**Integrity Monitoring Project Report**

1. Title Page

Title: Integrity Monitoring

Author: [Arbaz khan]

Date: [03|Aug|2024]

2. Table of Contents

* Executive Summary
* Introduction
* Methodology
* Code Overview
* Logging Setup
* Configuration Loading
* Checksum Calculation
* Directory Scanning
* Trusted Hashes Loading
* Integrity Checking
* Command Line Interface (CLI)
* Usage Instructions
* Conclusion
* Recommendations
* References
* Appendices

3. Executive Summary

The Integrity Monitoring project is a Python script designed to check the integrity of system files. It does this by calculating file checksums and comparing them with trusted values stored in a database.

4. Introduction

The goal of this project is to monitor files and ensure they haven't been tampered with. It uses cryptographic hashes to detect changes, providing a simple way to check if files are still intact.

5. Methodology

The project uses different hashing algorithms (SHA-256, SHA-512, MD5) to calculate the checksums of files. These checksums are then compared with the trusted values to detect any discrepancies.

6. Code Overview

Logging Setup

The script uses the logging module to record information, warnings, and errors during execution.

* Configuration Loading

Settings like the trusted database file path and hashing algorithm are loaded from a configuration file.

* Checksum Calculation

The script calculates the checksum of each file using the specified algorithm. This helps in verifying if a file has been changed.

* Directory Scanning

It scans a given directory, calculates checksums for all files, and stores these values.

* Trusted Hashes Loading

The script loads a list of trusted file checksums from a database file.

* Integrity Checking

It compares the newly calculated checksums with the trusted ones. If there are differences, it logs them as discrepancies.

* Command Line Interface (CLI)

Users can run the script with different options using the command line, such as specifying directories, algorithms, and trusted database files.

7. Usage Instructions

Command Examples

Display Help Message:

* Copy code

python integrity\_checker.py -h

Scan a Directory and Save Checksums:

* Copy code

python integrity\_checker.py -d /path/to/directory -o output.json

Check File Integrity:

* Copy code

python integrity\_checker.py -d /path/to/directory -o output.json

Verify Files from Input Checksums:

* Copy code

python integrity\_checker.py -i input.json

8. Conclusion

The Integrity Monitoring project helps in verifying the integrity of files by detecting any unauthorized changes. It's a useful tool for maintaining system security.

9. Recommendations

Regularly update the trusted database with verified checksums.

Consider supporting more hashing algorithms in the future.

Implement automated alerts for any detected discrepancies.

10. References

Python Logging Documentation

ConfigParser Documentation

Hashlib Documentation

Argparse Documentation

11. Appendices

* Steps:
* Example Configuration File
* make file
* Copy code
* [integrity\_checker]
* trusted\_db = trusted\_hashes.txt
* algorithm = sha256

This simple text report provides an overview of the Integrity Monitoring project, covering its purpose, implementation, and usage. The project is a practical tool for detecting unauthorized changes to system files.