**Qatar University**

**College of Engineering**

**CMPS 405: Operating Systems**

**Fall 2024**

**(Section B04)**

**Project Phase-1**

***Submitted to:***

Dr. Heba Dawoud

***Github repo link***: <https://github.com/nafin-nm1913152/CMPS405_Project_Grp7>

|  |  |  |
| --- | --- | --- |
| **Student** | **Name** | **ID** |
| **#1** | Nafin Mahmoud | 201913152 |
| **#2** | Khamis Al-Kubaisi | 201707893 |
| **#3** | Khalid Abdallah | 202106376 |
| **#4** | M.Reyad Tuama Halabi | 201902988 |

**Server-Side Scripts (VM1)**

* ***Server-Side Shell script 1: (network.sh)***

#!/bin/bash  
  
# Check if at least one IP is provided  
  
if [ "$#" -eq 0 ]; then  
    echo "No ip was provided"  
    exit 1  
fi  
  
# Check whether ping and traceroute are installed. Otherwise, install all network tools  
if ! command -v ping &> /dev/null || ! command -v traceroute &> /dev/null; then  
    echo "Installing ping and traceroute"  
    sudo apt-get update -y &> /dev/null  
    sudo apt-get install -y iputils-ping traceroute net-tools &> /dev/null  
fi    
  
# Creating log file for network  
log="network.log"  
echo "Testing Connection on $(date)" > $log  
  
# Checking connectivity  
check() {  
    # Taking one IP at a time  
    local target\_IP=$1  
    echo "Testing connection for $target\_IP"  
  
    # Test with ping  
    if ping -c 3 -W 5 "$target\_IP" &> /dev/null; then  
        # If good connection, then log result  
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Testing connection with $target\_IP is successful" | tee -a "$log"  
    else  
        # If ping fails, then call traceroute  
        echo "$(date '+%Y-%m-%d %H:%M:%S') - Testing connection with $target\_IP has failed, calling traceroute now" | tee -a "$log"  
        ./traceroute.sh "$target\_IP"  
    fi  
}  
  
# Check connection for each IP three times  
for i in {1..3}; do  
    for target\_IP in "$@"; do   
        check "$target\_IP"  
    done  
    echo "Pass no.$i is done"  
done

* ***Server-Side Shell script 1: (traceroute.sh)***

#!/bin/bash  
  
# Check if a target IP was provided  
  
if [ "$#" -eq 0 ]; then  
    echo "No ip was provided"  
    exit 1  
fi  
  
target\_IP=$1  
log="network.log"  
  
# Log the start of the traceroute process  
echo "Running traceroute for $target\_IP" | tee -a "$log"  
  
# Display routing table  
echo "Routing table:" | tee -a "$log"  
netstat -nr | tee -a "$log"  
  
# Display hostname  
echo "Hostname: $(hostname)" | tee -a "$log"  
  
# Testing local DNS server connection  
echo "Testing local DNS server connection" | tee -a "$log"  
nslookup [google.com](http://google.com) | tee -a "$log"  
  
# Tracing route to google  
echo "Tracing route to [google.com](http://google.com)" | tee -a "$log"  
traceroute [google.com](http://google.com) | tee -a "$log"  
  
# Tracing route to the target IP  
echo "Tracing route to $target\_IP" | tee -a "$log"  
traceroute "$target\_IP" | tee -a "$log"  
  
# Ping [google.com](http://google.com)  
echo "Pinging [google.com](http://google.com)" | tee -a "$log"  
ping -c 3 -W 5 [google.com](http://google.com) | tee -a "$log"   
  
# Check connectivity to target IP  
if ! ping -c 3 -W 5 "$target\_IP" &> /dev/null; then  
    echo "Cannot reach target $target\_IP, rebooting now" | tee -a "$log"  
    sudo reboot  
else  
    echo "Reconnected to $target\_IP" | tee -a "$log"  
fi

* ***Server-Side Shell script 2: (System.sh)***

#!/bin/bash

# System.sh script

# LOGs

DISK\_LOG="disk\_info.log"

SYS\_LOG="mem\_cpu\_info.log"

# Disk usage for HOME directory, including subdirectories

echo "Disk usage for HOME directory:" > "$DISK\_LOG"

du -h $HOME/\* >> "$DISK\_LOG"

# Total disk space for the file system containing the home directory

echo "Total Disk Space:" >> "$DISK\_LOG"

df -h $HOME | awk 'NR==2{print "Total: "$2", Used: "$3", Available: "$4", Usage: "$5}' >> "$DISK\_LOG"

# Memory and CPU information

echo "Memory and CPU Information:" > "$SYS\_LOG"

echo "---------------------------------" >> "$SYS\_LOG"

# Free and used memory as percentage

free -m | awk 'NR==2{printf "Memory Usage: Used: %.2f%%, Free: %.2f%%\n", $3\*100/$2, $4\*100/$2 }' >> "$SYS\_LOG"

# CPU model and cores

echo "CPU Model:" >> "$SYS\_LOG"

lscpu | grep 'Model name' | awk -F: '{print $2}' >> "$SYS\_LOG"

echo "CPU Cores:" $(nproc) >> "$SYS\_LOG"

# Save information to logs and display the output

cat "$DISK\_LOG"

cat "$SYS\_LOG"

**Client 1 (VM3)**

**Client 1 Side Shell Script 1: login.sh**

#!/bin/bash  
# Make sure sshpass is installed before proceeding  
if ! command -V sshpass &> /dev/null; then  
    echo "sshpass not found, installing..."  
    sudo apt update  
    sudo apt install sshpass -y  
fi  
# Initialize variables for tracking login attempts  
login\_attempts=0  
max\_login\_attempts=3  # maximum login attempts  
  
# Loop for login up to 3 times  
while [ $login\_attempts -lt $max\_login\_attempts ]; do  
    read -p "Enter Username: " user\_name  
    read -sp "Enter Password: " user\_pass  
    echo  
  
    # Attempt to log in (replace 'server\_ip' with actual IP)  
    ssh "$[user\_name@192.168.66.130](mailto:user_name@192.168.66.130)" echo "Login successful" > /dev/null 2>&1  
  
    # Check if attempt failed  
    if [ $? -ne 0 ]; then  
        # Log failed attempt with username and timestamp  
        echo "$user\_name $(date) - Invalid attempt" >> invalid\_attempts.log  
        ((login\_attempts++))  
        echo "Attempt $login\_attempts of $max\_login\_attempts failed."  
    else  
        # Successful login  
        echo "Login successful"  
        exit 0  
    fi  
done  
  
# show unauthorized message and handle logout after 3 failed attempts  
echo "Unauthorized user"  
sftp "$user\_name@192.168.66.130:client\_timestamp\_invalid\_attempts.log"  
sleep 30 && pkill -KILL -u "$user\_name"  
# After 3 failed attempts, show unauthorized message and handle logout  
echo "Unauthorized user!"  
sftp "$username@192.168.66.130:client\_timestamp\_invalid\_attempts.log"  
sleep 30 && pkill -KILL -u "$username"

**Client 1 Side Shell Script 2: Check.sh**

#!/bin/bash  
  
# Locate the files with 777 permissions and save to a temporary log  
find / -type f -perm 777 > permission\_log.log 2>/dev/null  
  
# Display and update permissions for each file found  
while read -r file\_entry; do  
    echo "File with 777 permissions: $file\_entry"  # Display file with 777 permission  
    chmod 700 "$file\_entry"  # Change permissions to 700  
done < permission\_log.log  
  
echo "Permission changes complete for files with 777 permissions."

**Client 2 (VM3)**

**Client 2 Side Shell script 1: (Search.sh)**

*#!/bin/bash*

search\_files() {

*# Finding the files that are larger than 1 Megabyte in client 2 account:*

*# ((2> /dev/null) is used to redirect error messages to /dev/null to discard them):*

files=$(find ~ -type f -size +1M 2> /dev/null)

*# Piping (files) to (wc -l) to find the number of files larger than 1 Megabyte:*

number\_of\_files=$(echo "$files" | wc -l)

*# Search date:*

search\_date=$(date)

*# Storing the results in (bigfile.log):*

*# (First, we will overwrite the file then we will append the number of files and search date):*

echo -e "$files\n" > bigfile.log

echo -e "Number of files: $number\_of\_files\n" >> bigfile.log

echo -e "Search date: $search\_date" >> bigfile.log

*# Checking if (bigfile.log) is not empty:*

*if* [ -s "bigfile.log" ]; *then*

*# Sending an email using the (curl) command:*

*# Note: 2FA has to be enabled in the email account and an app password must be used:*

curl -s --url 'smtps://smtp.gmail.com:465' --ssl-reqd \

--mail-from "mriyadh1500@gmail.com" \

--mail-rcpt "mriyadh1500@gmail.com"\

--user "mriyadh1500@gmail.com":omgbkpfxhqxknfle \

-T <(echo -e "$(cat "bigfile.log")")

*fi*

}

search\_files

**Client 2 Side Script Shell script 2: (Clientinfo.sh)**

*#!/bin/bash*

collect\_process\_info() {

*# Finding the process tree of all currently running processes:*

ps\_tree=$(pstree)

*# Finding the list of all zombie processes:*

zombie\_list=$(ps aux | awk '$8=="Z"')

*# Sometimes, there is no zombie processes at all:*

*if* [ -z "$zombie\_list" ]; *then*

zombie\_list="No zombie processes found."

*fi*

*# Finding processes CPU usage:*

cpu\_usage\_list=$(ps -eo pid,%cpu,comm)

*# Finding processes memory usage:*

memory\_usage\_list=$(ps -eo pid,%mem,comm)

*# Finding the top cpu-consuming processes and piping the result to the (head) command:*

*# 6 is used instead of 5 to display the fields line at the begenning:*

top\_cpu\_usage=$(ps -eo pid,%cpu,comm --sort=-%cpu | head -n 6)

*# Finding the top memory-consuming processes and pinping the result to the (head) command:*

*# 6 is used to display the fields line at the begenning:*

top\_memory\_usage=$(ps -eo pid,%mem,comm --sort=-%mem | head -n 6)

*# Storing the results in (process\_info.log):*

*# (First, we will overwrite the file then we will append):*

echo -e "Process tree:\n$ps\_tree\n" > *$1*

echo -e "List of all zombie processes:\n $zombie\_list\n" >> *$1*

echo -e "Processes CPU usage:\n $cpu\_usage\_list\n" >> *$1*

echo -e "Processes memory usage:\n $memory\_usage\_list\n" >> *$1*

echo -e "Top 5 cpu-consuming processes:\n $top\_cpu\_usage\n" >> *$1*

echo -e "Top 5 memory-consuming processes:\n $top\_memory\_usage\n" >> *$1*

}

*while* true

*do*

*# Calling the function and passing the name of the file where process information will be saved:*

collect\_process\_info process\_info.log

*# To connect to the server, we need two pieces of information: 1- IP address 2- Username:*

*# Getting the IP address of the server:*

echo "Enter the IP address of the server: "

read ip\_address

*# Getting the username of the account that we want to connect to in the server:*

echo "Enter the username: "

read username

*# Securely copying (process\_info.log) to the server:*

*# (scp already uses SSH for transferring files):*

sudo scp process\_info.log "$username@$ip\_address:/home/client\_02"

*# Making sure that (process\_info.log) is stored in the server every hour:*

sleep 3600

*done*