CYBER SECURITY

A Project Report on Automating System Management Tasks

Task 1: **Automate Log File Management**

Introduction

This project focuses on automating key system management tasks using Bash scripting in a Linux environment. The tasks include log file management, system health monitoring, and software installation automation. The deliverables demonstrate practical applications of Bash scripting to improve system administration efficiency and reliability.

Script Overview

The Bash script performs the following steps:

1. Configures directories for log files and backups.
2. Finds log files exceeding a specified size.
3. Compresses these files into a timestamped archive.
4. Moves the archive to a backup directory.
5. Deletes the original log files.

Scripting Code:

```
root@kali: /home/kali/Project/P1
File  Actions  Edit  View  Help
  GNU nano 8.2                                    New Buffer *
#!/bin/bash

# Configuration
LOG_DIR="/var/log"
ARCHIVE_DIR="/backup/logs"
MAX_SIZE_MB=100
BACKUP_FILE="log_backup_$(date +%Y%m%d%H%M%S).tar.gz"

# Ensure backup directory exists
mkdir -p $ARCHIVE_DIR

# Find log files larger than specified size and compress them
find $LOG_DIR -type f -size +$((MAX_SIZE_MB * 1024))c -exec tar -czvf $ARCHIVE_DIR/$BACKUP_FILE {} +

# Remove original log files
find $LOG_DIR -type f -size +$((MAX_SIZE_MB * 1024))c -exec rm {} +

echo "Backup created: $ARCHIVE_DIR/$BACKUP_FILE"
echo "Original log files deleted."

Save modified buffer?
 Y Yes
 N No          ^C Cancel
```

Save the code as: **log_management.sh**

And use the command to run the script

```
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x log_management.sh
./log_management.sh
```

Output:

```
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x log_management.sh
./log_management.sh
tar: Removing leading `/' from member names
/var/log/journal/30e662c5c81d4191bd2444a79c97d2e0/system.journal
tar: Removing leading `/' from hard link targets
/var/log/journal/30e662c5c81d4191bd2444a79c97d2e0/user-1000.journal
Backup created: /backup/logs/log_backup_20241228111442.tar.gz
Original log files deleted.
```

Task 2: Create a System Health Monitoring Script

Objective

To monitor CPU and memory usage in real-time and log the data every minute for a specified duration.

Script Overview

The script:

1. Collects CPU and memory usage using the top command.
2. Logs the data to a file at one-minute intervals for a given duration.
3. Ensures logs are timestamped for easy analysis.

Scripting Code:

```
GNU nano 8.2                    system_health_monitor.sh
#!/bin/bash

# Configuration
LOG_FILE="/var/log/system_health.log"
MONITOR_DURATION=60  # in seconds
INTERVAL=60          # in seconds

# Function to log system health
log_system_health() {
    CPU_USAGE=$(top -bn1 | grep "Cpu(s)" | sed "s/.*, *\([0-9.]*\)%* id.*/\1/" | awk '{print 100 - $1}')
    MEMORY_USAGE=$(free | grep Mem | awk '{print $3/$2 * 100.0}')
    TIMESTAMP=$(date +"%Y-%m-%d %H:%M:%S")
    echo "$TIMESTAMP - CPU Usage: $CPU_USAGE%, Memory Usage: $MEMORY_USAGE%" >> $LOG_FILE
}

# Monitor system health for the specified duration
for ((i=0; i<$MONITOR_DURATION; i+=INTERVAL)); do
    log_system_health
    sleep $INTERVAL
done

echo "System health monitoring completed. Logs saved in $LOG_FILE."

[ Read 22 lines ]
^G Help  ^O Write Out  ^F Where Is  ^K Cut  ^T Execute  ^C Location  M-U Undo  M-A Set Mark  M-] To Bracket  M-B Previous
^X Exit  ^R Read File  ^\ Replace  ^U Paste  ^J Justify  ^/ Go To Line  M-E Redo  M-6 Copy  ^B Where Was  M-F Next
```

Save the code as: system_health_monitor.sh

And use the command to run the script

```
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x system_health_monitor.sh
./system_health_monitor.sh
```

Output:

```
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x system_health_monitor.sh
./system_health_monitor.sh
System health monitoring completed. Logs saved in /var/log/system_health.log.
```

Real-World Application

System administrators use health monitoring scripts to detect performance issues, identify resource bottlenecks, and ensure system stability. The data collected helps in troubleshooting and capacity planning.

Task 3: Automate Software Installation

Objective

To read a list of software packages from a text file and install them using the system's package manager.

Script Overview

The script:

1. Reads package names from a text file.
2. Installs each package using the system's package manager (e.g., apt).
3. Logs successful and failed installations.

Real-World Application

This script is invaluable for provisioning servers or setting up development environments. Automating software installation reduces human error and ensures consistency across systems.

Scripting Code:

```
root@kali: /home/kali/Proje
File  Actions  Edit  View  Help
  GNU nano 8.2
#!/bin/bash

# Configuration
PACKAGE_LIST="/path/to/package_list.txt"

# Function to install packages
install_packages() {
    while IFS= read -r package; do
        echo "Installing $package ..."
        sudo apt-get update
        sudo apt-get install -y $package
    done < "$PACKAGE_LIST"
}

# Main script
install_packages
echo "Software installation completed."

^G Help        ^O Write Out   ^F Where Is    ^K Cut
^X Exit        ^R Read File   ^\ Replace     ^U Paste
```

Save the code as: **software_install.sh**

And use the command to run the script

```
File  Actions  Edit  View  Help
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x software_install.sh
./software_install.sh
```

Output:

```
root@kali: /home/kali/Project/P1
File  Actions  Edit  View  Help
┌──(root㉿kali)-[/home/kali/Project/P1]
└─# chmod +x software_install.sh
./software_install.sh
./software_install.sh: line 7: /path/to/package_list.txt: No such file or directory
Software installation completed.

┌──(root㉿kali)-[/home/kali/Project/P1]
└─#
```

Summary of Scripts

Functionality

- **Log Management**: Automates log file cleanup to free up disk space.
- **System Monitoring**: Provides real-time insights into system performance.
- **Software Installation**: Simplifies and automates the deployment of required software packages.

Applications

These scripts address common system administration tasks, making them efficient and error-free. They are particularly useful in environments requiring frequent log maintenance, health monitoring, or rapid server provisioning