International Journal of Scientific Research and Engineering Development-– Volume 8 Issue 2, Mar-Apr 2025
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

FILE INTEGRITY CHECKER USING CRYPTO GRAPHIC HASH FUNCTION

[1]J Jenisha Joseph, [2]Dr R Ravi

[1]Student, [2]Associate Professor,
[1]Electronics And Communication Engineering,
[2]Department Of Computer Science And Engineering,
[1]Francis Xavier Engineering College, Tirunelveli, India

Abstract
In cybersecurity, ensuring file integrity is essential for detecting unauthorized modifications, malware infections, and system intrusions.This project offers a Python-based File Integrity Checker that tracks file modifications using cryptographic hashing techniques. The application calculates, saves, and periodically validates file hashes, notifying users of any unauthorized changes found. This solution facilitates forensic investigations, improves data security, and guarantees adherence to cybersecurity regulations. The suggested approach's efficiency and portability make it appropriate for a range of cybersecurity uses, such as malware detection, server monitoring, and compliance checking. Furthermore, the program makes use of cryptographic hashing [5] to guarantee that file verification is impervious to manipulation and attacks, which makes it a crucial part of contemporary security systems. In addition, the project examines practical uses of file integrity verification and talks about possible improvements including real-time monitoring and interaction with more extensive security framework

Keywords—Cybersecurity, Hashing, File Integrity, Python, Digital Forensics, Data Protection, Security Monitoring.

I. Introduction

In order to avoid unauthorized changes that could result in system vulnerabilities, file integrity [3] must be maintained. In order to hide actions or insert malicious code, attackers frequently alter configuration files, or executable programs. By creating and maintaining cryptographic hashes for files, a File Integrity Checker (FIC) protects data security and enables users to identify changes through verification checks.

The goal of this project is to create an automated integrity monitoring system using a Python-based FIC. The system saves calculated hashes for further verification and supports a number of hashing [5] methods, including SHA-256 [10]. This tool can be used by organizations for forensic analysis, compliance, and cyber threat defense. This method is a crucial component of contemporary cybersecurity frameworks since it proactively detects unauthorized file modifications, in contrast to standard antivirus software, which only detects known dangers.

Applications for file integrity [3] checking can be found in many different sectors. Healthcare providers employ it to guarantee patient data confidentiality, while financial institutions rely on it to safeguard sensitive transaction records. File integrity monitoring is used by government organizations to protect sensitive data from online espionage. Integrating file integrity [3] checking into cybersecurity processes is becoming the norm as cyber threats continue to change.

Moreover, an essential part of incident response plans is file integrity [3] monitoring. Integrity logs are used by investigators to ascertain whether important system files have been compromised in the event of a security attack. Organizations can enhance security protocols and identify unwanted access before serious harm is done by incorporating hash-based verification.

II.Literature Review

Numerous studies emphasize how crucial file integrity [3] monitoring is to cybersecurity. For file verification, hashing [5] algorithms like MD5 [1], SHA-1, and SHA-256 [10] are frequently employed (Rivest, 1992). Advanced security solutions are offered by file integrity [3] monitoring tools like Tripwire and OSSEC; however, greater flexibility and automation are possible with a lightweight Python-based solution. Digital signatures, data encryption, and password storage are just a few of the uses for hash functions, which are an essential part of cryptographic security.

Schneier's (1996) research highlights the significance of cryptographic hashes in guaranteeing the validity of data. Automated monitoring solutions that combine hashing [5] with real-time alerts for intrusion detection have been investigated in recent studies (Kim et al., 2020). This project expands on these ideas by putting in place a stand-alone Python hash-based verification tool. Furthermore, current solutions frequently call for a great deal of setting, but this tool offers a straightforward yet efficient method of file integrity [3] monitoring.

According to some studies, file integrity [3] monitoring ought to be used in conjunction with other security protocols like encryption and access control. Patel et al. (2019) investigated hybrid models that combine anomaly detection based on machine learning with file integrity [3]

monitoring. These techniques add complexity and computational overhead even while they improve security. By adopting a basic approach, this project guarantees effectiveness and usability for every user.
Monitoring file integrity [3] is another crucial component of regulatory compliance. In order to protect data confidentiality and stop unwanted access, standards like HIPAA, GDPR, and PCI-DSS require the usage of integrity checks. In compliance-driven settings, implementing file integrity [3] verification helps firms avoid expensive fines for non-compliance while also enhancing security.

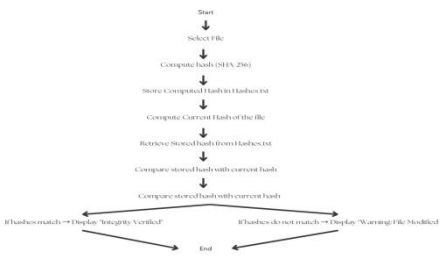
III.Proposed Method

The proposed File Integrity Checker consists of three main components:

- Hash Generation - Computes the cryptographic hash of a file using SHA-256 [10].
- Hash Storage - Saves the computed hash in a file for later reference.
- Integrity Verification - Compares the current hash of a file against the stored hash to detect changes.

The program notifies users when a file has been altered during the verification process. Software integrity checks, forensic investigations, and server security can all benefit from the system.
Flowchart for the Process

The File Integrity Checker's operation is depicted in the flowchart below:



- A file is chosen by the user to be hashed.
- The cryptographic hash of the file is computed by the system.
- A distinct record file contains the hash.
- The system recalculates the file's hash during verification.
- The stored hash and the new hash are contrasted.

The file remains unaltered if the hashes match; if otherwise, an alert is raised.To guarantee precise problem detection in a range of environmental circumstances, the system must go through extensive testing and calibration in real-world settings. This embedded system will provide a proactive and efficient way to preserve the functionality and health of PV modules, increase system dependability, and reduce downtime after it has been tested and improved. Predictive maintenance features that use past data to foresee module failures before they happen and interaction with wider IoT networks are possible future improvements.

IV.Hardware and Software Implementation

A. Hardware Requirements

Since the File Integrity Checker is made to function well on any common computer system, it doesn't require any specific hardware to be implemented. A computer with a dual-core processor, at least 4GB of RAM, and enough storage space to hold hash records are the very minimum system requirements. Because Python is installed on Linux, macOS, and Windows, the program can run on any of these operating systems.

B. Software Implementation

The File Integrity Checker is implemented in Python using the required libraries, such as argparse, hashlib, and os. The application has little graphical overhead because it is executed in a command-line environment. Users may quickly and easily generate, save, and validate file hashes thanks to the software's effective and intuitive architecture.

Development Environment:
Because of its robust extensions, debugging capabilities, and smooth Python integration, the tool is created in Visual Studio Code (VS Code). Virtual environments are used in the project's structure to efficiently handle dependencies.

Used Python Libraries:

hashlib: Offers cryptographic hashing [5] functions for calculating file integrity [3] hashes, including SHA-256 [10].
By enabling the command-line interface, argparse makes it simple for users to save and validate file hashes.
OS: Manages file activities and makes sure that various operating systems are compatible.

Execution and Testing: To guarantee stability and compatibility, the program was tested on a variety of operating systems, including Linux and Windows. The following are steps in the execution process:

- Executing the script within a terminal that supports Python.

- Storing file hashes in a specific text document called "hashes.txt."
- To find any illegal modifications, compare stored hashes to recalculated hashes.

Security considerations:
By not storing the original file data, the software protects user privacy and lowers the possibility of sensitive material being revealed. Furthermore, SHA-256 [10] offers a secure cryptographic algorithm that is impervious to manipulation and collisions.

Software Component	Version	Purpose
Python	3.10	Core Programming
Visual Studio Code	1.76+	Development Environment
hashlib Library	Built-in	Cryptographic Hashing
Argparse Library	Built-in	Command-Line Interface (CLI)
os Library	Built-in	File Handling and OS Access

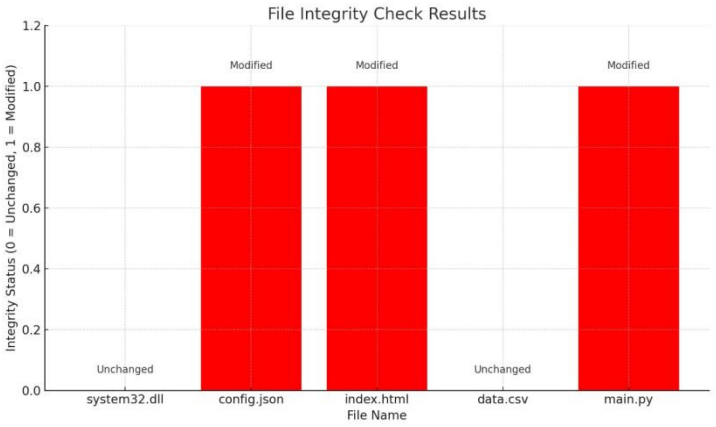
V.Result and Discussion

To evaluate the tool's accuracy in identifying changes, it was tested on a number of files. The findings verified that:

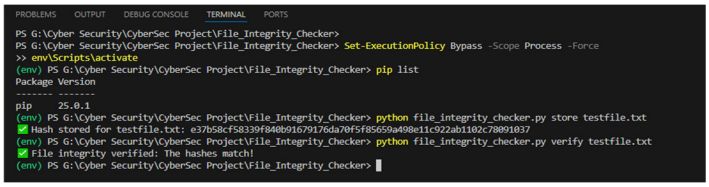
- The utility accurately checks the integrity of a file when it stays unaltered.
- The program recognizes when a file has been altered and notifies the user.

The File Integrity Checker is useful in situations like these because of these features:

- Finding Unauthorized Changes: Making sure that hackers haven't changed system files.
- Malware detection is the process of locating modifications to important files.
- Audits of compliance and security make sure that files adhere to security guidelines.



VI.Output



VII.Conclusion

An efficient tool for tracking file changes and identifying unwanted modifications is the File Integrity Checker. It offers a trustworthy way to guarantee data confidentiality and integrity by utilizing cryptographic hashing [5]. By enabling users to confirm the legitimacy of files and stop tampering, this project highlights the significance of proactive cybersecurity measures.
This method is especially helpful for businesses that need to safeguard confidential information, keep up with security regulations, and identify online dangers. In contrast to conventional security solutions, file integrity [3] monitoring offers a separate verification technique that improves system security as a whole.
Integration with cutting-edge security frameworks, automated integrity checks, and real-time monitoring are possible future enhancements. Non-technical consumers might also find the program easier to use if it had a graphical user interface (GUI). As it continues to develop, file integrity [3] verification will continue to be an essential part of contemporary cybersecurity tactics.

VIII.References

[1].Rivest, R. (1992). The MD5 Message-Digest Algorithm. RFC 1321.

[2].Schneier, B. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. Wiley.

[3].Kim, J., Lee, H., & Park, S. (2020). Automated File Integrity Monitoring Systems for Cybersecurity. Journal of Information Security, 45(3), 120-135.

[4].Patel, A., Mehta, R., & Singh, S. (2019). Hybrid Approaches for File Integrity Monitoring in Modern Security Systems. International Journal of Cyber Security, 12(4), 89-105.

[5].Stallings, W. (2017). Cryptography and Network Security: Principles and Practice (7th ed.). Pearson.

[6].Viega, J., & McGraw, G. (2001). Building Secure Software: How to Avoid Security Problems the Right Way. Addison-Wesley.

[7].Anderson, R. (2020). Security Engineering: A Guide to Building Dependable Distributed Systems (3rd ed.). Wiley.

[8].ISO/IEC 27001. (2013). Information Security Management Systems – Requirements. International Organization for Standardization.

[9].PCI Security Standards Council. (2018). Payment Card Industry Data Security Standard (PCI-DSS) v3.2.1. Retrieved from www.pcisecuritystandards.org

[10].NIST. (2015). Secure Hash Standard (SHS) – FIPS PUB 180-4. National Institute of Standards and Technology, U.S. Department of Commerce.

