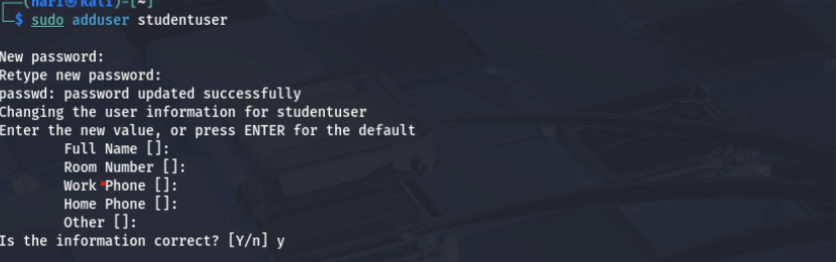
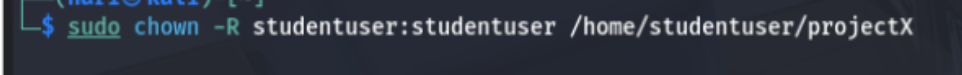
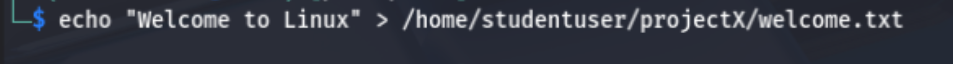
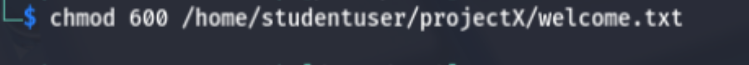
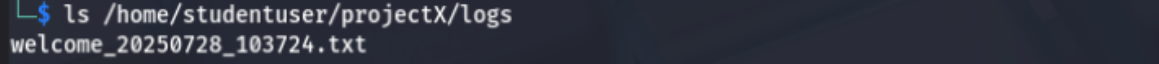
Linux & Networking POC

## Task 1: Linux Essentials & File Permissions

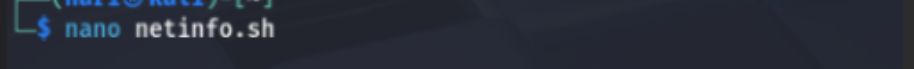
* Create a new user called 'studentuser'.  
  ****
* Create the following directory structure:
* /home/studentuser/projectX/logs  
   /home/studentuser/projectX/scripts  
  **sudo mkdir -p /home/studentuser/project/{logs,scripts}  
  sudo chrown -R studentuser:studentuser /home/studentuser/project  
  sudo -u studentuser bash**
* Create a file 'welcome.txt' in 'projectX' with the content 'Welcome to Linux'.  
  **echo “Welcome to linux” > /home/studentuser/project/welcome.txt**
* Set permissions so only 'studentuser' can read/write the file.  
  **chmod 600 /home/studentuser/project/welcome.txt**
* Create a script 'backup.sh' in 'scripts' that copies 'welcome.txt' to 'logs' with a timestamp.  
  **nano /home/studentuser/project/scripts/backup.sh**

**#!/bin/bash  
src=”/home/studentuser/projectX/welcome.txt”**

**Dest=”/home/studentuser/project/logs/welcome\_$(date+%Y%m%d\_%H%M%S).txt”**

**Cp “$src” “$dest”**

## Task 2: Networking Toolkit PoC

* Instructions:
* Write a script 'netinfo.sh' that:  
  **nano netinfo.sh  
    
    
    
  cat << 'EOF' > /home/studentuser/netinfo.sh**

**#!/bin/bash**

**OUTPUT="/home/studentuser/network\_report.txt**

**{**

**echo "===== IP & Network Info ====="**

**ip -br addr # Short-form IP display**

**echo ""**

**echo "===== Default Gateway ====="**

**ip route | grep default**

**echo ""**

**echo "===== Open Ports (TCP/UDP) ====="**

**ss -tuln**

**echo ""**

**echo "===== Ping google.com ====="**

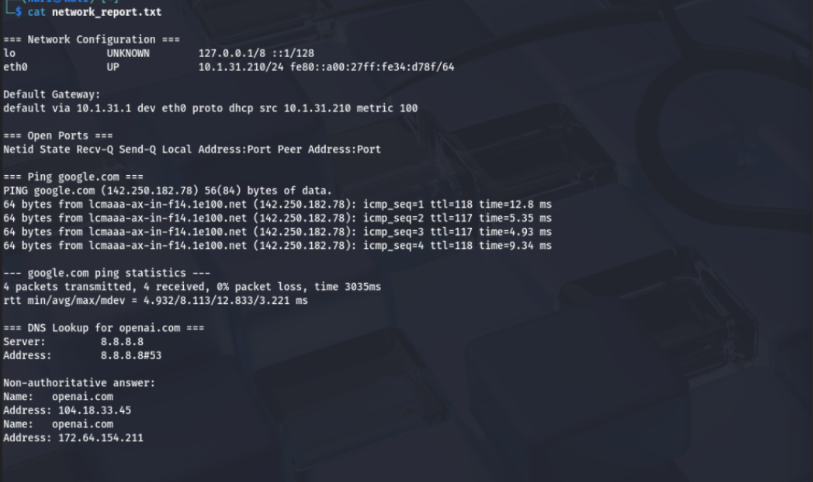
**ping -c 4 google.com**

**echo ""**

**echo "===== DNS Lookup for openai.com ====="**

**nslookup openai.com**

**} > "$OUTPUT"**

* - Displays IP address, subnet mask, and default gateway.`1  
  
* - Lists open ports using 'ss' or 'netstat'.
* - Pings google.com and logs the response.
* - Resolves the IP address of 'openai.com'.
* Store the output in 'network\_report.txt'. 

1. 'echo "\nPing google.com:"' \
2. 'ping -c 4 google.com' \
3. 'echo "\nDNS Lookup for openai.com:"' \
4. 'nslookup openai.com' > network\_report.txt' | tee netinfo.sh
5. chmod +x netinfo.sh
6. ./netinfo.sh

## Task 3: Mini Server Monitor Script

* Instructions:

Create a script 'monitor.sh' that:   
  
 Open the script for editing   
**nano /home/studentuser/monitor.sh  
code inside nano:**

**#!/bin/bash**

**LOG="/var/log/monitor.log"**

**timestamp=$(date "+%Y-%m-%d %H:%M:%S")**

**echo "[$timestamp] ===== Starting System Check =====" >> "$LOG"**

**# Check if nginx is running**

**if ! pgrep nginx > /dev/null; then**

**echo "[$timestamp] nginx is NOT running. Starting nginx..." >> "$LOG"**

**sudo systemctl start nginx**

**else**

**echo "[$timestamp] nginx is running." >> "$LOG"**

**# Show memory usage**

**echo "[$timestamp] Memory Usage:" >> "$LOG"**

**free -h >> "$LOG"**

**# Show CPU load**

**echo "[$timestamp] CPU Load:" >> "$LOG"**

**uptime >> "$LOG"**

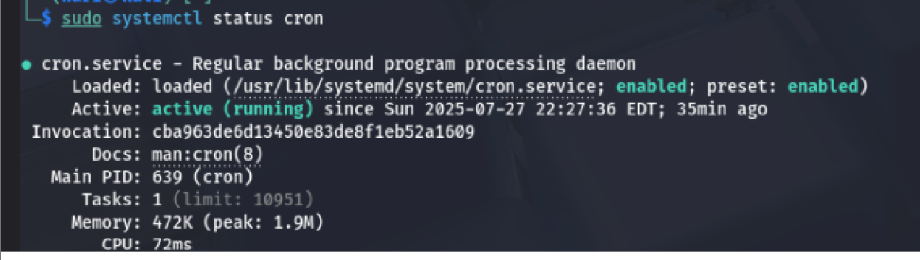
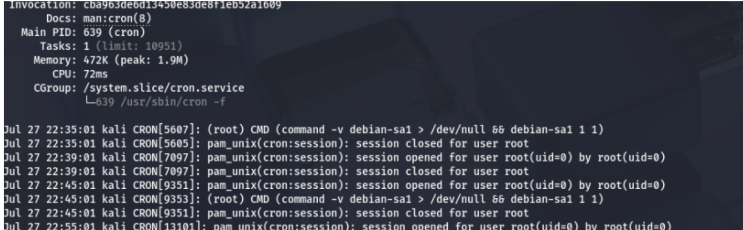
**# Show disk usage**

**echo "[$timestamp] Disk Usage:" >> "$LOG"**

**df -h >> "$LOG"**

**echo "[$timestamp] ===== System Check Complete =====" >> "$LOG"**

**echo "" >> "$LOG"**

* - Checks if 'nginx' is running and starts it if not. 
* - Displays memory usage, CPU load, and disk usage. 
* - Logs all activities with timestamps. 
* Schedule it via cron to run every 5 minutes. 

## Task 4: File Watcher Script

## Step 1: Install inotify-tools (if not already installed) sudo apt install inotify-tools -y

## Step 2: Create the script #!/bin/bash

## WATCH\_DIR="/home/studentuser/projectX/logs"

## LOG\_FILE="/home/studentuser/log\_monitor.txt"

## echo "Watching directory: $WATCH\_DIR"

## inotifywait -m -e create --format '%f' "$WATCH\_DIR" | while read FILENAME

## do

## if [[ "$FILENAME" == \*.txt ]]; then

## TIMESTAMP=$(date "+%Y-%m-%d %H:%M:%S")

## echo "[$TIMESTAMP] New file detected: $FILENAME" >> "$LOG\_FILE"

## Step 3: Make it executable chmod +x /home/studentuser/watch\_dir.sh Step 4: Run the script in background (or a separate terminal) /home/studentuser/watch\_dir.sh Step 5: Test it(another terminal) touch /home/studentuser/projectX/logs/test1.txt cat /home/studentuser/log\_monitor.txt Task 5: SSH Login Audit

Step 1: Create the script  
**nano /home/studentuser/ssh\_audit.sh  
** THE SCRIPT  
  
**#!/bin/bash**

**OUTPUT="/home/studentuser/ssh\_audit.txt"**

**timestamp=$(date "+%Y-%m-%d %H:%M:%S")**

**echo "[$timestamp] SSH LOGIN AUDIT" > "$OUTPUT"**

**# Last 5 successful logins**

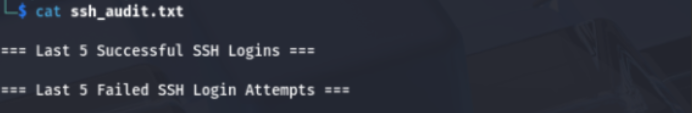
**echo -e "\nLast 5 Successful SSH Logins:" >> "$OUTPUT"**

**last -i | grep 'sshd' | head -n 5 >> "$OUTPUT"**

**# Last 5 failed login attempts**

**echo -e "\nLast 5 Failed SSH Attempts:" >> "$OUTPUT"**

**grep "Failed password" /var/log/auth.log | tail -n 5 >> "$OUTPUT"**

Step 2: Make the script executable  
**chmod +x /home/studentuser/ssh\_audit.sh**  
Step 3: Run the script  
/**home/studentuser/ssh\_audit.sh**  
step 4: View the results  
**cat /home/studentuser/ssh\_audit.txt**

## Task 6: Crontab Practice

Open crontab for the current user:  
 **crontab -e**  
Cron Job 1: Print "Good morning!" at 8 AM daily

**0 8 \* \* \* echo "Good morning!" >> /home/studentuser/greetings.log**

Cron Job 2: Backup projectX every Sunday  
 **0 3 \* \* 0 tar -czf /home/studentuser/projectX\_backup\_$(date +\%Y\%m\%d).tar.gz /home/studentuser/projectX/**

Cron Job 3: Delete .log files older than 7 days every Friday at midnight

**0 0 \* \* 5 find /home/studentuser/projectX/ -type f -name "\*.log" -mtime +7 -exec rm {} \;**

## Task 7: Port Scanner Script

Step 1: Create the script  
**nano /home/studentuser/port\_scanner.sh**  
THE CODE IN THE SCRIPT  
**#!/bin/bash**

**# Ask the user for an IP address**

**read -p "Enter the IP address to scan: " ip**

**echo "Scanning ports 20 to 25 on $ip..."**

**echo "Scan started at: $(date)"**

**echo**

**# Loop through ports 20 to 25**

**for port in {20..25}**

**do**

**timeout 1 bash -c "echo > /dev/tcp/$ip/$port" 2>/dev/null && \**

**echo "Port $port is OPEN" || \**

**echo "Port $port is CLOSED"**

**done**

**echo**

**echo "Scan completed at: $(date)"**

Step 2: Make it executable  
**chmod +x /home/studentuser/port\_scanner.sh**

Step 3: Run the script  
**/home/studentuser/port\_scanner.sh**

## Task 8: Website Availability Checker

## Create sites.txt nano /home/studentuser/sites.txt SAMPLES https://openai.com

## https://example.com

## https://thiswebsitedoesnotexist.tld

## Create the Script: check\_sites.sh

## nano /home/studentuser/check\_sites.sh SCRIPT #!/bin/bash

## INPUT="/home/studentuser/sites.txt"

## LOG="/home/studentuser/site\_status.log"

## echo "Website Status Check - $(date)" >> "$LOG"

## while read -r site; do

## if curl -s --head --request GET "$site" | grep "200 OK" > /dev/null; then

## echo "$site is UP at $(date)" >> "$LOG"

## else

## echo "$site is DOWN or unreachable at $(date)" >> "$LOG"

## fi

## done < "$INPUT"

## echo "Check complete." >> "$LOG"

## echo "Results saved to $LOG"

## Make the Script Executable

## chmod +x /home/studentuser/check\_sites.sh

## 4. Run the Script /home/studentuser/check\_sites.sh

## Task 9: Environment and Disk Report

Step 1: Create the script  
**nano /home/studentuser/env\_report.sh**SCRIPT  
 **#!/bin/bash**

**REPORT="/home/studentuser/environment\_report.txt"**

**TIMESTAMP=$(date)**

**echo "===== ENVIRONMENT AND DISK REPORT =====" > "$REPORT"**

**echo "Generated at: $TIMESTAMP" >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "1. Current User:" >> "$REPORT"**

**whoami >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "2. Hostname:" >> "$REPORT"**

**hostname >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "3. System Uptime:" >> "$REPORT"**

**uptime >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "4. Mounted Filesystems:" >> "$REPORT"**

**mount | grep "^/" >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "5. Disk Usage:" >> "$REPORT"**

**df -h >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "6. Environment Variables:" >> "$REPORT"**

**echo "PATH = $PATH" >> "$REPORT"**

**echo "SHELL = $SHELL" >> "$REPORT"**

**echo "" >> "$REPORT"**

**echo "Environment report saved to $REPORT"**

Step 2: Make it executable  
**chmod +x /home/studentuser/env\_report.sh**

Step 3: Run it  
**/home/studentuser/env\_report.sh**

## Task 10: Compress & Archive Automation

1. Create the Script  
**nano /home/studentuser/archive\_logs.sh**  
SCRIPT  
**#!/bin/bash**

**# Set variables**

**LOG\_DIR="/home/studentuser/projectX/logs"**

**BACKUP\_DIR="/home/studentuser/projectX/backup"**

**DATE=$(date +%Y-%m-%d)**

**ARCHIVE\_NAME="archive\_$DATE.tar.gz"**

**TMP\_LIST="/tmp/logs\_to\_archive.txt"**

**# Create backup directory if it doesn't exist**

**mkdir -p "$BACKUP\_DIR"**

**# Find .log files larger than 10MB and store the list**

**find "$LOG\_DIR" -type f -name "\*.log" -size +10M > "$TMP\_LIST"**

**# Check if any files were found**

**if [ -s "$TMP\_LIST" ]; then**

**# Create a compressed archive with those files**

**tar -czf "$ARCHIVE\_NAME" -T "$TMP\_LIST"**

**# Move the archive to the backup directory**

**mv "$ARCHIVE\_NAME" "$BACKUP\_DIR"**

**echo "Archive $ARCHIVE\_NAME created and moved to $BACKUP\_DIR"**

**else**

**echo "No .log files larger than 10MB found in $LOG\_DIR"**

**fi**

**# Clean up temp file**

**rm -f "$TMP\_LIST"**

2. Make the Script Executable

**chmod +x /home/studentuser/archive\_logs.sh**

1. Run the Script

**/home/studentuser/archive\_logs.sh**