Create Hash Values

Project Description

As a security analyst, one of your key responsibilities is to implement security controls that protect organizations from a wide range of threats. One essential control is **hashing**.

Consider this scenario:

A malicious program disguises itself as a legitimate one. Even if just one line of code is changed, its hash value will differ from that of the original file. This difference helps security teams detect tampering and take action quickly. While there are many tools available to compare hash values, a skilled security analyst should understand how to perform hash comparisons manually using Linux commands.

Generate hashes for files

First checking the `/home/analyst` directory, which contains two files: `file1.txt` and `file2.txt`. Using the `cat` command, displayed the contents of both files, which appeared to be identical. However, to determine whether the files are truly the same, generated SHA-256 hash values for each using the `sha256sum` command. Despite the identical content shown by `cat`, the resulting hashes were different:

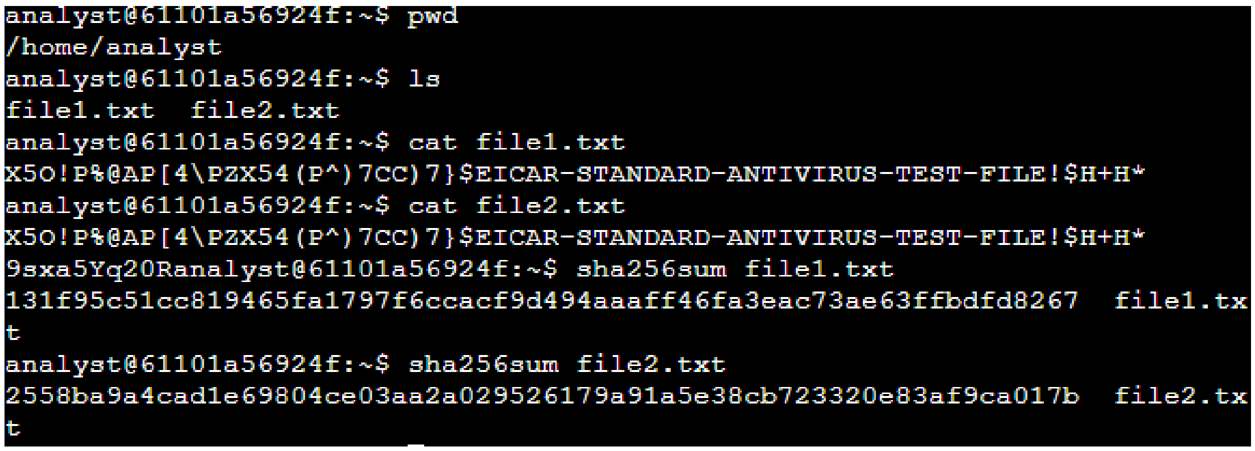
`file1.txt` produced a hash of

`131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267`.

While `file2.txt` generated

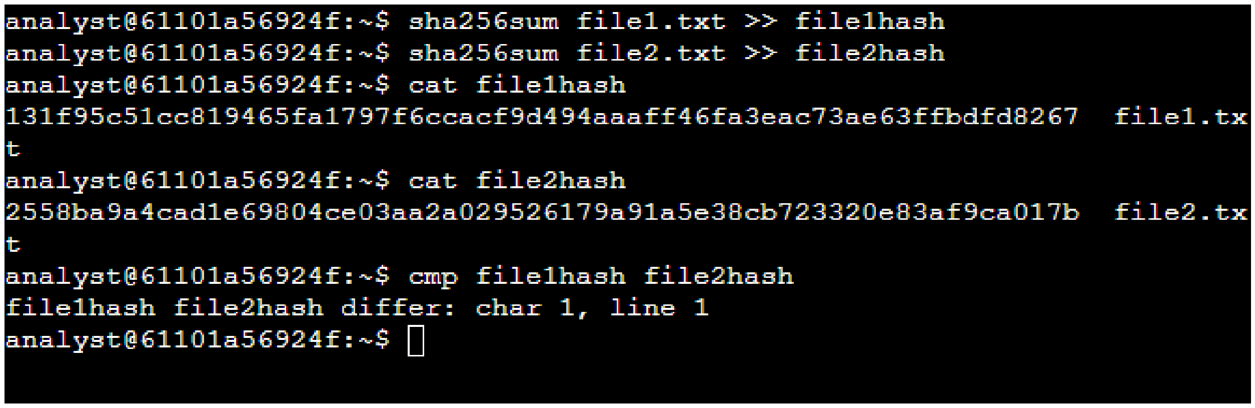
`2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b`.

This confirms that the two files are not exactly the same, even if they appear so when viewed with `cat`.


```
analyst@61101a56924f:~$ pwd
/home/analyst
analyst@61101a56924f:~$ ls
file1.txt  file2.txt
analyst@61101a56924f:~$ cat file1.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
analyst@61101a56924f:~$ cat file2.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
9sxa5Yq20Ranalyst@61101a56924f:~$ sha256sum file1.txt
131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267  file1.tx
t
analyst@61101a56924f:~$ sha256sum file2.txt
2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b  file2.tx
t
```

Compare Hashes

Now generated SHA-256 hash values for `file1.txt` and `file2.txt` and redirected the output to two separate files: `file1hash` and `file2hash`. Using the `cat` command, you displayed the contents of both hash files to compare the values manually. Even though the contents of the original text files appeared identical, the hash values written to `file1hash` and `file2hash` were different. To confirm this, used the `cmp` command, which compares files byte by byte. The output indicated a difference at the very first character of the first line, confirming that the two hash files are not the same. This proves that `file1.txt` and `file2.txt` are different at some level, even if not visibly obvious.


```
analyst@61101a56924f:~$ sha256sum file1.txt >> file1hash
analyst@61101a56924f:~$ sha256sum file2.txt >> file2hash
analyst@61101a56924f:~$ cat file1hash
131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267  file1.tx
t
analyst@61101a56924f:~$ cat file2hash
2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b  file2.tx
t
analyst@61101a56924f:~$ cmp file1hash file2hash
file1hash file2hash differ: char 1, line 1
analyst@61101a56924f:~$ 
```