Compare Hash values

As a security analyst, you'll need to implement security controls to protect organizations against a range of threats.

That's where hashing comes in. Previously, you learned that a hash function is an algorithm that produces a code that can't be decrypted. Hash functions are used to uniquely identify the contents of a file so that you can check whether it has been modified. This code provides a unique identifier known as a hash value or digest.

For example, a malicious program may mimic an original program. If one code line is different from the original program, it produces a different hash value. Security teams can then identify the malicious program and work to mitigate the risk.

Many tools are available to compare hashes for various scenarios. But for a security analyst it's important to know how to manually compare hashes.

In this lab activity, we'll create hash values for two files and use Linux commands to manually examine the differences.

In this scenario, we need to investigate whether two files are identical or different.

Here's how you'll do this task: First, you'll display the contents of two files and create hashes for each file. Next, you'll examine the hashes and compare them.

Generate hashes for files

The lab starts in your home directory, /home/analyst, as the current working directory. This directory contains two files file1.txt and file2.txt, which contain different data.

In this task, you need to display the contents of each of these files. You'll then generate a hash value for each of these files and send the values to new files, which you'll use to examine the differences in these values later.

Use the ls command to list the contents of the directory.

Two files, file1.txt and file2.txt, are listed.

Use the cat command to display the contents of the file1.txt and file2.txt:

```
analyst@0c54d261cee8:~$ ls
file1.txt  file2.txt
analyst@0c54d261cee8:~$ cat file1.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
analyst@0c54d261cee8:~$ cat file2.txt
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
9sxa5Yq20Ranalyst@0c54d261cee8:~$
```

Although the contents of both files appear identical when you use the cat command, you need to generate the hash for each file to determine if the files are actually different.

Use the sha256sum command to generate the hash of the file1.txt file:

Use the sha256sum command to generate the hash of the file2.txt file:

```
9sxa5Yq20Ranalyst@0c54d261cee8:~$ sha256sum file1.txt
131f95c51cc819465fa1797f6ccacf9d494aaaff46fa3eac73ae63ffbdfd8267  file1.txt
analyst@0c54d261cee8:~$ sha256sum file2.txt
2558ba9a4cad1e69804ce03aa2a029526179a91a5e38cb723320e83af9ca017b  file2.txt
```

Compare hashes

In this task, you'll write the hashes to two separate files and then compare them to find the difference.

Use the sha256sum command to generate the hash of the file1.txt file, and send the output to a new file called file1hash:

Use the sha256sum command to generate the hash of the file2.txt file, and send the output to a new file called file2hash:

```
analyst@0c54d261cee8:~$ sha256sum file1.txt >> file1hash
analyst@0c54d261cee8:~$ sha256sum file2.txt >> file2hash
```

Now, you can use the cmp command to compare the two files byte by byte. If a difference is found, the command reports the byte and line number where the first difference is found.

Use the cmp command to highlight the differences in the file1hash and file2hash files :the hash

```
analyst@0c54d261cee8:~$ cmp file1hash file2hash
file1hash file2hash differ: char 1, line 1
```

Summary

Though the contents of both files appear to be identical, only hash values of each file that can determine if they are the same or not.