TASK-1
File Integrity Checker Tool

Introduction

In cybersecurity, ensuring that files are not modified without authorization is a critical requirement. File Integrity Checking is a security mechanism used to detect changes in files by comparing cryptographic hash values. Even a small modification in a file result in a completely different hash value, making hashing a reliable technique for integrity verification.

This project implements a **File Integrity Checker** using Python. The tool calculates hash values of files in a directory, stores them securely, and later compares them to detect file modification, deletion, or tampering.

Aim

The aim of this project is to develop a Python-based tool that monitors file integrity by calculating and comparing cryptographic hash values.

Objectives

- To calculate hash values of files using SHA-256
- To store original hash values securely
- To detect file modification or deletion
- To understand real-world file integrity monitoring mechanisms

Tools & Technologies Used

Component	Description
Operating System	Kali Linux
Programming Language	Python 3
Libraries	hashlib, os, json
Editor	Nano
Hash Algorithm	SHA-256

Step-by-Step Procedure:

Step 1: Created Python Script

nano file_integrity_checker.py

Step 2: Created Python Script

nano file_integrity_checker.py

Step 3: Code

```
import hashlib
import os
import json

HASH_FILE = "hashes.json"

def calculate_hash(file_path):
    sha256 = hashlib.sha256()
    try:
        with open(file_path, "rb") as f:
            while chunk := f.read(4096):
                sha256.update(chunk)
        return sha256.hexdigest()
    except FileNotFoundError:
        return None

def store_hashes(directory):
    hashes = {}
    for root, dirs, files in os.walk(directory):
        for file in files:
            file_path = os.path.join(root, file)
            file_hash = calculate_hash(file_path)
            if file_hash:
                hashes[file_path] = file_hash

    with open(HASH_FILE, "w") as f:
        json.dump(hashes, f, indent=4)

    print("[+] Hashes stored successfully.")

def check_integrity():
    try:
        with open(HASH_FILE, "r") as f:
            old_hashes = json.load(f)
    except FileNotFoundError:
        print("[-] Hash file not found.")
        return

    print("\nIntegrity Check Results:\n")
    for file_path, old_hash in old_hashes.items():
        new_hash = calculate_hash(file_path)

        if new_hash is None:
            print(f"[!] File missing: {file_path}")
        elif new_hash != old_hash:
            print(f"[⚠] File modified: {file_path}")
        else:
            print(f"[✓] File unchanged: {file_path}")

def main():
    print("\nFILE INTEGRITY CHECKER")
    print("1. Store file hashes")
    print("2. Check file integrity")

    choice = input("Enter choice (1/2): ")

    if choice == "1":
        directory = input("Enter directory path to monitor: ")
        store_hashes(directory)
    elif choice == "2":
        check_integrity()
    else:
        print("Invalid choice.")

if __name__ == "__main__":
    main()
```

Step 4: Created a Test File

cd /home/kali/Documents

nano test.txt

Step 5: Run the Tool (Store Hashes)

python3 ~/file_integrity_checker.py

Input:1
Step 6: Checking File Integrity

python3 ~/file_integrity_checker.py

Input: 2

Output:

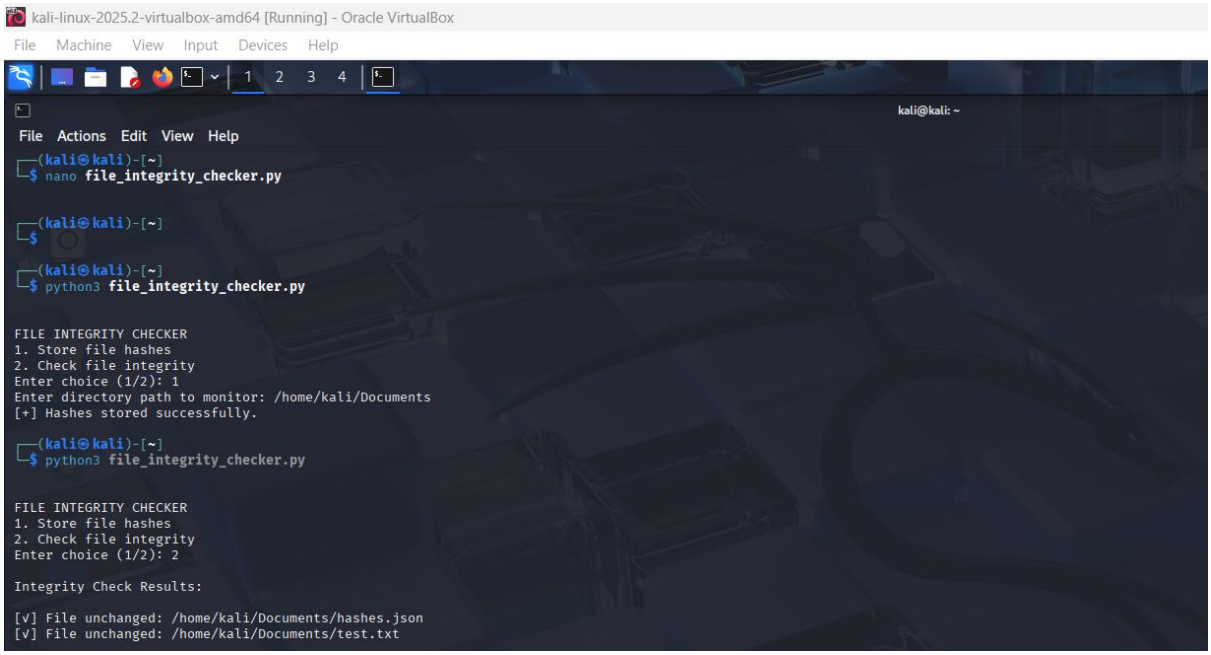
File unchanged → file integrity maintained

File modified → unauthorized change detected

File missing → file deleted or removed

Result:

The File Integrity Checker successfully detected file integrity status using SHA-256 hashing. The tool correctly identified unchanged, modified, and missing files, demonstrating effective file integrity monitoring.



Applications

- Cybersecurity monitoring
- SOC environments
- Malware detection
- Digital forensics
- Compliance auditing

Conclusion

This project demonstrates the practical implementation of a File Integrity Checker using Python. By leveraging cryptographic hashing, the tool ensures reliable detection of unauthorized file modifications. The project provides hands-on experience with core cybersecurity concepts and real-world security monitoring techniques.