652)

[Display Captured Traffic 12](#_Toc167549653)

[Log User Login Information 13](#_Toc167549654)

[Count Users per Port 14](#_Toc167549655)

[Filter Traffic by IP Address 15](#_Toc167549656)

[Count Online Users within Time Range 15](#_Toc167549657)

[Filter Specific IP Visiting Web Browser 15](#_Toc167549658)

[Backup PCAP Files 16](#_Toc167549659)

[IV. Problem Faced and Solutions 17](#_Toc167549660)

[V. Conclusion 18](#_Toc167549661)

[References 19](#_Toc167549662)

[Appendix A 20](#_Toc167549663)

# Introduction

## Overview of TCPDUMP

This section provides a brief introduction to TCPDUMP, explaining what it is and its primary functions. TCPDUMP is a command-line packet analyzer tool used to capture and display network traffic. It is essential for network diagnostics, troubleshooting, and security analysis.

## Importance in Network Diagnostics and Security Analysis

This part discusses why TCPDUMP is crucial for network administrators and security professionals. It highlights how TCPDUMP helps in identifying network issues, analyzing network performance, detecting security breaches, and monitoring network traffic for unusual activity.

# Installation

## Installing TCPDUMP on Various Linux Distributions

This section covers the step-by-step process of installing TCPDUMP on different Linux distributions such as Ubuntu, CentOS, and Fedora. It includes commands for installing TCPDUMP using package managers like apt, yum, and dnf.[1]

## Configuration and Usage

### Basic Command-Line Options

Here, the fundamental command-line options of TCPDUMP are explained. This includes options for specifying the network interface, setting capture filters, saving captured packets to a file, and adjusting the verbosity of output.

### Filtering Expressions

This part delves into the syntax and usage of filtering expressions in TCPDUMP. Filtering expressions are used to capture only specific types of traffic based on criteria like IP addresses, ports, protocols, and packet content.

# Design and Procedure

The flowchart represents a script for managing network traffic capture and analysis using TCPDUMP on a Linux system. It begins with the script's Start point, leading to the Main Menu where the user is presented with various options. These options include Capture Traffic, which initiates network traffic capture and saves it to a PCAP file, Display Traffic, which lists and allows reading from saved PCAP files, Log User Login, which logs user login events, User Per Port, which counts users on specified ports, Filter Traffic, which filters network traffic based on an IP address and saves the filtered data to a PCAP file, Online Users, which counts online users within a specified time range, Filter IP, which filters traffic for a specific IP accessing web services and saves it, and Backup PCAP Files, which backs up existing PCAP files and optionally deletes old ones. Each function returns to the Main Menu upon completion, allowing the user to perform another action or choose to Exit, terminating the script at the End point.

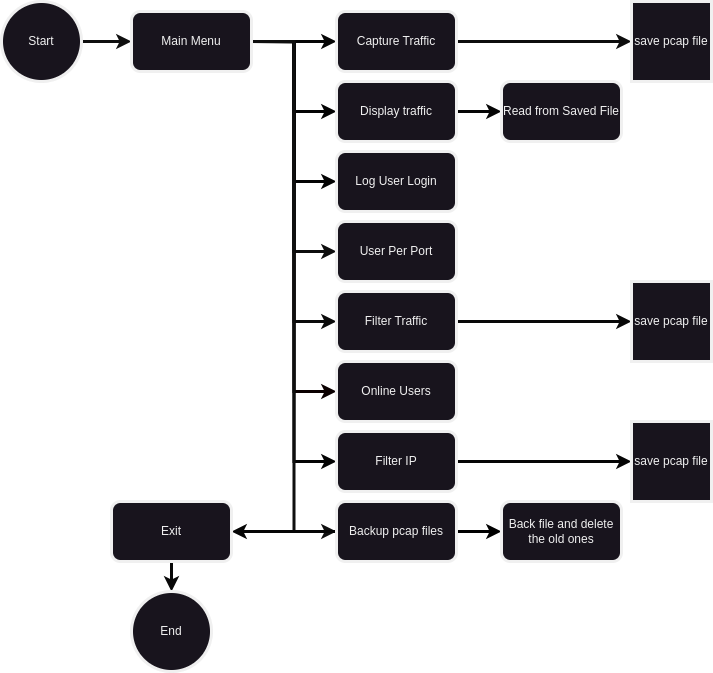


Figure 1. Flowchart

## Code Explanation

**File 1: tcp\_functions.sh**

The provided script is a comprehensive Bash program for network traffic capture and analysis using `tcpdump` on a Linux system. It begins with a function to capture network traffic, saving it to a file while handling interruptions gracefully. Another function allows users to display captured traffic files by listing available PCAP files and selecting one to read using `tcpdump`. Additionally, the script includes a function to log user login information, utilizing `journalctl` to append session details to a log file.

Further functionality includes counting the number of users accessing specified ports using `netstat` and filtering network traffic based on an IP address, saving the filtered traffic to a PCAP file. The script also features a function to count online users within a specified time range, leveraging the `who` command and `awk` for filtering. Moreover, it provides a method to filter traffic for a specific IP address accessing web services and back up PCAP files by compressing them into a tarball. The backup function also includes options for deleting the original files either automatically or interactively. Each function handles errors and interruptions to ensure robust operation.

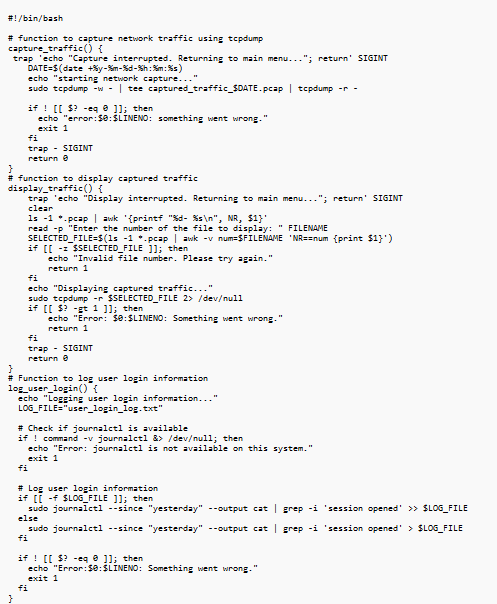


Figure 2. tcp\_functions\_script Part 1

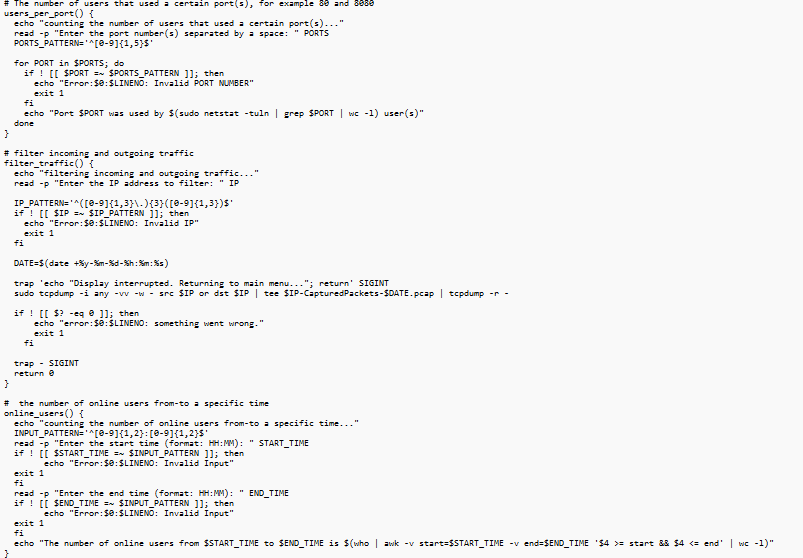


Figure 3. tcp\_functions\_script Part 2

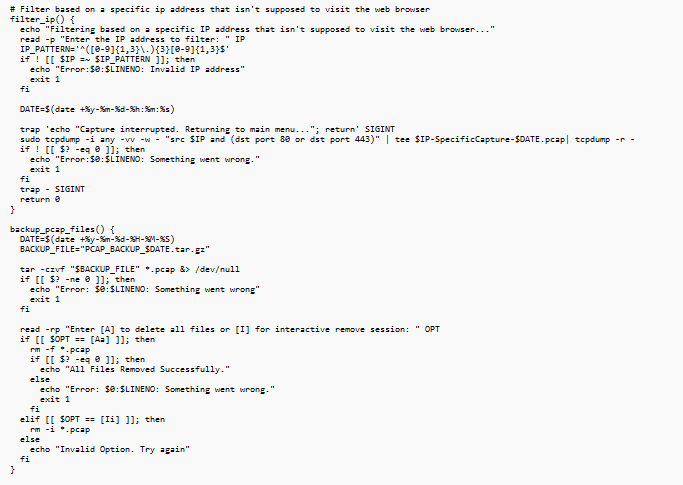


Figure 4. tcp\_functions\_script Part 3

**File 2: menu.sh**

This script is a Bash program designed to facilitate network traffic management and analysis using `tcpdump` on a Linux system. It starts by setting strict error handling with `set -uo pipe fail` and sourcing a library of TCP dump functions from a specified path. The core functionality is encapsulated in the `main\_menu` function, which dynamically retrieves and displays the host and default gateway IP addresses. The menu presents users with a variety of options, including starting a network traffic capture, displaying captured traffic, logging user login information, counting users on specific ports, filtering traffic based on IP addresses, counting online users within a time range, filtering specific IP traffic accessing web services, and backing up PCAP files. Each option invokes a corresponding function, facilitating diverse network analysis tasks.

The `main` function perpetually invokes the `main\_menu` in a loop, allowing users to perform multiple actions sequentially without restarting the script. It waits for user input to continue after each task and then clears the screen for the next operation. The script exits the loop and terminates if the user selects the exit option from the menu. This design ensures a user-friendly interface for managing network traffic, providing a comprehensive toolset for network diagnostics and security analysis through an interactive menu-driven approach.

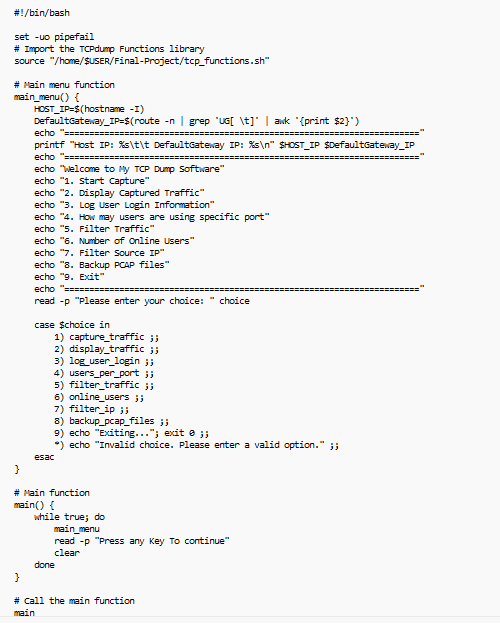


Figure 5. tcp\_code of menu.sh

## Outputs

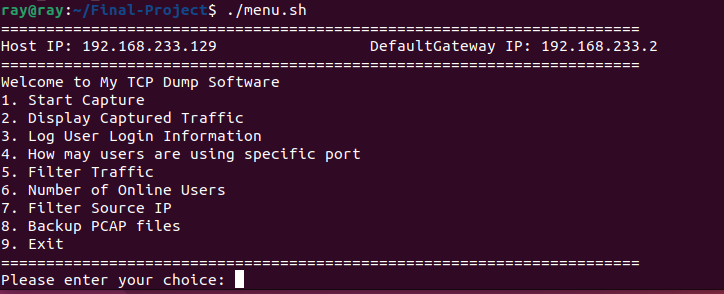


Figure 6. tcp main menu

## Capture Network Traffic

**Detailed Steps for Capturing Network Traffic**

This part provides a detailed explanation of how the script captures network traffic using TCPDUMP. It includes information on starting the capture, saving the captured data to a file, and handling interruptions and errors during the capture process.[2]

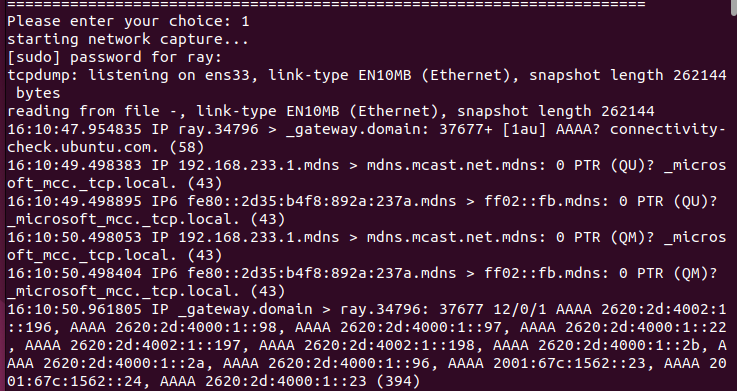


Figure 7. Capturing packets

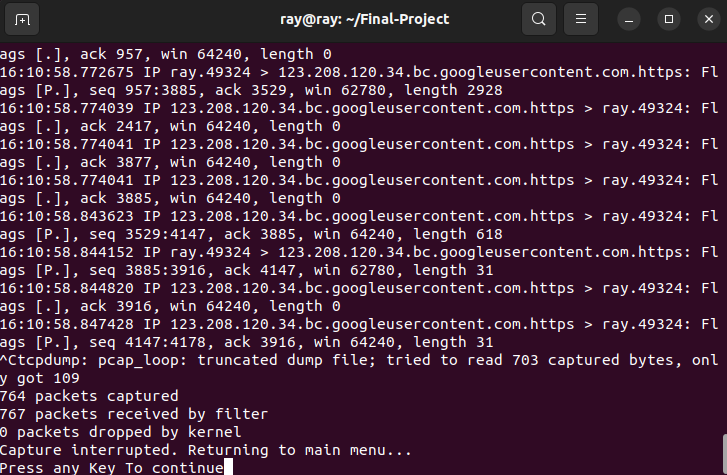


Figure 8. Ending the capturing

## Display Captured Traffic

**Process for Displaying Captured Traffic Files**

This section explains how the script lists and allows users to select and display previously captured traffic files. It details the user interaction for selecting a file and the command used to read and display the contents of the capture file.

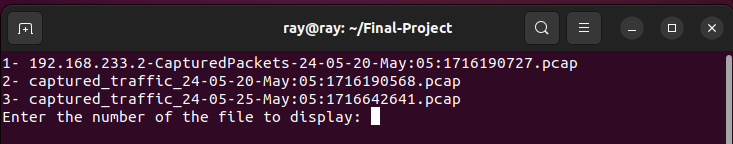


Figure 9. Displaying pcap files



Figure 10. reading a pcap file

## Log User Login Information

**Steps to Log User Login Information Using `journalctl`**

Here, the process of logging user login information using the `journalctl` command is explained. This includes checking for the availability of `journalctl`, filtering the logs for user session events, and writing the information to a log file.[4]

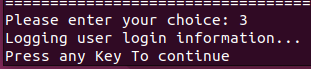


Figure 11. Login user option

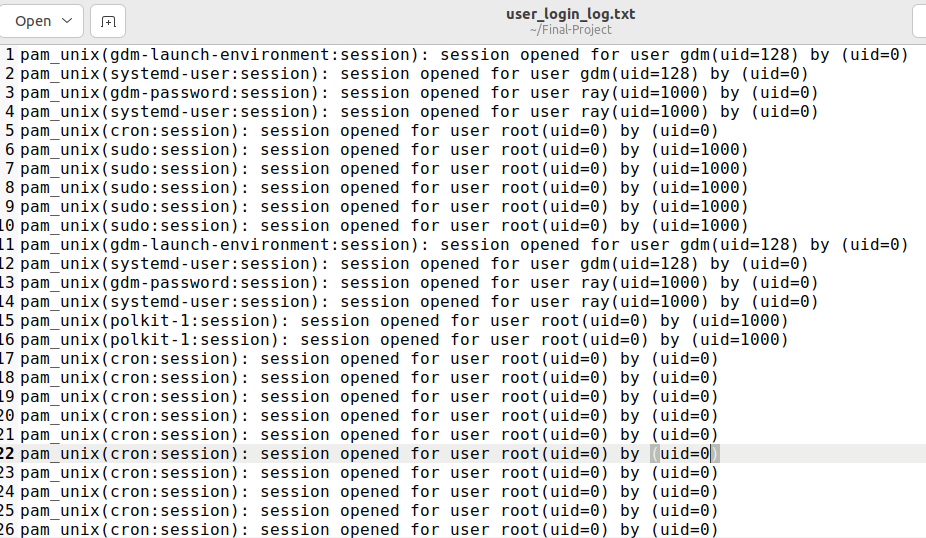


Figure 12. saved log file

## Count Users per Port

**Procedure for Counting Users on Specific Ports**

This part describes how the script counts the number of users accessing specific ports. It includes user input for specifying ports, validating the input, and using commands like `netstat` to count active connections on the specified ports.[3]

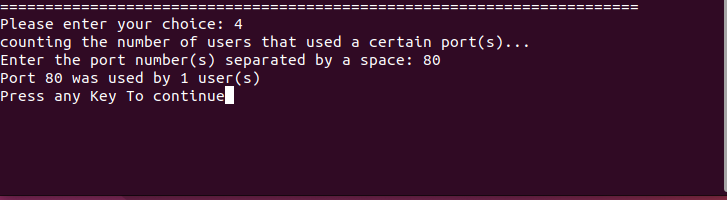


Figure 13. users per port

## Filter Traffic by IP Address

**Filtering Network Traffic Based on IP Address**

This section explains how to filter network traffic based on a specified IP address. It includes validating the IP address input, setting up the capture filter to include traffic to or from the IP address, and capturing the filtered traffic.[3]

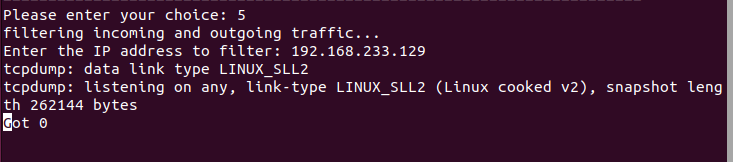


Figure 14. Capturing Packet for a specific IP

## Count Online Users within Time Range

**Steps to Count Online Users within a Specific Time Range**

This part provides a step-by-step explanation of counting online users within a specified time range. It includes validating the time input, using the `who` command to list logged-in users, and filtering the results based on the specified time range.

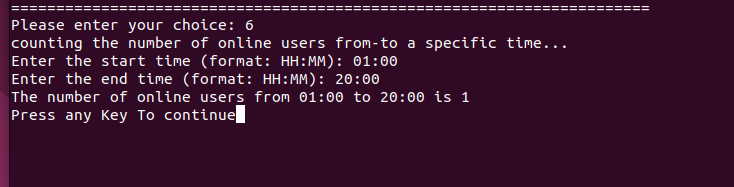


Figure 15. Number of online users

## Filter Specific IP Visiting Web Browser

**Filtering Traffic for Specific IP Accessing Web Services**

This section details how to filter network traffic for a specific IP address accessing web services (ports 80 and 443). It includes user input for the IP address, setting up the appropriate filter expression, and capturing the relevant traffic.

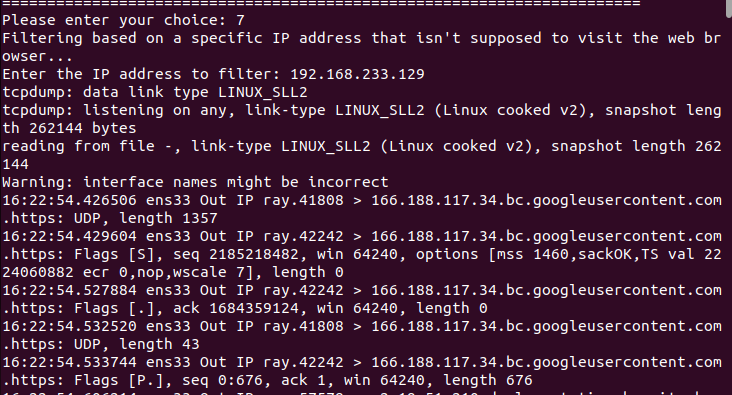


Figure 16. Filter for specific IP using browser

## Backup PCAP Files

**Procedure for Backing Up PCAP Files and Deletion Options**

This part explains how the script backs up PCAP files by compressing them into an archive. It also covers the options provided to the user for deleting the original PCAP files, either interactively or in bulk.

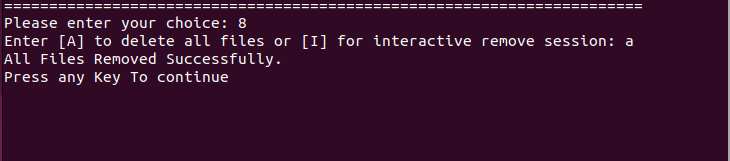


Figure 17. Backup and delete old file

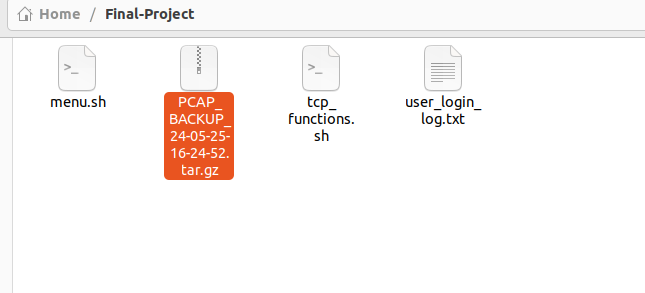


Figure 18. The backup file

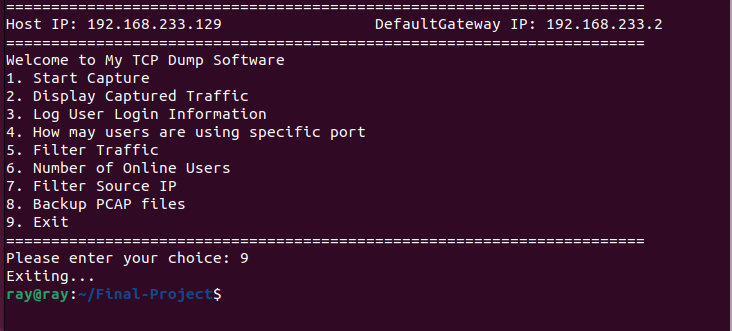


Figure 18. Exiting

# Problem Faced and Solutions

In this project, several problems were encountered and resolved to improve functionality and user experience. Initially, there was an issue with the inability to display the capture process and return to the main menu, which was addressed by using the trap command to catch interrupt signals and ensure a smooth transition back to the menu. Another challenge was the script overwriting capture files by saving packets into the same file. This was solved by incorporating timestamps into file names, thereby creating unique files for each capture session based on the date and time.

Additionally, the script lacked proper error handling, which could lead to unexpected terminations and user frustration. To address this, comprehensive error checking conditions were implemented to handle various failure scenarios gracefully. These enhancements collectively improved the robustness of the script, ensuring it could handle user inputs and system errors more effectively, thus providing a more reliable and user-friendly experience.

# Conclusion

All in all, through this project we achieved a user-friendly network monitoring tool built upon the powerful capabilities of TCPDUMP. The tool transcends the limitations of simply capturing traffic by offering a comprehensive suite of functionalities. Users can initiate targeted captures, filter based on specific criteria, get user login information, and even pinpoint the number of users on particular ports – all through a convenient menu-driven interface.

Furthermore, this project included other topics we mentioned in class to make it more interesting. The implementation of the trap command ensures a seamless return to the main menu even when capture processes are interrupted. Timestamped filenames eliminate the risk of overwriting valuable data, while robust error handling safeguards against unexpected script termination. These features collectively contribute to a user experience that is both intuitive and reliable.

By leveraging TCPDUMP's core functionalities and extending them with a user-centric approach, this project empowers network administrators and security professionals with a valuable tool for network troubleshooting and analysis. It allows them to gain granular insights into network traffic patterns, identify potential security vulnerabilities, and maintain a healthy and secure network environment.

# References

[1] The TCPDUMP Group. "TCPDUMP & LIBPCAP." Internet: https://www.tcpdump.org, [2024-05-25].

[2] Linux Man Pages. "tcpdump - Linux man page." Internet: http://man7.org/linux/man-pages/man1/tcpdump.1.html, [2024-05-25].

[3] Linux Man Pages. "journalctl - Linux man page." Internet: http://man7.org/linux/man-pages/man1/journalctl.1.html, [2024-05-25].

[4] Linux Man Pages. "netstat - Linux man page." Internet: http://man7.org/linux/man-pages/man8/netstat.8.html, [2024-05-25].

# Appendix A

**tcp\_functions.sh**

#!/bin/bash

# function to capture network traffic using tcpdump

capture\_traffic() {

trap 'echo "Capture interrupted. Returning to main menu..."; return' SIGINT

DATE=$(date +%y-%m-%d-%h:%m:%s)

echo "starting network capture..."

sudo tcpdump -w - | tee captured\_traffic\_$DATE.pcap | tcpdump -r -

if ! [[ $? -eq 0 ]]; then

echo "error:$0:$LINENO: something went wrong."

exit 1

fi

trap - SIGINT

return 0

}

# function to display captured traffic

display\_traffic() {

trap 'echo "Display interrupted. Returning to main menu..."; return' SIGINT

clear

ls -1 \*.pcap | awk '{printf "%d- %s\n", NR, $1}'

read -p "Enter the number of the file to display: " FILENAME

SELECTED\_FILE=$(ls -1 \*.pcap | awk -v num=$FILENAME 'NR==num {print $1}')

if [[ -z $SELECTED\_FILE ]]; then

echo "Invalid file number. Please try again."

return 1

fi

echo "Displaying captured traffic..."

sudo tcpdump -r $SELECTED\_FILE 2> /dev/null

if [[ $? -gt 1 ]]; then

echo "Error: $0:$LINENO: Something went wrong."

return 1

fi

trap - SIGINT

return 0

}

# Function to log user login information

log\_user\_login() {

echo "Logging user login information..."

LOG\_FILE="user\_login\_log.txt"

# Check if journalctl is available

if ! command -v journalctl &> /dev/null; then

echo "Error: journalctl is not available on this system."

exit 1

fi

# Log user login information

if [[ -f $LOG\_FILE ]]; then

sudo journalctl --since "yesterday" --output cat | grep -i 'session opened' >> $LOG\_FILE

else

sudo journalctl --since "yesterday" --output cat | grep -i 'session opened' > $LOG\_FILE

fi

if ! [[ $? -eq 0 ]]; then

echo "Error:$0:$LINENO: Something went wrong."

exit 1

fi

}

# The number of users that used a certain port(s), for example 80 and 8080

users\_per\_port() {

echo "counting the number of users that used a certain port(s)..."

read -p "Enter the port number(s) separated by a space: " PORTS

PORTS\_PATTERN='^[0-9]{1,5}$'

for PORT in $PORTS; do

if ! [[ $PORT =~ $PORTS\_PATTERN ]]; then

echo "Error:$0:$LINENO: Invalid PORT NUMBER"

exit 1

fi

echo "Port $PORT was used by $(sudo netstat -tuln | grep $PORT | wc -l) user(s)"

done

}

# filter incoming and outgoing traffic

filter\_traffic() {

echo "filtering incoming and outgoing traffic..."

read -p "Enter the IP address to filter: " IP

IP\_PATTERN='^([0-9]{1,3}\.){3}([0-9]{1,3})$'

if ! [[ $IP =~ $IP\_PATTERN ]]; then

echo "Error:$0:$LINENO: Invalid IP"

exit 1

fi

DATE=$(date +%y-%m-%d-%h:%m:%s)

trap 'echo "Display interrupted. Returning to main menu..."; return' SIGINT

sudo tcpdump -i any -vv -w - src $IP or dst $IP | tee $IP-CapturedPackets-$DATE.pcap | tcpdump -r -

if ! [[ $? -eq 0 ]]; then

echo "error:$0:$LINENO: something went wrong."

exit 1

fi

trap - SIGINT

return 0

}

# the number of online users from-to a specific time

online\_users() {

echo "counting the number of online users from-to a specific time..."

INPUT\_PATTERN='^[0-9]{1,2}:[0-9]{1,2}$'

read -p "Enter the start time (format: HH:MM): " START\_TIME

if ! [[ $START\_TIME =~ $INPUT\_PATTERN ]]; then

echo "Error:$0:$LINENO: Invalid Input"

exit 1

fi

read -p "Enter the end time (format: HH:MM): " END\_TIME

if ! [[ $END\_TIME =~ $INPUT\_PATTERN ]]; then

echo "Error:$0:$LINENO: Invalid Input"

exit 1

fi

echo "The number of online users from $START\_TIME to $END\_TIME is $(who | awk -v start=$START\_TIME -v end=$END\_TIME '$4 >= start && $4 <= end' | wc -l)"

}

# Filter based on a specific ip address that isn't supposed to visit the web browser

filter\_ip() {

echo "Filtering based on a specific IP address that isn't supposed to visit the web browser..."

read -p "Enter the IP address to filter: " IP

IP\_PATTERN='^([0-9]{1,3}\.){3}[0-9]{1,3}$'

if ! [[ $IP =~ $IP\_PATTERN ]]; then

echo "Error:$0:$LINENO: Invalid IP address"

exit 1

fi

DATE=$(date +%y-%m-%d-%h:%m:%s)

trap 'echo "Capture interrupted. Returning to main menu..."; return' SIGINT

sudo tcpdump -i any -vv -w - "src $IP and (dst port 80 or dst port 443)" | tee $IP-SpecificCapture-$DATE.pcap| tcpdump -r -

if ! [[ $? -eq 0 ]]; then

echo "Error:$0:$LINENO: Something went wrong."

exit 1

fi

trap - SIGINT

return 0

}

backup\_pcap\_files() {

DATE=$(date +%y-%m-%d-%H-%M-%S)

BACKUP\_FILE="PCAP\_BACKUP\_$DATE.tar.gz"

tar -czvf "$BACKUP\_FILE" \*.pcap &> /dev/null

if [[ $? -ne 0 ]]; then

echo "Error: $0:$LINENO: Something went wrong"

exit 1

fi

read -rp "Enter [A] to delete all files or [I] for interactive remove session: " OPT

if [[ $OPT == [Aa] ]]; then

rm -f \*.pcap

if [[ $? -eq 0 ]]; then

echo "All Files Removed Successfully."

else

echo "Error: $0:$LINENO: Something went wrong."

exit 1

fi

elif [[ $OPT == [Ii] ]]; then

rm -i \*.pcap

else

echo "Invalid Option. Try again"

fi

}

**menu.sh**

#!/bin/bash

set -uo pipefail

# Import the TCPdump Functions library

source "/home/$USER/Final-Project/tcp\_functions.sh"

# Main menu function

main\_menu() {

HOST\_IP=$(hostname -I)

DefaultGateway\_IP=$(route -n | grep 'UG[ \t]' | awk '{print $2}')

echo "======================================================================="

printf "Host IP: %s\t\t DefaultGateway IP: %s\n" $HOST\_IP $DefaultGateway\_IP

echo "======================================================================="

echo "Welcome to My TCP Dump Software"

echo "1. Start Capture"

echo "2. Display Captured Traffic"

echo "3. Log User Login Information"

echo "4. How may users are using specific port"

echo "5. Filter Traffic"

echo "6. Number of Online Users"

echo "7. Filter Source IP"

echo "8. Backup PCAP files"

echo "9. Exit"

echo "======================================================================="

read -p "Please enter your choice: " choice

case $choice in

1) capture\_traffic ;;

2) display\_traffic ;;

3) log\_user\_login ;;

4) users\_per\_port ;;

5) filter\_traffic ;;

6) online\_users ;;

7) filter\_ip ;;

8) backup\_pcap\_files ;;

9) echo "Exiting..."; exit 0 ;;

\*) echo "Invalid choice. Please enter a valid option." ;;

esac

}

# Main function

main() {

while true; do

main\_menu

read -p "Press any Key To continue"

clear

done

}

# Call the main function

main