RIPHAH
INTERNATIONAL
UNIVERSITY

Project Proposal: File Integrity Monitoring (FIM) System For Linux

Subject:
Operating System

Instructor:
Sir Yawar Abbas

Submitted By

Muhammad Husnain

SAP: 62056

Introduction

In today's digital environment, ensuring the integrity of critical files is essential for detecting unauthorized changes, potential security breaches, or system tampering. This project aims to develop a simple File Integrity Monitoring (FIM) system tailored for Linux environments. The system will monitor specified directories for changes and provide basic alerts, making it suitable for educational purposes or small-scale use.

The problem addressed is the need for a lightweight tool to track file modifications in real-time, which can help in basic intrusion detection or compliance auditing on personal or lab systems.

Objectives

- To create a basic FIM tool that detects file changes (additions, modifications, deletions) in selected directories.
- To implement real-time monitoring to avoid resource-intensive polling.
- To provide a simple web-based interface for viewing alerts and logs.
- To ensure the system is easy to set up and run on a Linux machine (e.g., Kali Linux).

Key Features

The system will include:

1. **Directory Monitoring**: Watch user-specified directories for file events like creation, modification, deletion, or moves.
2. **Event-Based Detection**: Use **pyinotify** for real-time notifications from the Linux kernel, avoiding full rescans.
3. **Baseline Management**: Create and maintain a simple JSON-based snapshot of file hashes (using SHA-256) to compare against changes.
4. **Alert Logging**: Log detected changes to a file and store them in memory for display.
5. **Web-Based Interface**: Alerts dashboard showing a table of changes (timestamp and message)
6. **Background Operation**: Run as a daemon process (e.g., via systemd) without blocking the terminal.

Methodology

1. **Setup**: User specifies directories via command-line arguments. The system loads or creates a baseline of file hashes.
2. **Initial Scan**: Perform a one-time parallel scan of files to establish or verify the baseline.
3. **Monitoring**: Use pyinotify to watch for events in a separate thread. On events, recompute hashes, detect changes, and log alerts.
4. **Web Interface**: Flask runs the server in the main thread, handling login and displaying alerts from memory.
5. **Deployment**: Run as a background service using systemd for persistence.

The system will handle basic error cases like permission issues but won't include advanced tampering protection in this minimal version.

Conclusion

This FIM system provides a practical introduction to system security tools, real-time monitoring, and web development in Python. It meets semester project requirements by being functional yet straightforward, with potential for future expansions like notifications or cross-platform support. The source code will be documented and shared via GitHub for review.