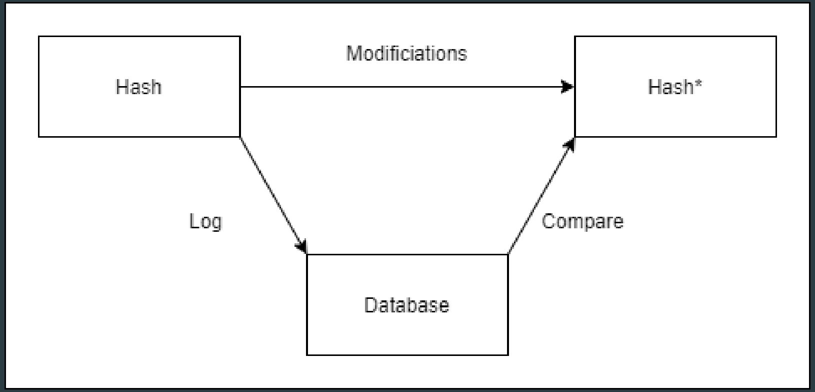
File Integrity Monitor

Contents

- Introduction to the basic functionality of the software and justification for its existence.
- Implementation details and screeshot(s).
- Sources

What does it do?

- It is a software that logs and maintains a database of file hashes. The tool can be used for the purpose of monitoring file integrity.
- Every time a file is modified its hash value is changed. If the hash value has changed, we can assume that the file has been modified in some way.
- When a file is modified, we can compare its new hash to the previously logged hash value, to verify that it has been modified.



Basic operations

Where could it potentially be used?

- Here are few ideas, where this software or the idea behind it could be used for:
  - It could be used to verify the integrity of the files downloaded from the internet. Malicious parties could try to modify downloaded files using *man-in-the-middle attack*.
  - It could be used by server databases to log the changes of the files stored in them. Can be used to expose unintended changes and malicious activity.
  - It could be used for the purpose of verification of file versions.

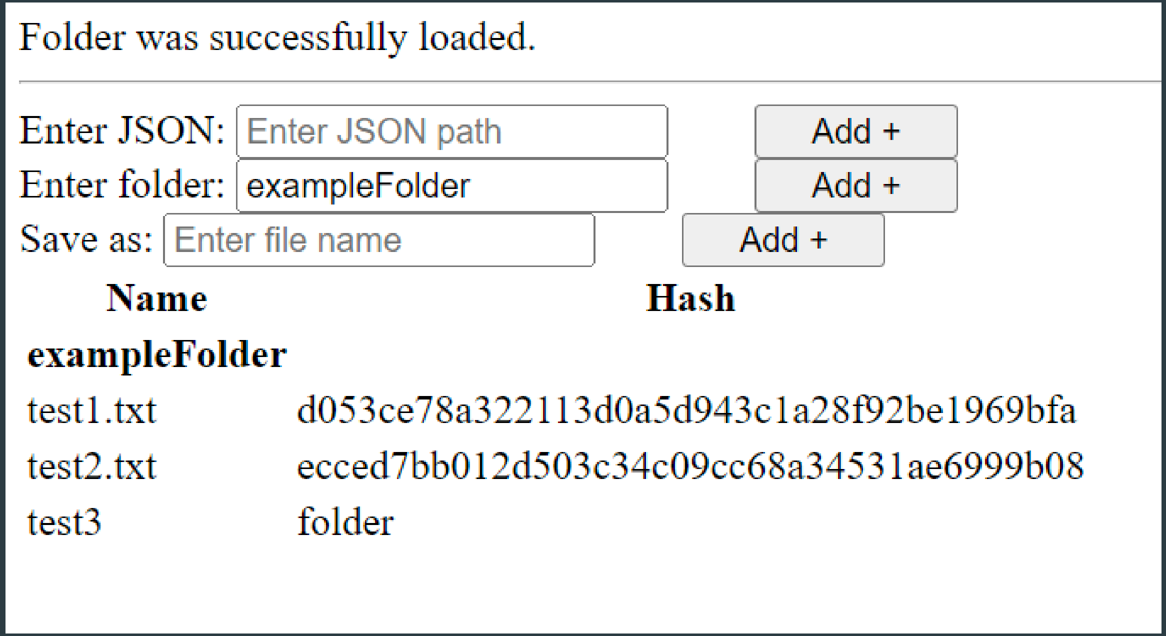
How is it implemented?

- It is written using Python and JavaScript languages.
- It uses Python *eel* architecture, which can be used to build user-interfaces using HTML.
- The program is tested using Python’s *unittest* and JavaScript’s *Mocha/Chai* frameworks.
- Currently calculates the hashes using SHA-1, however, this is just a placeholder solution.
- Hashes are stored into a JSON a file.

Current Progress

- Still a work in progress. Approximately 60 % away from the final version, with most of the backend features implemented and tested.
- Can almost be considered the *minimum viable product*.
- The frontend features are currently in progress.

Screenshots



Folder was successfully loaded.

Enter JSON: Enter JSON path

Enter folder: exampleFolder

Save as: Enter file name

Name	Hash
exampleFolder	
test1.txt	d053ce78a322113d0a5d943c1a28f92be1969bfa
test2.txt	ecced7bb012d503c34c09cc68a34531ae6999b08
test3	folder

Current UI

Personal Criticisms of the Software

- An attacker can currently modify the file hash and directly write the new hash to the JSON database, fooling the system. Can be avoided using encryption.
- An attacker could modify the files and fool the software by calculating a colliding hash to the file. Can be achieved by adding malicious modifications to the file and modifying the file in some invisible area until the hash function calculates a matching hash. Can be avoided by using salting and stronger hashing methods.
- These will be mitigated later.

Thank You!

Sources

- https://www.logsign.com/blog/how-to-check-the-integrity-of-a-file/
- https://www.imperva.com/learn/application-security/man-in-the-middle-attack-mitm/
- https://www.beyondtrust.com/resources/glossary/file-integrity-monitoring
- https://www.comparitech.com/blog/information-security/what-is-a-collision-attack/