File Integrity Checker

Generated on 2025-09-02 10:23:18

Objective: Detect file tampering using SHA256 or MD5 hashing.

Tools: Python, hashlib, cron

Mini Guide

1. Write a Python script to calculate and store file hashes.
2. Schedule regular hash checking and compare with originals.
3. Alert user if hash mismatches are found.
4. Log all activities in a secure file.

Usage Instructions

Store initial hashes:

```
python checker.py store important.txt --algo sha256
```

Verify hashes later:

```
python checker.py verify
```

Schedule periodic runs with cron (Linux/macOS) or Task Scheduler (Windows).

Security Features

- Supports SHA256 (default) and MD5 hashing.
- Logs all events in integrity.log.
- Detects missing files or tampered content.
- Example log entries are shown in the next section.

checker.py

```
import os
import hashlib
import json
import datetime
LOG_FILE = "integrity.log"
HASH_FILE = "hashes.json"
def calculate_hash(file_path, algo="sha256"):
    h = hashlib.new(algo)
    with open(file_path, "rb") as f:
        for chunk in iter(lambda: f.read(4096), b""):
            h.update(chunk)
    return h.hexdigest()
def store_hashes(files, algo="sha256"):
    hashes = {}
    for f in files:
        if os.path.isfile(f):
            hashes[f] = calculate_hash(f, algo)
    with open(HASH_FILE, "w") as out:
        json.dump({"algo": algo, "hashes": hashes}, out, indent=2)
    log("Stored initial hashes for files: " + ", ".join(files))
def verify_hashes():
    if not os.path.exists(HASH_FILE):
        log("Hash file not found.")
        return
    with open(HASH_FILE) as f:
        data = json.load(f)
    algo = data.get("algo", "sha256")
    original = data["hashes"]
    for f, h in original.items():
        if os.path.isfile(f):
            current = calculate_hash(f, algo)
            if current != h:
                log(f"[ALERT] Hash mismatch detected for {f}!")
            else:
                log(f"OK: {f} unchanged.")
        else:
            log(f"[WARNING] File missing: {f}")
def log(message):
    ts = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    with open(LOG_FILE, "a") as out:
        out.write(f"[{ts}] {message}\n")
    print(f"[{ts}] {message}")
if __name__ == "__main__":
    import argparse
    parser = argparse.ArgumentParser(description="File Integrity Checker")
    parser.add_argument("action", choices=["store", "verify"], help="Store initial or verify
hashes")
    parser.add_argument("files", nargs="*", help="Files to process (for 'store' only)")
    parser.add_argument("--algo", default="sha256", choices=["sha256","md5"], help="Hash algorithm")
    args = parser.parse_args()
    if args.action == "store":
        if not args.files:
            print("Please provide files to store hashes.")
        else:
            store_hashes(args.files, args.algo)
    elif args.action == "verify":
        verify_hashes()
```

Sample Log Output

[2025-09-02 06:30:00] Stored initial hashes for files: important.txt

[2025-09-02 06:31:00] OK: important.txt unchanged.

[2025-09-02 06:32:00] [ALERT] Hash mismatch detected for important.txt!