Assignment Report - Directory Monitoring using Bash Script Report

Course: Ethical Hacking

Name: Karthik Rajeev

Roll no: 2462096

Index:

Introduction

File system monitoring is a critical process for **security assurance, auditing, and operational stability**. Changes to files in sensitive directories may signal **malware activity, unauthorized access, or configuration tampering**. This project focuses on developing a **Bash-based directory monitoring tool** for tracking activities in /home/student/Downloads. The script captures **file creation, deletion, modification, and attribute changes** in real time, logging them with timestamps for later review.

Aim of the Project

- Develop a script to **automatically detect and record** file system events.
- Provide **clear, timestamped logs** of directory changes.
- Enhance **security monitoring** and **incident response** capabilities.
- Maintain a **lightweight, easily deployable** solution.

Tools & Technologies Utilized:

- **Bash Scripting** – Core implementation.
- **inotifywait** (inotify-tools package) – Real-time event detection.
- **Linux Environment** – Execution platform.
- **Log Files** – Event storage and auditing.
- **Nano / Vim** – Script editing.

Implementation Methodology

Step 1: Install Required Tool

Install inotify-tools package for real-time file system monitoring:

```
sudo apt install inotify-tools
```

Step 2:

Step 2: Create the Bash Script

- Open a new script file:

```
nano directory_monitor.sh
```

Step 3: Write the Script

```
 #!/bin/bash

MONITOR_DIR="/home/student/Downloads"

LOG_FILE="monitor.log"

echo "Monitoring $MONITOR_DIR..."

inotifywait -m -e create -e delete -e modify -e attrib

"$MONITOR_DIR" --format '%T %e %f' --timefmt '%Y-%m-%d %H:%M:%S' |

while read date time event file; do

   echo "[$date $time] $event: $file" | tee -a "$LOG_FILE"

done
```

Step 4: Make the Script Executable

```
chmod +x directory_monitor.sh
```

Step 5: Run the Script

```
./directory_monitor.sh
```

Challenges Encountered

- Unavailability of `inotify-tools` by default on some systems.

- Permission errors when attempting to monitor restricted directories.

- Handling **rapid multiple file changes** without missing logs.

- Preventing log files from growing excessively during prolonged monitoring.

.

Sample Output

```
2025-08-12 21:10:22 CREATE sample.txt

2025-08-12 21:11:05 MODIFY project.docx

2025-08-12 21:12:48 DELETE old_notes.pdf
```

Learning Outcomes

- Hands-on experience with **real-time monitoring tools** in Linux.
- Understanding the **inotify API** and its applications.
- Improved skills in **shell scripting** and **automation**.
- Awareness of **security practices** for file system integrity.
- Ability to create **custom monitoring solutions** for different scenarios.

Conclusion:

This project demonstrates that even with **simple Bash scripting**, it is possible to create a **powerful, real-time directory monitoring system**. Using inotifywait, the solution remains **lightweight, accurate, and adaptable** to various monitoring needs. Such a system is beneficial for **security auditing, compliance checks, troubleshooting, and forensics** in both personal and professional environments.